

An Innovative Honeypot Architecture for Detecting and Mitigating Hardware Trojans in IoT Devices

Amira Hossam Eldin Omar ¹, Hassan Soubra ², Donatien Koulla Moulla ^{3,*} and Alain Abran ⁴

¹ Department of Computer Science and Engineering, German University in Cairo, New Cairo 4721301, Egypt; amira.h.roshdy@gmail.com

² Department of Computer Engineering, Ecole Centrale d'Electronique-ECE Lyon, 24 rue Salomon Reinach, 69007 Lyon, France; hsoubra@ece.fr

³ Centre for Augmented Intelligence and Data Science, School of Computing, University of South Africa, Johannesburg 1709, South Africa

⁴ Department of Software Engineering and Information Technology, Ecole de Technologie Supérieure, Montréal, QC H3C 1K3, Canada; alain.abran@etsmtl.ca

* Correspondence: moulldk@unisa.ac.za

Abstract: The exponential growth and widespread adoption of Internet of Things (IoT) devices have introduced many vulnerabilities. Attackers frequently exploit these flaws, necessitating advanced technological approaches to protect against emerging cyber threats. This paper introduces a novel approach utilizing hardware honeypots as an additional defensive layer against hardware vulnerabilities, particularly hardware Trojans (HTs). HTs pose significant risks to the security of modern integrated circuits (ICs), potentially causing operational failures, denial of service, or data leakage through intentional modifications. The proposed system was implemented on a Raspberry Pi and tested on an emulated HT circuit using a Field-Programmable Gate Array (FPGA). This approach leverages hardware honeypots to detect and mitigate HTs in the IoT devices. The results demonstrate that the system effectively detects and mitigates HTs without imposing additional complexity on the IoT devices. The Trojan-agnostic solution offers full customization to meet specific security needs, providing a flexible and robust layer of security. These findings provide valuable insights into enhancing the security of IoT devices against hardware-based cyber threats, thereby contributing to the overall resilience of IoT networks. This innovative approach offers a promising solution to address the growing security challenges in IoT environments.

Keywords: IoT; honeypots; hardware Trojan; VHDL; FPGA; Raspberry Pi; Python; socket programming



Citation: Omar, A.H.E.; Soubra, H.; Moulla, D.K.; Abran, A. An Innovative Honeypot Architecture for Detecting and Mitigating Hardware Trojans in IoT Devices. *IoT* **2024**, *5*, 730–755. <https://doi.org/10.3390/iot5040033>

Academic Editor: Amiