

HARDWARE BACKDOOR ANALYSIS TOOLKIT

(HBAT)



By

Bushra Batool

Asma Rashid

Tehreem Fatima

Izza Batool

Supervised by:

Dr. Haider Abbas

Submitted to

Faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad,

in partial fulfillment for the requirements of

B.E Degree in Electrical (Telecommunication) Engineering.

June 2023

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

“Hardware Backdoor Analysis Toolkit”

is carried out by

Bushra Batool, Asma Rashid, Tehreem Fatima and Izza Batool

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical (Telecom.) Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Dr. Haider Abbas

HoD, Information Security Department

Department of Electrical Engineering

Military College of Signals (MCS)

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

Date: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan Wa Ta'ala is the sole guidance in all domains.

We would like to express our gratitude to our parents, teachers and our supervisor Dr. Haider

Abbas for his professional guidance and support throughout this thesis.

This is dedicated to individuals who exhibit diligence and hard work, those who display unwavering devotion and persistence, those who demonstrate resilience when confronted with misfortune, and all those who strive to bring positivity in times of darkness. This is dedicated to those who never give up.

ACRONYMS

B

BAT Binary Analysis Toolkit

C

CLI Command Line Interface

CVE Common Vulnerabilities and Exposures

E

eMMC Embedded Multi Media Card

F

FAT Firmware Analysis Toolkit

FACT Firmware Analysis and Comparison Toolkit

G

gzip GNU zip

H

HBAT Hardware Backdoor Analysis Toolkit

I

IoT Internet of Things

IP Internet Protocol.

J

JFFS2 Journaling Flash File System version 2

JTAG Joint Test Action Group

N

NAS Nordic Automation Systems

O

OS Operating System

Q

QEMU Quick Emulator

R

ROM Read Only Memory.

S

SDG Sustainable Development Goal.

U

UART Universal Asynchronous Receiver-Transmitter.

UI User Interface.

Plagiarism Certificate (Turnitin Report)

This thesis has ____ similarity index. Turnitin report endorsed by Supervisor is attached.

Bushra Batool
NUST Serial no 00000294137

Asma Rashid
NUST Serial no 00000282833

Tehreem Fatima
NUST Serial no 00000325369

Izza Batool
NUST Serial no 00000291543

Signature of Supervisor

ABSTRACT

Technology has become an indispensable part of society and everyday life. Amidst the emerging technology, embedded devices have made a remarkably huge entry into the market, with the most popular category being IoT devices. These devices have firmware embedded in them. However, the firmware might be subjected to some vulnerabilities. This is a crucial issue in the present-day world that cannot be avoided knowing the increase in potential attack surface. Considering the importance of this issue, it is highly essential to build measures for the protection and prevention of these devices from the attacker. In the present world several devices like routers, printers, and smartphones have undergone vulnerability analysis and analysis reports of these devices are available online. A variety of analysis tools are available online but the shortcomings they face are their particularity. Most tools are intended for a specific aspect of firmware analysis. However, the aim of our project is to provide a compact backdoor analysis toolkit that would cater to IP cameras of vendors locally and internationally and perform a detailed security analysis on them.

In this thesis, we developed a complete framework combining online existing tools using static and dynamic analysis to verify the complications and challenges that are faced in large-scale security analysis. We have collected firmware images belonging to both local and international vendors.

TABLE OF CONTENTS

1	Introduction.....	01
1.1	Overview.....	01
1.1.1	Firmware.....	02
1.1.2	Firmware Segments.....	03
1.1.2.1	Initial Bootloader.....	03
1.1.2.2	Second Level Boot Loader.....	03
1.1.2.3	Boot Loaders Environment variables.....	03
1.1.2.4	Linux Kernel.....	03
1.1.2.5	Root File System.....	04
1.1.3	Need for Firmware Analysis.....	05
1.2	Problem Statement.....	05.
1.3	Proposed Solution.....	05
1.3.1	Software.....	05
1.3.2	Hardware.....	05
1.4	Contributions.....	07
1.5	Challenges.....	08
1.5.1	Acquiring the Firmware.....	08
1.5.2	Acquiring the Source Code.....	08
1.5.3	Encrypted Firmware.....	08

1.5.4	Unpacking Firmware.....	08
1.5.5	Identifying JTAG Pins.....	09
1.6	Objectives.....	10
1.6.1	General Objectives.....	10
1.6.2	Academic Objectives.....	10
1.7	Scope.....	10
1.8	Deliverables.....	10
1.8.1	Firmware Extraction Tool.....	10
1.8.2	Firmware Decryption Tool.....	10
1.8.3	Vulnerability Analysis Tool.....	11
1.8.4	Detailed Report	11
1.9	Sustainable Development Goals.....	11
1.9.1	Primary SDG.....	11
1.9.2	Secondary SDG.....	11
1.10	Outline.....	11
2	Background.....	12
2.1	Industrial Background.....	12
2.1.1	What is Firmware Analysis.....	12
2.1.2	What is Backdoor Analysis Toolkit?	13
2.2	Existing Solutions and their Drawbacks.....	14

2.2.1	Conventional Security Technologies.....	14
2.2.2	Manual Code Review.....	14
2.2.3	Tools for Static Code Analysis.....	15
2.2.4	Dynamic Analysis Tools.....	15
2.2.5	Machine learning-based tools.....	15
2.2.6	Commercial Solutions.....	15
2.3	Research Papers.....	16
2.3.1	Firmware Analysis and Stimulation.....	16
2.3.2	Survey of Firmware Extraction and Modification Techniques.....	17
3	Methodology.....	18
3.1	Problem.....	18
3.2	Design.....	18
3.2.1	Datasets.....	18
3.3	Pre-Existing Tools.....	20
3.3.1	Binwalk.....	20
3.3.2	Firmadyne.....	20
3.3.3	Firmwalker.....	20
3.3.4	QEMU.....	20
4	Implementation.....	21
4.1	Development Environment.....	21

4.2	Toolkit Design.....	22
4.2.1	Interface.....	22
4.2.2	Architecture.....	22
4.2.3	Supporting Systems.....	23
4.2.4	System Tools.....	24
4.2.5	Dependencies.....	24
4.2.6	Generic.....	25
4.2.7	Static and Dynamic Analysis.....	26
4.2.8	Cross Platform.....	26
4.3	Unpacking and Extraction Tools.....	26
4.3.1	Binwalk.....	26
4.3.2	Gunzip.....	28
4.3.3	Unsquash.....	29
4.3.4	Unlzma.....	29
4.4	Emulation.....	29
4.4.1	QEMU.....	29
4.5	Tp Link Firmware.....	30
4.5.1	Analysis.....	30
4.5.2	Results.....	34
5	Conclusion.....	35

5.1 Limitations.....	35
6 Future Work.....	36
Bibliography.....	38

LIST OF FIGURES

Figure 1: Percentage Ratio of Most Hacked IoT Devices.....02

Figure 2: Firmware segments.....04

Figure 3: Proposed Block Diagram.....07

Figure 4: Identifying JTAG pin.....09

Figure 5: H-BAT Architecture.....23

Figure 6: OS Applications.....25

Chapter 1: Introduction

1.1 Overview

Presently, we live in a world where technology has become a necessity. From a simple smartphone to compact smart watches, from home internet routers to internet protocol cameras installed everywhere; we're heading towards technological advancement at a greater pace than ever. Talking about these technological and internet-connected devices, it is essential to know how these devices work, in order to protect them from cyberattacks.

All these IoT devices possess small, embedded software that makes them specific to their tasks.

This software is called Firmware. Reverse engineering [1] is a vital topic of discussion when it comes to firmware security analysis. It is an ethical way of learning about a device or system by obtaining its information through an authorized party. However, obtaining the firmware itself is a hectic task as the vendors have taken measures to limit access to the firmware source code, or even have not given public access to these firmware images.

According to a report by Kaspersky, 1.5 billion IoT cyberattacks took place in just the initial six months of 2021 [2]. With the advancement in technology, cyber-attacks have greatly impacted society. The interconnection of various devices makes it even more threatening to the data possessed by them. The hacker can access the data of all the devices by just getting access to one of these interconnected devices. Networking companies have security assessment teams that stimulate cyberattacks and reverse-engineer these attacks for vulnerability assessments. This helps in the security assessment of devices to assure their protection against such cyberattacks. Reverse

engineering [1] allows us to find a backdoor to the embedded systems through their firmware and exploiting these backdoors forms the basis of security analysis.

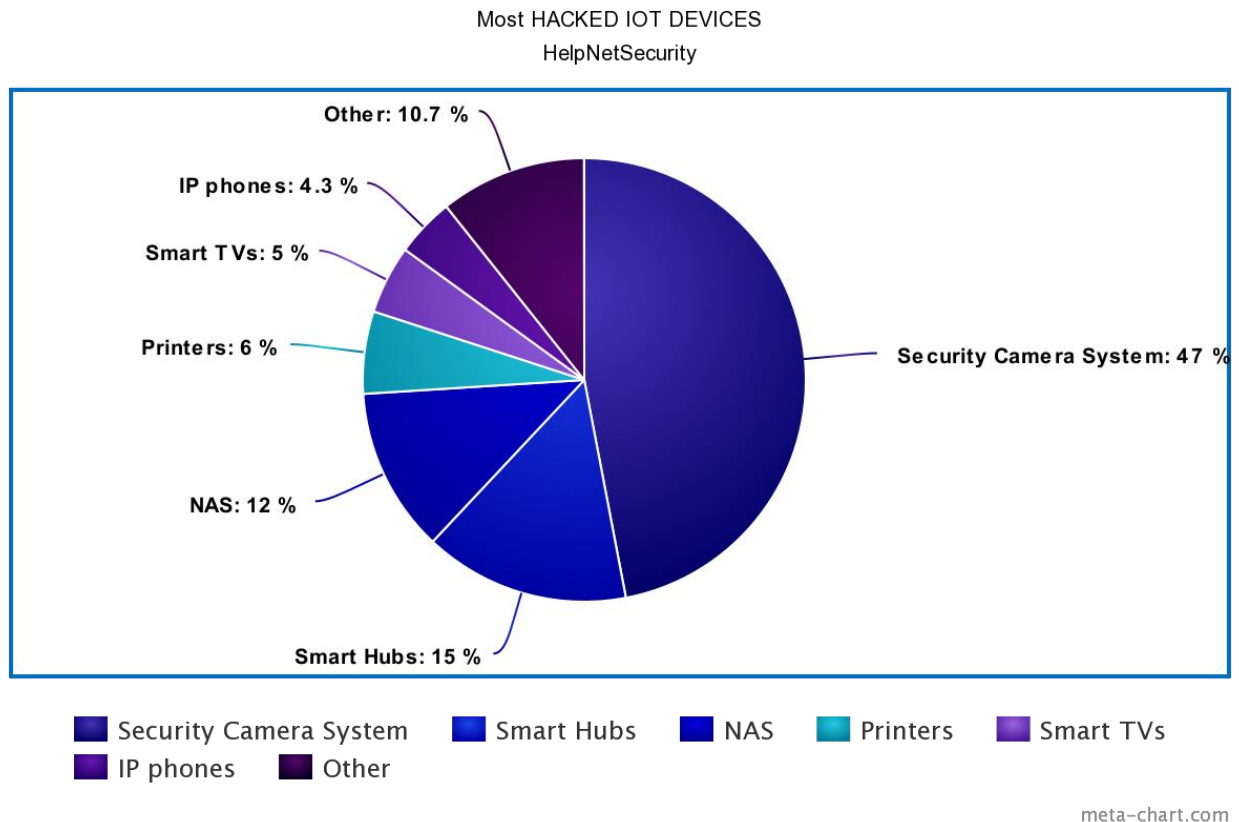


Figure 1: Percentage Ratio of Most Hacked IoT Devices

1.1.1 Firmware

Firmware is a set of codes that are kept on a computer hardware device to accomplish an explicit task that a vendor wants that device to perform.

There exist various fields where firmware has spread its roots including networking devices like routers, switches, smartphones, your laptops, desktops, smartwatches, and cameras.

1.1.2 Firmware Segments

1.1.2.1 Initial Boot Loader:

The initial bootloader is the first phase of firmware that runs when the device is powered on. It initializes the hardware and starts the boot process. *The primary function of the initial boot loader is to locate and load the second-level boot loader.*

1.1.2.2 Second-Level Boot Loader (U-Boot):

The second-level boot loader, also known as the Universal Boot Loader (U-Boot), is a more complex and feature-rich bootloader than the initial boot loader. It is responsible for loading the operating system kernel and other necessary components of the firmware.

1.1.2.3 Boot Loader Environment Variables:

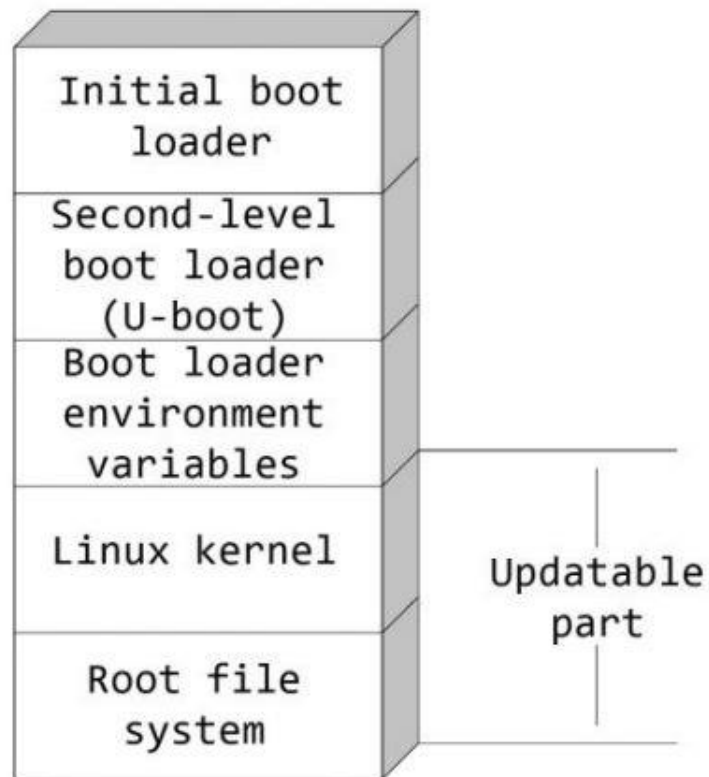
The Boot Loader Environment (BLE) is a section of the firmware that stores configuration settings for the boot loader, such as kernel command line arguments, memory settings, and other system parameters. These variables can be changed by the user to modify the boot process or system behavior.

1.1.2.4 Linux Kernel:

The Linux kernel is the core component of the firmware that manages the system's hardware and provides low-level services to other components of the firmware. It is responsible for managing system resources such as memory, input/output devices, and network interfaces.

1.1.2.5 Root File System:

The root file system is the top-level directory structure of the firmware, which contains all system files and directories necessary for the firmware to function. It includes configuration files, system binaries, and libraries, as well as user data and applications. The root file system is mounted by the Linux kernel during the boot process, and it is the starting point for all file operations in the firmware.



Firmware segments

Figure 2: Firmware segments

1.1.3 Need for Firmware Analysis

As we have seen, the firmware has a huge applicability. It also comes with vulnerabilities which can lead to:

- Classified data disclosures like passwords, API keys, private certificates, etc.
- Conceding devices and corrupting with data.
- Imitating the firmware image with malicious scripts.
- Comprehending the working of the firmware.

1.2 Problem Statement

Large-scale analysis of firmware images requires a great deal of time if done manually through already existing tools like Binwalk [3]. Despite the presence of various tools that are used for the security analysis of firmware-embedded devices, there is a limitation to it. Most tools are very specific in the analysis and only provide a certain aspect of vulnerability assessment. This thesis is intended to provide a compact mod kit that provides an automated analysis of IP cameras and generates a detailed vulnerability assessment report that can act as a basis for improving the device's stability and making future devices resistant to such attacks. Furthermore, this thesis focuses on both the software and hardware aspects of firmware analysis.

1.3 Proposed Solution

1.3.1 Software:

The solution consists of a step-by-step approach, which is as follows:

Stage 1

Data collection and research:

For firmware Analysis, the foremost step is to gather data and information using online search engines regarding the target firmware.

Stage 2

Obtaining firmware:

The following are the possible methods for obtaining firmware images:

- Download it from the vendor's website
- Capture it during the firmware update process of the respective device
- Extract it directly from the hardware

Stage 3

Analyzing firmware:

This step helps determine the type of firmware and its encryption. Entropy analysis [4] is used to identify whether the firmware is encrypted or not. This can be done using binwalk [3].

Stage 4

Extracting File Systems:

This step involves the extraction of file systems from the firmware image which can also be done using binwalk [3]. These file systems include gzip [5], SquashFS [6], CramFS [7], and JFFS2 [8].

Stage 5

Analyzing the contents of the filesystem:

This step involves examining the filesystem contents such as the files and the directories present within a filesystem. This can be done by first mounting the filesystem on your computer or virtual machine [9].

To analyze a filesystem's contents, we can use tools such as Firmwalker [10].

Stage 6

Firmware Emulation:

This stage is the most essential one as it marks the basis for identifying common vulnerabilities and exposures CVE [11]. Emulation [12] can be done using different approaches; full system, partial system, or using a virtual machine [9].

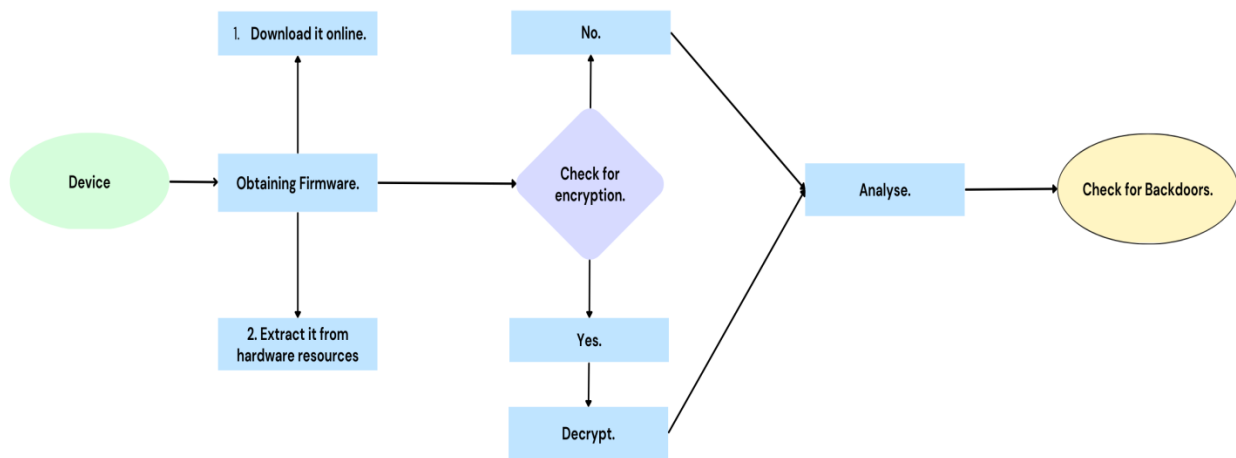


Figure 3: Proposed Block Diagram

1.4 Contribution

Our framework provides an interface for the large-scale security analysis of IP cameras' firmware images. Two major local and international vendors were targeted for this analysis; D-Link and TP-

Link. We gathered various firmware, out of which all were successfully extracted and analyzed. We have also provided a user-friendly interface that can be used by the vendors easily.

1.5 Challenges

With large-scale analysis come huge challenges that need to be faced and addressed in order to run a successful security analysis.

1.5.1 Acquiring the Firmware

Obtaining the firmware itself is a great challenge. The vendors of the devices usually do not provide the firmware of the devices on their websites as mentioned in section 1.1.

Few firmware is present as an open source for the public to access.

Extracting the firmware directly from the device also requires access to the device. It's not feasible to buy or access every single device to extract its firmware.

1.5.2 Acquiring Source Code

Not only obtaining the firmware is an issue that is faced while firmware analysis, but also acquiring the source code of the device. Vendors usually

1.5.3 Encrypted Firmware

The greatest challenge is to analyze the firmware that encrypted. It's a hectic task to find the encryption schemes for the encrypted firmware through brute force attacks.

1.5.4 Unpacking Firmware

It can be a challenge to unpack the firmware because this might generate false positives such as repetitive random binaries that may result in an accident or be created by the vendors for the purpose of obfuscation in order to prevent the hacker from correct extraction of the firmware.

1.5.5 Identifying JTAG Pins

Finding the JTAG pins on the circuit board can usually be quite a laborious act. Some pins may be hidden under other components like the battery or a capacitor. Vendors often obfuscate or disable the JTAG interfaces which can make it a challenge to extract the firmware through the hardware device.

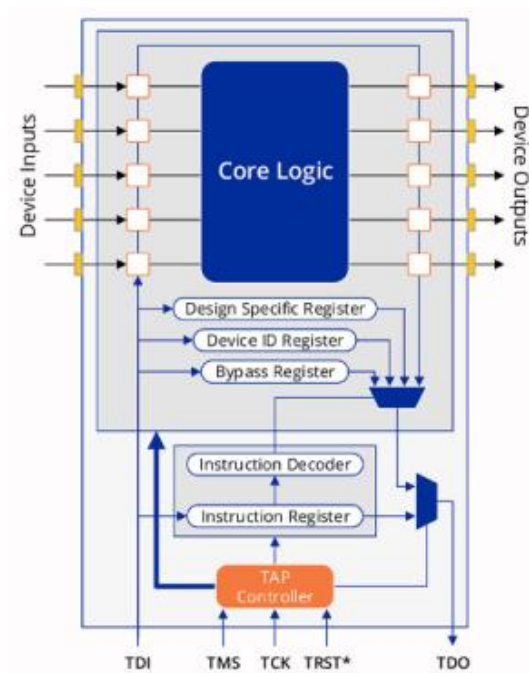


Figure 4: Identifying JTAG Pins

1.6 Objectives

1.6.1 General Objectives:

“To replace manual analysis by automated bulk analysis of IP camera firmware images using well-known tools and forming a customized toolkit for hardware backdoor security analysis.”

1.6.2 Academic Objectives:

- To form a security analysis toolkit for IoT devices
- To implement Cyber Security techniques and enhance knowledge in this domain.
- To lead a successful team by working together
- To benefit society with our project

1.7 Scope

This project is a vital tool for vendors operating IoT devices such as IP cameras. It can be used by vendors to check the security of their IP cameras to ensure customer satisfaction and confidentiality. Its user-friendly interface is feasible use for those who have little to no knowledge of running a security analysis manually.

1.8 Deliverables

1.8.1 Firmware Extraction Tool:

This toolkit serves as a tool that can extract the firmware from a device whether they are encrypted or not.

1.8.2 Firmware Decryption Tool:

The Hardware Backdoor Analysis Toolkit is equipped that it is capable of decrypting encrypted firmware of specific IoT devices.

1.8.3 Vulnerability Analysis Tool:

This toolkit is capable of detecting and alleviating potential backdoors.

1.8.4 Detailed Report:

A thorough report comprising all the results of vulnerability analysis, containing identified vulnerabilities and backdoors.

1.9 Sustainable Development Goals

1.9.1 Primary SDG

- Industry, innovation, and infrastructure

1.9.2 Secondary SDG

- Security of IoT devices against cyberattacks.
- Assisting local vendors in achieving customer satisfaction by protecting this privacy.

1.10 Outline

Chapter 2 contains the literature review and the background and analysis study this thesis is based upon.

Chapter 3 contains the design and development of the project.

Chapter 4 introduces a detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 2: Background

A new product is launched by modifying and enhancing the features of previously launched similar products. A literature review is an important step in the development of an idea for a new product. Likewise, for the development of a product, and for its replacement, related to the traffic system, a detailed study regarding all similar projects is compulsory. Our research is divided into the following points.

- Industrial Background
- Existing solutions and their drawbacks
- Research Papers

2.1 Industrial background

2.1.1 What is Firmware Analysis?

The process of evaluating the software that is embedded in a hardware device is referred to as firmware analysis. Firmware is a sort of software that controls the behavior of a device and is stored in non-volatile memory (such as ROM or flash memory). Firmware can be found in a variety of devices, such as routers, printers, cameras, and even some appliances.

Firmware analysis can be performed for a variety of reasons, including learning how a device operates, discovering security flaws, reverse engineering [1], and debugging. Typically, the procedure entails employing specialized tools and techniques to extract the firmware from the device, analyze it for flaws or defects, and obtain insights into its behavior. Examining the firmware code, looking for specific strings or patterns, or analyzing the firmware's behavior under different settings, can all be a part of this.

2.1.2 What is backdoor Analysis Toolkit?

A backdoor analysis toolkit is a software tool created to assist in identifying and analyzing the potential vulnerabilities in existing firmware. A backdoor is a covert means of getting around standard authentication or security mechanisms in a computer system.

It can be difficult to identify backdoors in software or hardware systems because attackers may employ sophisticated techniques to hide their presence. Backdoors can be intentionally or accidentally placed into these systems. A backdoor analysis toolkit employs a range of methods to find and evaluate suspected backdoors, including scanning for anomalous network traffic, checking system logs for suspicious activity, and looking for tampering indications in software or firmware code.

- **Industrial Uses:**

Such a tool could have industrial uses such as identifying and stopping cyberattacks or espionage attempts on business or government computer networks, assessing the security of hardware and software systems, or investigating the strategies and tactics employed by attackers to compromise systems.

The backdoor analysis toolkit's industrial roots can be traced to the growing demand for cybersecurity in sectors including manufacturing, healthcare, finance, and government. Backdoors, a form of malware, pose a serious risk to the security of these sectors since they permit unauthorized access to computer systems.

- **Advancements:**

Cybersecurity professionals created backdoor analysis toolkits, which are computer programs made to find and analyze backdoors in computer systems, to counteract this issue. These toolkits discover backdoors and their properties using a range of methodologies, including static analysis, dynamic analysis, and machine learning algorithms.

The creation of backdoor analysis toolkits is a continuous process since hackers are always improving the methods, they use to get around existing security measures. As a result, to remain ahead of the most recent threats, these toolkits are constantly updated and enhanced.

Backdoor analysis toolkits are now a crucial part of many industrial sectors' cybersecurity arsenals since they offer vital defense against backdoor malware and other online threats.

2.2 Existing Solutions and their Drawbacks

2.2.1 Conventional Security Technologies:

Conventional security technologies, such as firewalls and intrusion detection systems, may not be effective against more complex attacks, but they can assist in detecting some forms of backdoors. Furthermore, they could produce a lot of false positives, which take time to investigate.

2.2.2 Manual Code Review:

Manual code review is a costly and time-consuming technique, but it has the potential to be successful in locating backdoors. This approach, however, can only be used on a small scale and calls for highly qualified individuals.

2.2.3 Tools for Static Code Analysis:

Static code analysis tools are automated programs that examine software's source code without running it. Although these tools can find backdoors, they could produce a lot of false positives and could not be able to find some forms of backdoors that are only active in specific circumstances.

2.2.4 Dynamic Analysis Tools:

Dynamic analysis tools analyze software as it is being used, which can be used to find backdoors that are enabled under specific circumstances. Although these solutions may not be effective against all sorts of backdoors, they can be expensive and time-consuming to set up.

2.2.5 Machine learning-based tools:

These tools analyze data using algorithms, making them useful for finding backdoors. However, these technologies need a lot of training data, and they could be exposed to hostile attempts that trick the algorithm.

2.2.6 Commercial Solutions:

There are several commercial backdoor analysis tools available, but they can be pricey and are not always effective. Furthermore, these tools could not be adaptable or might need specialized knowledge to operate well.

Overall, there isn't a one-size-fits-all approach to backdoor analysis, and organizations may need to employ a variety of tools and strategies to efficiently find and close system backdoors.

2.3 Research Papers

After thorough reading and mutual understanding, we selected a few research papers that would help us to get near our goal.

- Firmware Analysis and Simulation <https://www.exploit-db.com/docs/49201>
- Breaking all the things: a systematic survey of firmware extraction and modification techniques for IoT devices by Sebastia Vasile, David Oswald, and Tom Chothia. <https://research.birmingham.ac.uk/en/publications/breaking-all-the-things-a-systematic-survey-of-firmware-extractio>
- Methods for extracting firmware from OT devices for vulnerability research by Nozomi Networks Labs, 2022. <https://www.nozominetworks.com/blog/methods-for-extracting-firmware-from-ot-devices-for-vulnerability-research/>

2.3.1 Firmware Analysis and Simulation:

Firmware analysis is a procedure that can be a useful tool for finding flaws and vulnerabilities in firmware. One method of performing firmware analysis is to extract the firmware and simulate it with an expert operating system like attifyOS [14]. This can aid in locating potential backdoors, unsafe protocols, and other potential vulnerabilities.

It might be useful to uncover potential vulnerabilities that can be exploited using a web application by simulating firmware on an interface based on a web application. This can assist in locating potential attack pathways and aid in thwarting harmful assaults.

Overall, Firmware analysis can be a useful method for finding firmware flaws and enhancing the general security of systems that use firmware.

2.3.2 Survey of Firmware Extraction and Modification Techniques:

This paper describes a study that examined multiple hardware-based firmware extraction techniques for commonly used products like smart voice assistants, access control and alarm systems, and home automation gadgets. Over 45% of the study revealed, a publicly accessible UART [15] interface sufficed to obtain a firmware dump, although in other circumstances, more sophisticated but still practical techniques like JTAG [13] or eMMC [16] readout were needed.

It also contains a full analysis of the security concept for the Amazon Echo Plus, which covers crucial safeguards against hardware-level breaches. Based on the results of the inquiry, the report suggests countermeasures to weaken the pertinent techniques.

Overall, the study emphasizes the significance of firmware security for commonly used products and offers information on potential weaknesses and extraction techniques for software. Additionally, it highlights the necessity for manufacturers to put strong security measures in place to guard against hardware-level attacks and stop unauthorized access to private data.

Chapter 3: Methodology

3.1 Problem

Over 2200 cyberattacks are borne every single day around the world, which is a considerable amount. Every home or office device that is connected to the internet has a loophole for the hacker to attack through, and so is the case with IP cameras. Despite the presence of tons of security analysis tools for IoT devices, most of them are focused on one certain aspect of analysis such as static analysis, emulation [12], and extraction. Fewer such frameworks are present that provide the vendors with automated security analysis by incorporating existing tools.

3.2 Design

A framework has been implemented for the purpose of obtaining the objective by integrating several security analysis tools like binwalk [3], Firmadyne [17], Firmware Analysis Toolkit (FAT) [18], and QEMU [19].

3.2.1 Dataset

To obtain the firmware, we initially opted for two vendors, one international and one local.

The firmware was obtained in the three following ways:

- Through the website of the vendors
- Contacting the vendor through email
- Obtaining it directly from hardware using JTAG [13]

Obtaining the firmware was a long process and with it came challenges such as no response to emails or unavailability of firmware images on the websites. However, most of the firmware images belonged to D-Link and TP-Link as they were available on the website.

Another considerable aspect of this thesis is that we obtained the firmware images of different versions of the same IP camera in order to validate if the new version possesses the same vulnerability as the previous versions or if has it been addressed and catered for.

The following tables show the statistics of the dataset.

Table 3.2.1 Firmware with respect to vendors

Vendor	Total No. of Firmware
D-Link	3
TP-Link	7

Table 3.2.2 Firmware Versions

D-Link	TP-Link
DCS-935L_A1_FW_1.07.03	TL-WR841N(US)_V14.8
DCS-935L_A1_FW_1.06.02	TL-WR841N(US)_V14.6
DCS-935L_A1_FW_1.10.01	TL-WR841N_V14
	TL-WR841N(US)_V13
	TL-WR841N(US)_V12
	TL-WR841N_V11
	TL-WR841N_V9

3.3 Pre-Existing Tools

3.3.1 Binwalk:

It is a widely used software tool that offers fast and user-friendly functionality for analyzing, extracting, and reverse engineering [1] firmware images. It is particularly known for its signature analysis and extraction capabilities and is often integrated into other analysis tools.

3.3.2 Firmadyne:

Firmadyne [17] is a dynamic analysis tool that utilizes QEMU [19] to emulate the firmware.

3.3.3 Firmwalker

Its function is to search for specific predefined patterns, including sensitive configuration files, saved passwords, IP addresses, scripts, and other information that can be helpful to analysts in identifying and verifying vulnerabilities.

3.3.4 QEMU

QEMU [20] is a versatile system emulator that can support a range of architectures. It is an invaluable tool for testing and debugging vulnerabilities in situations where physical access to firmware on a device is not possible. Firmadyne [17] leverages the power of QEMU [19] to emulate firmware for analysis purposes.

Chapter 4: Implementation

4.1 Development Environment

Operating System:

Ubuntu 22.04

Development:

Programming Language: Python 3.10.6, Bash, HTML, CSS.

IDE: Pycharm, VS Code.

Hardware Platform:

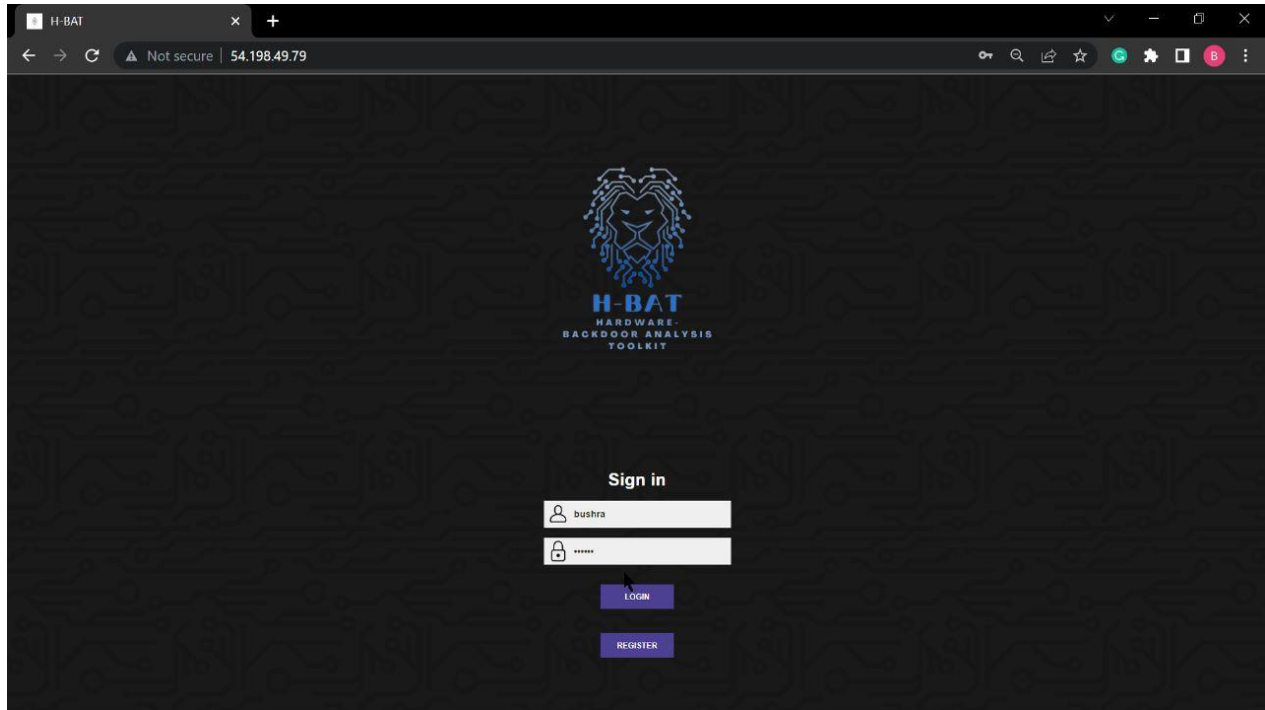
Laptop: Dell Alienware 14

Memory: 16 GB

System type: 64-bit operating system, x64-based processor

4.2 Toolkit Design

4.2.1 Interface



4.2.2 Architecture

The architecture of hardware backdoor analysis toolkit comprises of various modules as described below:

4.2.2.1 Extraction Module:

The extraction module of Hardware Backdoor Analysis Toolkit is a crucial component of the hardware backdoor analysis toolkit that focuses on extracting the firmware from a device to initiate the analysis process. It consists of several submodules, namely file system extraction, architecture detection, file detection, and root detection. The primary objective of this module is to facilitate the extraction and preparation of the firmware for subsequent analysis, decryption, and vulnerability assessment.

4.2.2.1.1. File System Extraction:

The file system extraction submodule is responsible for extracting the firmware from the device. It employs techniques and methods to identify and retrieve the firmware data stored within the device. This process involves accessing the device's memory, storage, or firmware update mechanisms to obtain the firmware image or relevant firmware files.

4.2.2.1.2. Architecture Detection:

The architecture detection submodule focuses on determining the hardware architecture of the device. It analyzes the extracted firmware to identify the underlying processor architecture, such as ARM, MIPS, x86, or others. This information is crucial for subsequent stages of analysis, as it helps in selecting the appropriate tools and techniques specific to the device's architecture.

4.2.2.1.3. File Detection:

The file detection submodule aims to identify and extract specific files within the firmware image that are relevant to the analysis. It employs file signature analysis, pattern matching, or known file structures to locate critical files such as executables, configuration files, libraries, or other firmware components. This step helps in isolating the essential components for further analysis.

4.2.2.1.4. Root Detection:

The root detection submodule focuses on identifying the root file system within the firmware. It aims to locate and extract the top-level directory structure, which contains system files, directories, configurations, and other essential components necessary for the device's functioning. Analyzing the root file system provides valuable insights into the device's software stack and potential areas of vulnerability.

The extraction module plays a pivotal role in the hardware backdoor analysis toolkit, as it sets the foundation for subsequent stages of analysis, including decryption, reverse engineering, vulnerability assessment, and mitigation. By effectively extracting and preparing the firmware, the toolkit enables researchers and security professionals to delve deeper into the firmware's inner workings, identify potential backdoors, and enhance the overall security of the device through vulnerability analysis and mitigation strategies.

4.2.2.2 Vulnerability Analyzer Module:

The vulnerability analyzer module is a crucial component of the hardware backdoor analysis toolkit, designed to identify and assess potential vulnerabilities within the firmware of a device. It consists of several submodules, including finding weak configuration files, version detection, user mode emulation, and kernel details analysis. The main objective of this module is to enhance the security of the device by uncovering and mitigating potential vulnerabilities.

4.2.2.2.1. Finding Weak Configuration Files:

The submodule for finding weak configuration files focuses on analyzing the extracted firmware to identify any configuration files that may contain weak settings or credentials. It scans through the configuration files, such as network configurations or administrative settings, to detect common security flaws like default passwords, exposed credentials, or insecure protocols. By identifying and addressing these weaknesses, the module helps strengthen the overall security posture of the device.

4.2.2.2.2. Version Detection:

The version detection submodule aims to determine the specific versions of software and libraries used within the firmware. It analyzes the extracted firmware to identify version information of critical components such as the operating system, third-party libraries, or firmware-specific modules. This information is crucial for vulnerability assessment, as it allows researchers to match known vulnerabilities associated with specific software versions and assess the device's exposure to such risks.

4.2.2.2.3. User Mode Emulation:

The user mode emulation submodule focuses on emulating the device's user environment within a controlled virtual environment. It enables the analysis of the firmware's behavior and interaction with various user-level processes. By simulating user interactions, the module can identify any potential vulnerabilities or malicious activities initiated through user interactions. This helps in detecting backdoors or suspicious behavior that may go unnoticed in a static analysis of the firmware.

4.2.2.2.4. Kernel Details Analysis:

The kernel details analysis submodule aims to extract and analyze the firmware's kernel components. It focuses on identifying the specific version, patch level, and configuration of the underlying kernel used in the device. This information is crucial for evaluating potential kernel-level vulnerabilities, including known vulnerabilities or misconfigurations that can be exploited by attackers. By assessing the kernel details, the module aids in identifying and mitigating potential risks associated with the device's kernel.

The vulnerability analyzer module plays a vital role in the hardware backdoor analysis toolkit by systematically evaluating the firmware for potential vulnerabilities. By analyzing weak configuration files, detecting software versions, performing user mode emulation, and assessing kernel details, the module enables security professionals to identify and mitigate potential backdoors and vulnerabilities, thus enhancing the overall security of the device.

4.2.2.3 System Emulation:

The system emulation module is a crucial component of the hardware analysis toolkit, designed to simulate the behavior of the device's firmware in a controlled environment. It consists of several submodules, including pre-emulation, network identification, final emulation, and a testing module comprising tools such as Nmap Port scanner, Binwalk, Firmadyne, and FirmAE. The primary aim of this module is to extract, decrypt, and analyze the firmware, perform reverse engineering, identify potential backdoors, and enhance the device's security through vulnerability analysis.

4.2.2.3.1. Pre-emulation:

The pre-emulation submodule prepares the environment for system emulation. It involves setting up the necessary virtualized environment, configuring the emulated system, and providing the required resources for firmware analysis. This submodule ensures that the emulation environment closely resembles the actual device to achieve accurate results during analysis.

4.2.2.3.2. Network Identification:

The network identification submodule focuses on identifying and analyzing the network-related aspects of the firmware. It involves examining network configurations, protocols, and communication channels used by the device. By identifying network vulnerabilities and potential backdoors, this submodule contributes to enhancing the security of the device's network connectivity.

4.2.2.3.3. Final Emulation:

The final emulation submodule executes the emulated firmware in the controlled environment. It simulates the device's behavior, including its interactions with the operating system, applications, and external components. By running the firmware in an emulated environment, this submodule enables thorough analysis, reverse engineering, and detection of potential backdoors or suspicious activities.

4.2.2.3.4. Testing Module:

The testing module is a crucial part of the system emulation module and comprises various tools used for analysis and vulnerability assessment. This includes:

- **Nmap Port scanner:** Nmap is a powerful port scanning tool that can be used to identify open ports and services running on the emulated device. It helps detect potential security vulnerabilities or misconfigurations associated with network services.

- **Binwalk:** Binwalk is a firmware analysis tool that enables the extraction of files and data from the firmware image. It helps in identifying embedded components, such as executable code, libraries, or configuration files, which may contain potential backdoors.

- **Firmadyne:** Firmadyne is a firmware emulation and analysis framework specifically designed for embedded devices. It enables the emulation of firmware to identify vulnerabilities and perform detailed analysis of the device's behavior.

- **FirmAE:** FirmAE is a firmware analysis environment that focuses on identifying and analyzing potential backdoors and vulnerabilities in embedded systems. It provides tools and techniques for comprehensive analysis, including dynamic and static analysis methods.

These tools within the testing module enable deep analysis, vulnerability identification, and backdoor detection, contributing to the overall goal of enhancing the device's security through effective firmware analysis and vulnerability mitigation.

The system emulation module plays a vital role in the hardware analysis toolkit by providing a controlled environment for firmware analysis, emulation of device behavior, and comprehensive testing. By utilizing tools like Nmap Port scanner, Binwalk, Firmadyne, and FirmAE, this module assists in the extraction, decryption, reverse engineering, and identification of potential backdoors and vulnerabilities, ultimately enhancing the security of the analyzed device.

4.2.2.4 Report Generator:

The report generator module is a crucial component of the hardware backdoor analysis toolkit, designed to generate comprehensive reports based on the results obtained from the firmware analysis and vulnerability assessment. This module consists of several submodules, including license collector, vulnerability aggregator, sum of exploits, vulnerability report, and web reports. The primary aim of this module is to compile all the findings and analysis into a structured and informative report, aiding in the identification and mitigation of potential vulnerabilities, and enhancing the overall security of the analyzed device.

4.2.2.4.1. License Collector:

The license collector submodule focuses on gathering information about the licenses and third-party components used in the firmware. It identifies the licenses associated with different

software or libraries embedded within the firmware. This submodule ensures compliance with license agreements and provides transparency regarding the use of third-party software.

4.2.2.4.2. Vulnerability Aggregator:

The vulnerability aggregator submodule collects and consolidates information about the vulnerabilities identified during the analysis. It compiles data from various sources, including vulnerability databases, security advisories, and the toolkit's own analysis results. This submodule ensures a comprehensive overview of potential security weaknesses in the device's firmware.

4.2.2.4.3. Sum of Exploits:

The sum of exploits submodule calculates the severity and impact of identified vulnerabilities by assessing the available exploits associated with each vulnerability. It determines the potential risk level and helps prioritize the vulnerabilities based on their criticality. This submodule aids in understanding the potential consequences and assists in making informed decisions regarding vulnerability mitigation.

4.2.2.4.4. Vulnerability Report:

The vulnerability report submodule generates a detailed report summarizing the vulnerabilities, their impact, and potential mitigation strategies. It includes information about the identified backdoors, weak configurations, insecure protocols, and other security weaknesses. This submodule provides a clear and concise overview of the vulnerabilities detected during the analysis.

4.2.2.4.5. Web Reports:

The web reports submodule focuses on generating user-friendly reports that can be accessed through a web-based interface. It provides interactive dashboards, graphical representations, and drill-down capabilities for easier navigation and exploration of the analysis results. This submodule enhances the usability and accessibility of the generated reports.

By incorporating these submodules, the report generator module ensures the comprehensive documentation of the analysis findings. The generated reports assist security professionals and stakeholders in understanding the vulnerabilities, making informed decisions, and implementing effective mitigation strategies. The report generator module plays a critical role in communicating the outcomes of the hardware backdoor analysis, promoting transparency, and ultimately enhancing the security of the analyzed device.

The following diagram describes the design of HBAT:

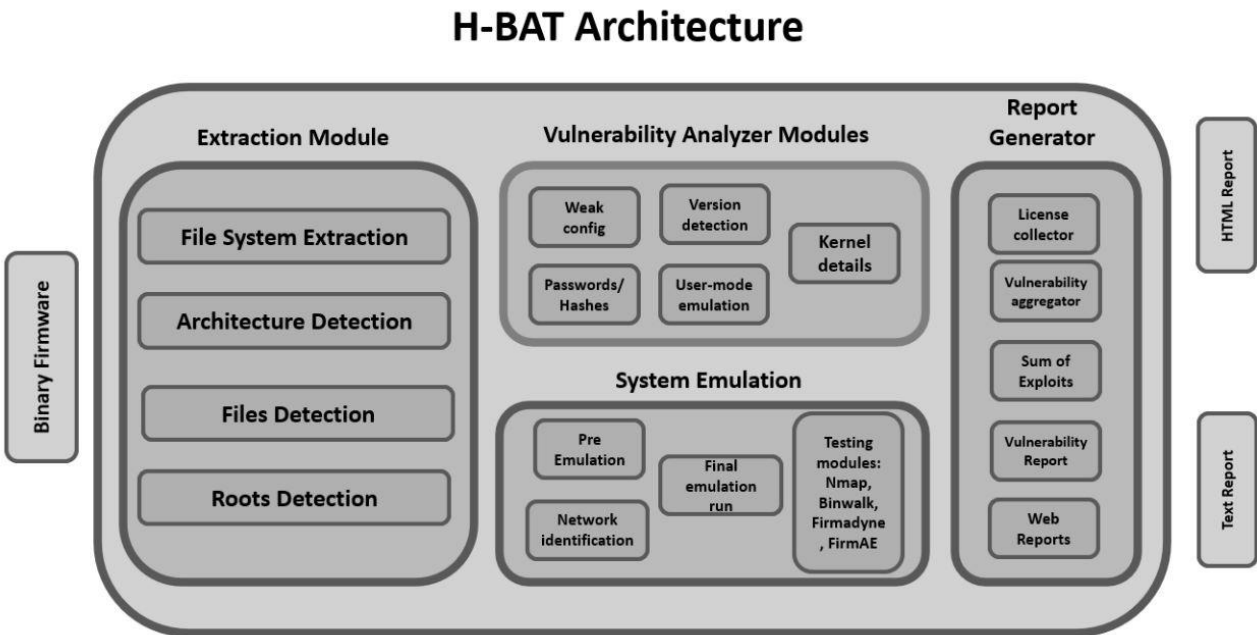


Figure 5: H-BAT Architecture

4.2.3 Supporting Systems

HBAT supports the following architectures:

1. MIPS
2. ARM

4.2.4 System tools

The installer installs all the necessary external tools for smooth operation but requires many system-internal tools.

Awk	basename	bc	cat	chmod	chown
Cp	cut	date	dirname	dpkg-deb	echo
Eval	find	grep	head	kill	ln
Ls	md5sum	mkdir	mknod	modinfo	mv
Netstat	printf	pwd	readelf	realpath	rm
Rmdir	rpm	sed	seq	sleep	sort
Strings	wc	touch	uniq	tree	unzip

4.2.5 Dependencies

H-BAT uses numerous external tools, which are installed with the installer script. The applications that must be installed on the OS for the toolkit to work properly are listed here:

- **OpenJDK:**
The Java class library, Java compiler, Java Runtime Environment (JRE), and Java virtual machine (JVM) are some of the important parts that make up the OpenJDK project.
- **GHIDRA:**
Ghidra is a framework for software reverse engineering (SRE).
- **OpenSSL:**
OpenSSL is a free command-line program that you can use to install your SSL/TLS certificate, make CSRs, generate private keys, and find certificate details.
- **Nmap Port Scanner:**
You can use the Nmap Port Scanner to test the effectiveness of your firewall and security settings.

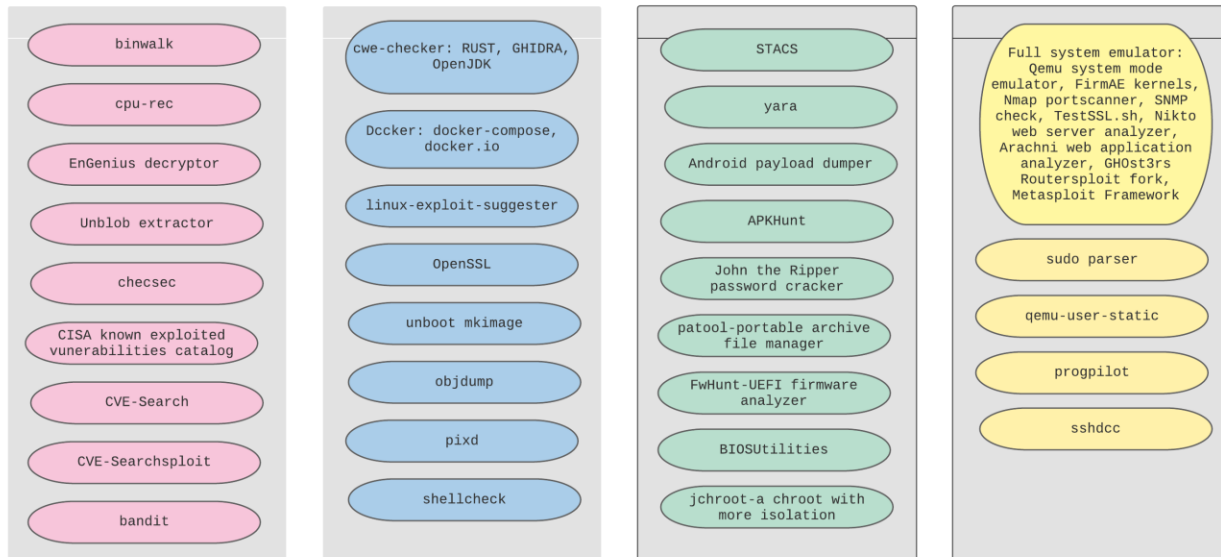


Figure 6: OS Applications

4.2.6 Generic

H-BAT is devised with the assumption that the firmware to be analyzed is either compressed or encrypted. Various pre-processing formulation is performed prior to the processes of extraction and unpacking. In addition, it is rather simple to improve the toolkit by adding more features to the existing framework. There are two steps involved in integrating each tool. The first step creates a docker image which is a lightweight, standalone, and executable package that includes everything needed to run an application, including the code, libraries, and system dependencies. The docker image encapsulates an environment in which an application can run, making it easy to deploy and manage across different systems. The second step is a Python automation script that executes the tool either on the original firmware image or the output generated.

4.2.7 Static and Dynamic Analysis

The analysis performed in the hardware backdoor analysis toolkit involves both static and dynamic analysis.

Static analysis is performed on the extracted firmware image to identify potential vulnerabilities and hidden backdoors. This involves examining the firmware code without executing it, looking for suspicious patterns, and analyzing the behavior of specific functions and modules.

Dynamic analysis is also performed to test the behavior of the firmware in a controlled environment. This involves running the firmware image on a virtual machine and monitoring its execution to identify potential vulnerabilities and backdoors.

Together, both static and dynamic analysis techniques are used to comprehensively evaluate the firmware image and identify any potential security weaknesses.

4.2.8 Cross Platform

The framework utilizes Docker to encapsulate all its tools, and the corresponding Docker images are available on Docker Hub for public access. This allows any user to download and use the tool.

4.3 Unpacking and Extraction Tools

4.3.1 Binwalk

Binwalk [3] is the tool used for processing raw firmware images. It helps to unpack the firmware and read its filesystems. It gives a great deal of information about the filesystems, the compression type, the file header, etc.

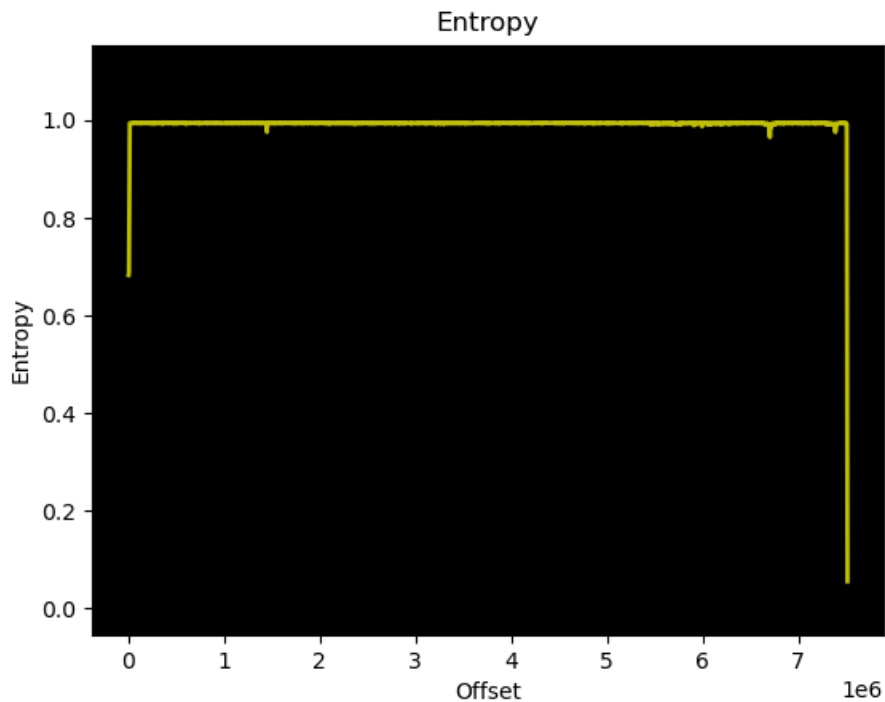
It can also help in entropy analysis, for example, if extraction fails, based on the entropy, it's possible to determine whether the image is encrypted, compressed, or has been archived using a custom proprietary format.

To extract firmware images, a combination of extraction and signature analysis is employed. Binwalk [3] offers the options of -eMr to perform this task, which can recursively extract the firmware along with the archived carved files, such as a filesystem, up to a maximum of 8 levels. Additionally, Binwalk's [3] primary functionality lies in its signature analysis, which is applied by default.

```
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Downloads$ binwalk -E DCS-935L_A1_FW_1.10.01_20161128_r4156.bin
```

DECIMAL	HEXADECIMAL	ENTROPY
0	0x0	Falling entropy edge (0.683039)
12288	0x3000	Rising entropy edge (0.994036)
7512064	0x72A000	Falling entropy edge (0.617941)

Figure 1

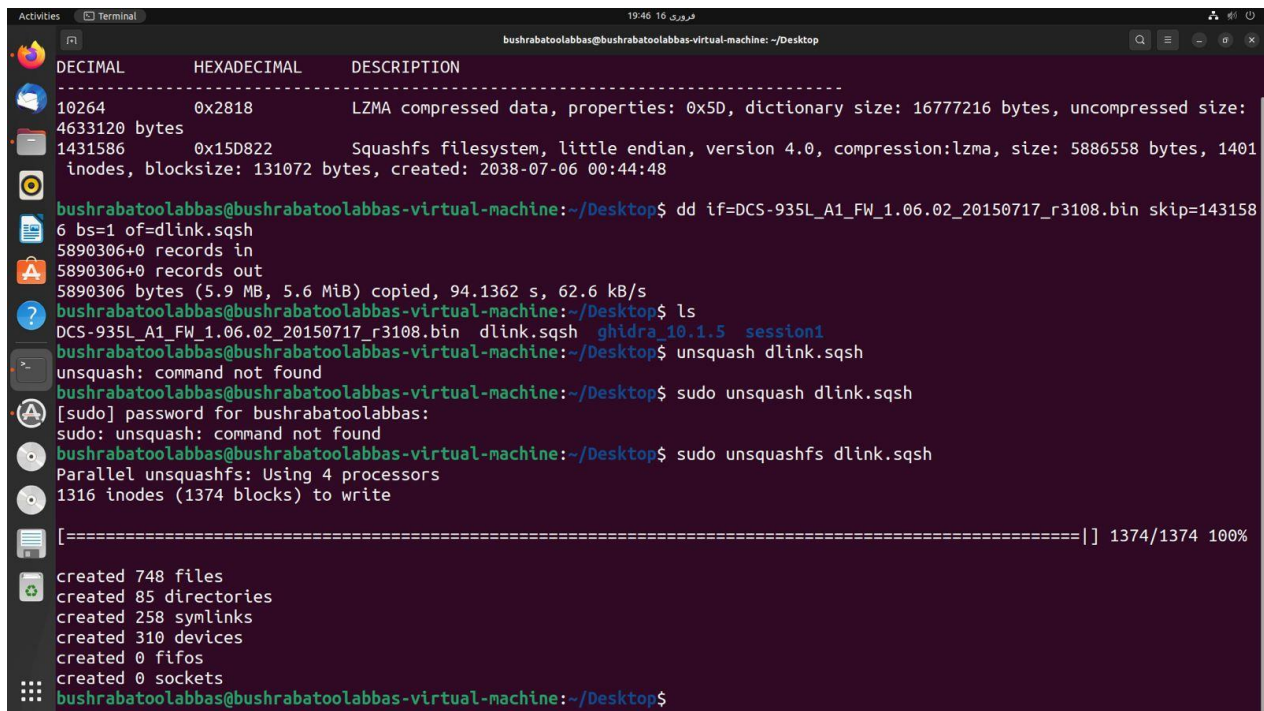


4.3.2 Gunzip

Files that have been compressed using the gzip command or the .gz file extension can be unzipped using the gunzip command.

```
gunzip [options] file.gz
```

4.3.3 Unsquash



The terminal screenshot shows the following commands and output:

```
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
10264        0x2818       LZMA compressed data, properties: 0x5D, dictionary size: 16777216 bytes, uncompressed size: 4633120 bytes
1431586      0x15D822    Squashfs filesystem, little endian, version 4.0, compression:lzma, size: 5886558 bytes, 1401 inodes, blocksize: 131072 bytes, created: 2038-07-06 00:44:48

bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$ dd if=DCS-935L_A1_FW_1.06.02_20150717_r3108.bin skip=143158
6 bs=1 of=dlink.sqsh
5890306+0 records in
5890306+0 records out
5890306 bytes (5.9 MB, 5.6 MiB) copied, 94.1362 s, 62.6 kB/s
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$ ls
DCS-935L_A1_FW_1.06.02_20150717_r3108.bin  dlink.sqsh  ghidra_10.1.5  session1
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$ unsquash dlink.sqsh
unsquash: command not found
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$ sudo unsquash dlink.sqsh
[sudo] password for bushrabatoolabbas:
sudo: unsquash: command not found
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$ sudo unsquashfs dlink.sqsh
Parallel unsquashfs: Using 4 processors
1316 inodes (1374 blocks) to write

[=====] 1374/1374 100%

created 748 files
created 85 directories
created 258 symlinks
created 310 devices
created 0 fifos
created 0 sockets
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$
```

4.3.4 Unlzma

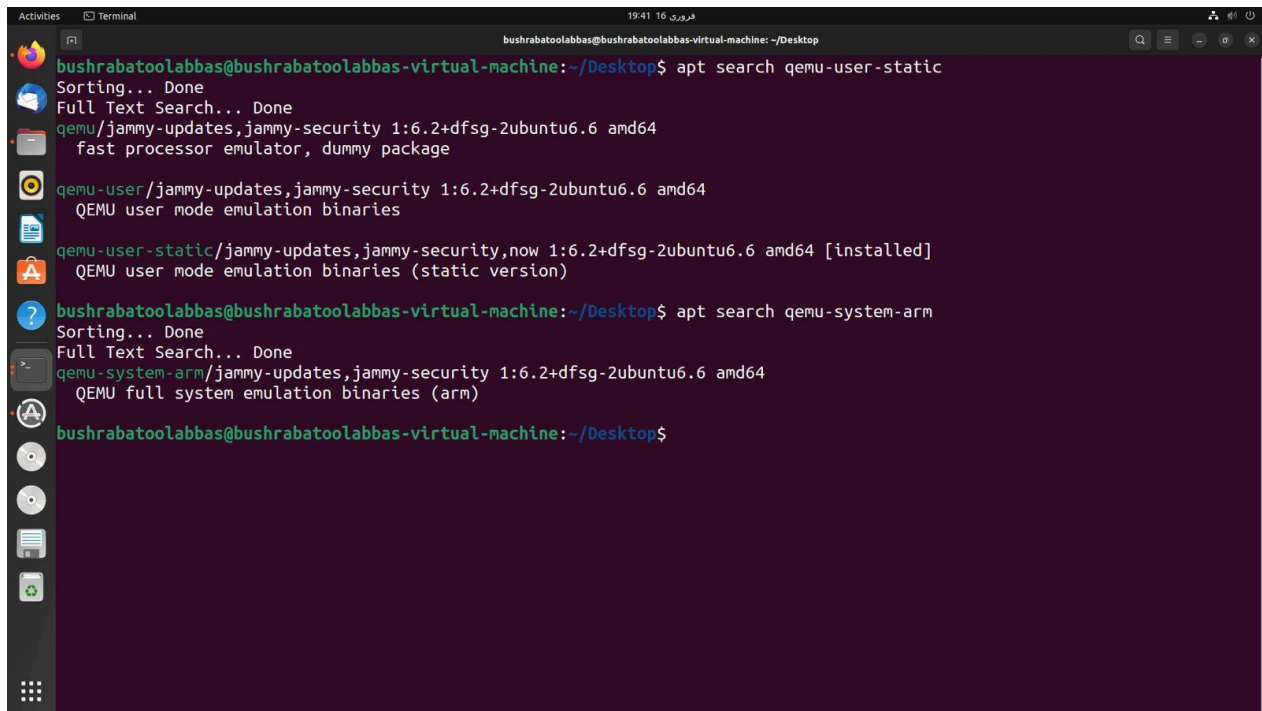
The unlzma command is used to decompress files that have been compressed with the high-compression LZMA algorithm.

```
unlzma [options] file.lzma
```

4.4 Emulation:

4.4.1 QEMU

For the filesystems that are not compatible with the host computer architecture, QEMU is used to emulate them. QEMU supports full system emulation using prebuilt images.

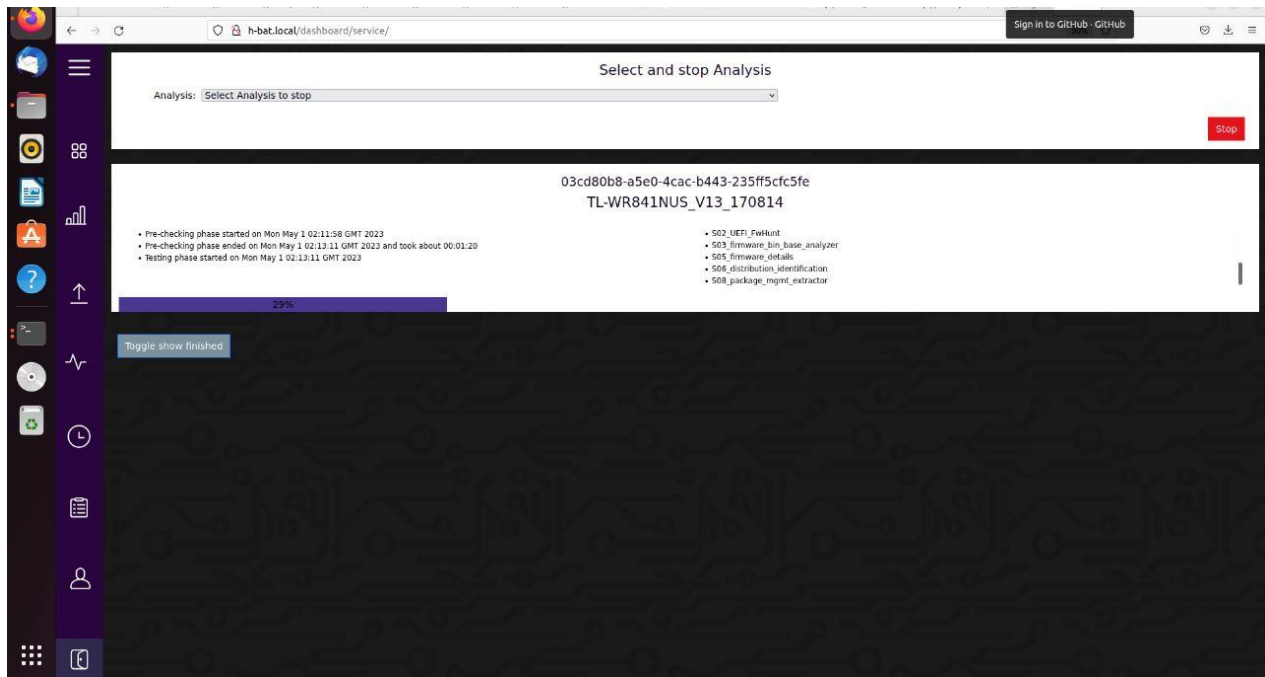


```
bushrabatoolabbas@bushrabatoolabbas-virtual-machine: ~/Desktop
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$ apt search qemu-user-static
Sorting... Done
Full Text Search... Done
qemu/jammy-updates,jammy-security 1:6.2+dfsg-2ubuntu6.6 amd64
fast processor emulator, dummy package
qemu-user/jammy-updates,jammy-security 1:6.2+dfsg-2ubuntu6.6 amd64
QEMU user mode emulation binaries
qemu-user-static/jammy-updates,jammy-security,now 1:6.2+dfsg-2ubuntu6.6 amd64 [installed]
QEMU user mode emulation binaries (static version)
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$ apt search qemu-system-arm
Sorting... Done
Full Text Search... Done
qemu-system-arm/jammy-updates,jammy-security 1:6.2+dfsg-2ubuntu6.6 amd64
QEMU full system emulation binaries (arm)
bushrabatoolabbas@bushrabatoolabbas-virtual-machine:~/Desktop$
```

4.5 Tp Link Firmware:

4.5.1 Analysis:

1. Examine all available files and use deep extraction mode to extract as much information as possible.
2. Analyze the firmware using binwalk and ent, select and extract the appropriate modules, and store the firmware structure in log files.
3. Search for files that may indicate the operating system and scan for markers of known distributions. Also, identify binaries that use weak functions, search for device tree blobs and startup files, and check for default run level.
4. Check for HTTP and webserver-related files with weak permissions, and look for password-related files to extract passwords and root accounts.
5. Scrape firmware for certification files and check their expiration date, printing a warning for outdated certificates.
6. Search for command injection by examining web-based files in folders such as www and looking for basic code execution patterns within them.



Analysis results:

```

.....
.....[+] Interesting function in /usr/sbin/msger (-rw-r--r-- root root) found:
.....
..... 24: 00401790 0 FUNC GLOBAL DEFAULT UND fprintf
..... 36: 00401710 0 FUNC GLOBAL DEFAULT UND printf
..... 38: 004016f0 0 FUNC GLOBAL DEFAULT UND strcpy
.....
.....[+] Interesting function in /usr/sbin/cfg (-rw-r--r-- root root) found:
.....
..... 17: 00400d10 0 FUNC GLOBAL DEFAULT UND sprintf
..... 22: 00400ce0 0 FUNC GLOBAL DEFAULT UND strcat
..... 23: 00400cd0 0 FUNC GLOBAL DEFAULT UND fprintf
..... 30: 00400ca0 0 FUNC GLOBAL DEFAULT UND printf
.....
.....[+] Interesting function in /usr/sbin/rtkmib (-rw-r--r-- root root) found:
.....
..... 22: 00400e30 0 FUNC GLOBAL DEFAULT UND fprintf
..... 32: 00400dd0 0 FUNC GLOBAL DEFAULT UND printf
.....
.....[+] Interesting function in /usr/sbin/ntpclient (-rw-r--r-- root root) found:
.....
..... 41: 00404160 0 FUNC GLOBAL DEFAULT UND fprintf
..... 68: 00404020 0 FUNC GLOBAL DEFAULT UND printf
.....
.....[+] Interesting function in /usr/sbin/ddns (-rw-r--r-- root root) found:
.....
..... 18: 00402480 0 FUNC GLOBAL DEFAULT UND sprintf
..... 52: 004022e0 0 FUNC GLOBAL DEFAULT UND strcpy
.....
.....[+] Interesting function in /usr/sbin/bundle_restore (-rw-r--r-- root root) found:
.....
..... 16: 00400cd0 0 FUNC GLOBAL DEFAULT UND sprintf
..... 21: 00400ca0 0 FUNC GLOBAL DEFAULT UND fprintf
..... 25: 00400c80 0 FUNC GLOBAL DEFAULT UND system
.....
.....[+] Interesting function in /usr/sbin/eventd (-rw-r--r-- root root) found:
.....
..... 70: 0041ce60 0 FUNC GLOBAL DEFAULT UND sprintf
..... 161: 0041cae0 0 FUNC GLOBAL DEFAULT UND strcat
..... 163: 0041cac0 0 FUNC GLOBAL DEFAULT UND fprintf
..... 206: 0041c940 0 FUNC GLOBAL DEFAULT UND system
..... 245: 0041c7d0 0 FUNC GLOBAL DEFAULT UND printf
..... 250: 0041c7a0 0 FUNC GLOBAL DEFAULT UND strcpy
.....
.....[+] Interesting function in /usr/sbin/openssl (-rw-r--r-- root root) found:
.....
..... 24: 004638d0 0 FUNC GLOBAL DEFAULT UND strcpy
..... 110: 00463450 0 FUNC GLOBAL DEFAULT UND printf
..... 387: 004624e0 0 FUNC GLOBAL DEFAULT UND fprintf
..... 988: 004603b0 0 FUNC GLOBAL DEFAULT UND strcat
.....
.....[+] Interesting function in /usr/sbin/hnap_tool_wlfi_sitesurvey (-rw-r--r-- root root)
.....
..... 45: 00402670 0 FUNC GLOBAL DEFAULT UND fprintf
..... 55: 00402600 0 FUNC GLOBAL DEFAULT UND system
.....
.....[+] Interesting function in /usr/sbin/mdb (-rw-r--r-- root root) found:
.....
..... 35: 0040acd0 0 FUNC GLOBAL DEFAULT UND sprintf
..... 74: 0040aab0 0 FUNC GLOBAL DEFAULT UND strcat
..... 75: 0040aaa0 0 FUNC GLOBAL DEFAULT UND fprintf
..... 98: 0040a950 0 FUNC GLOBAL DEFAULT UND system
..... 117: 0040a8a0 0 FUNC GLOBAL DEFAULT UND strcpy
.....
.....[+] Interesting function in /usr/sbin/mDNSClientPosix (-rw-r--r-- root root) found:
.....
..... 98: 0044c640 0 FUNC GLOBAL DEFAULT UND sprintf
..... 200: 0044c470 0 FUNC GLOBAL DEFAULT UND fprintf
..... 354: 0044c300 0 FUNC GLOBAL DEFAULT UND strcpy
.....
.....[+] Interesting function in /usr/sbin/rtsp/rtspd (-rw-r--r-- root root) found:
.....
..... 98: 0041b9f0 0 FUNC GLOBAL DEFAULT UND sprintf
..... 188: 0041b700 0 FUNC GLOBAL DEFAULT UND strcat
..... 191: 0041b6e0 0 FUNC GLOBAL DEFAULT UND fprintf
..... 292: 0041b370 0 FUNC GLOBAL DEFAULT UND printf

```

```

author: Andrey 'xatry' Konovalov

[+] [CVE-2013-0268] msr

Details: https://www.exploit-db.com/exploits/27297/
Exposure: less probable
Download URL: https://www.exploit-db.com/download/27297
Requirements: pkg=linux-kernel,ver>=2.6.18,ver<3.7.6
exploit-db: 27297

[+] [CVE-2012-0056,CVE-2010-3849,CVE-2010-3850] full-nelson

Details: http://vulnfactory.org/exploits/full-nelson.c
Exposure: less probable
Tags: ubuntu=(9.10|10.10){kernel:2.6.(31|35)-(14|19)-(server|generic)},ubuntu=
}
Download URL: http://vulnfactory.org/exploits/full-nelson.c
Requirements: pkg=linux-kernel,ver>=2.6.0,ver<=2.6.36
exploit-db: 15704

[+] [CVE-2010-4347] american-sign-language

Details: https://www.exploit-db.com/exploits/15774/
Exposure: less probable
Download URL: https://www.exploit-db.com/download/15774
Requirements: pkg=linux-kernel,ver>=2.6.0,ver<=2.6.36
exploit-db: 15774

[+] [CVE-2010-3904] rds

Details: http://www.securityfocus.com/archive/1/514379
Exposure: less probable
Tags: debian=6.0{kernel:2.6.(31|32|34|35)-(1|trunk)-amd64},ubuntu=10.10|9.10,f
686.PAE},ubuntu=10.04{kernel:2.6.32-(21|24)-generic}
Download URL: http://web.archive.org/web/20101020044048/http://www.vsecurity.c
t.c
Requirements: pkg=linux-kernel,ver>=2.6.30,ver<2.6.37
exploit-db: 15285

[+] [CVE-2010-3848,CVE-2010-3850,CVE-2010-4073] half_nelson

Details: https://www.exploit-db.com/exploits/17787/
Exposure: less probable
Tags: ubuntu=(10.04|9.10){kernel:2.6.(31|32)-(14|21)-server}
Download URL: https://www.exploit-db.com/download/17787
Requirements: pkg=linux-kernel,ver>=2.6.0,ver<=2.6.36
exploit-db: 17787

[+] [CVE-2010-3437] pktcdvd

Details: https://www.exploit-db.com/exploits/15150/
Exposure: less probable
Tags: ubuntu=10.04
Download URL: https://www.exploit-db.com/download/15150
Requirements: pkg=linux-kernel,ver>=2.6.0,ver<=2.6.36
exploit-db: 15150

[+] [CVE-2010-3301] ptrace_kmod2

```


4.5.2 Results:



Chapter 5: Conclusion

In this thesis, we discussed how reverse engineering [1] can lead to backdoors that can help identify the vulnerabilities that an IP camera device may be subjected to. These vulnerabilities act as a threat and can lead to providing an attack surface to the hacker. Our proposed toolkit runs a security analysis on the IP cameras and thus analyses their firmware to generate a vulnerability assessment report. It includes a selection of instruments and procedures for finding, examining, and eliminating backdoors from software systems. We have used already existing tools such as binwalk [3], firmadyne [17], and firmware analysis toolkit (FAT) [18]. Our backdoor analysis toolkit can assist organizations in strengthening their overall security posture and lowering the likelihood that an attacker will use a backdoor. It's crucial to remember that no toolkit or tool is impenetrable, and security must be viewed as an ongoing process that calls for constant oversight and development.

5.1 Limitations

- HBAT is limited to D-Link and TP-Link IP camera firmware series.
- It is time-consuming
- It cannot perform bulk analysis

Chapter 6: Future Work

1. Expansion of Capabilities:

This toolkit can be extended to examine the firmware for an extensive range of IoT devices. It could be performed by acquiring different methodologies for analyzing firmware for various architectures or by recognizing unfamiliar means for extracting and decrypting firmware.

2. Development of Advanced Vulnerability Analysis Techniques:

Supplementary backdoor analysis techniques could be developed to detect further subtle and sophisticated vulnerabilities. Various researchers and pen testers can study using different machine-learning algorithms to examine firmware performance and detect unfamiliar patterns which might suggest the presence of vulnerability within a device.

3. Integration of a Larger Backdoor and Vulnerability Database:

The toolkit can be developed to incorporate a wider database of previously found vulnerabilities that researchers and pen testers can utilize to evaluate against the firmware being targeted. This could aid in detecting vulnerabilities that had been found earlier and offer a quicker and more accurate analysis method.

4. Development of a User-Friendly Interface:

An intelligible interface for the toolkit could be developed to make it further available to people who are not experts in this field. This would make it more accessible to the users increasing its scalability by eventually enhancing the protection of various IoT devices.

5. Collaboration with Device Manufacturers:

Further work might involve association with the device's manufacturers to feature the toolkit in their security assessing, authentication, and authorization stages. This will help in detecting and alleviating probable backdoors in IoT devices prior to their release in the target market.

Conclusively, the Hardware Backdoor Analysis Toolkit has the substantial capability for imminent advancements and expansion. Through persistent refinement and expansion of the capabilities of the toolkit, researchers and pen testers can stay on top of growing threats and secure their devices from probable backdoors.

Bibliography

1. Reverse Engineering [online] Available at:
<https://en.wikipedia.org/wiki/Reverse_engineering>
2. Report by Kaspersky [online] Available at: <<https://www.pymnts.com/news/security-and-risk/2021/kaspersky-detects-iot-cyberattacks-double-last-year/amp/>>
3. Binwalk [online] Available at: <<https://www.kali.org/tools/binwalk/>>
4. Entropy Analysis online Available at:
<<https://reverseengineering.stackexchange.com/questions/21555/what-is-an-entropy-graph>>
5. Gzip Compression [online] Available at: <<https://www.javatpoint.com/linux-gzip>>
6. Squashfs Filesystem online Available at:
<<https://www.kernel.org/doc/html/next/filesystems/squashfs.html>>
7. Cramfs Filesystem online Available at:
<<https://www.kernel.org/doc/html/latest/filesystems/cramfs.html>>
- JFFS2 Filesystem online Available at: <https://en.wikipedia.org/wiki/JFFS2>
9. Virtual Machine online Available at:
<<https://www.vmware.com/topics/glossary/content/virtual-machine.html>>
10. Firmwalker online Available at: <<https://github.com/craigz28/firmwalker>>
11. CVE online Available at: <<https://www.balbix.com/insights/what-is-a-cve/>>

12. Emulation online Available at: <<https://www.fortinet.com/blog/threat-research/Using-emulation-against-anti-reverse-engineering-technique>>
13. JTAG online Available at: <<https://www.alexforencich.com/wiki/en/reverse-engineering>>
14. AttifyOS online Available at: < <https://www.attify.com/attifyos> >
15. UART online Available at: <<http://blog.k3170makan.com/2019/06/hardware-reverse-engineering-uart.html> >
16. EMMC online Available at: < <https://www.riverloopsecurity.com/blog/2020/03/hw-101-emmc>>
17. Firmadyne online Available at: < <https://github.com/firmadyne/firmadyne> >
18. Firmware Analysis Toolkit FAT online Available at: < <https://github.com/attify/firmware-analysis-toolkit> >
19. QEMU online Available at: <<https://ariadne.space/2021/05/05/using-qemu-user-emulation-to-reverse-engineer-binaries> >
20. Breaking all the things: a systematic survey of firmware extraction and modification techniques for IoT devices by Sebastian Vasile, David Oswald, and Tom Chothia online Available at: <<https://research.birmingham.ac.uk/en/publications/breaking-all-the-things-a-systematic-survey-of-firmware-extraction>>
21. Methods for extracting firmware from OT devices for vulnerability research by Nozomi Networks Labs, 2022 online Available at: <https://www.nozominetworks.com/blog/methods-for-extracting-firmware-from-ot-devices-for-vulnerability-research>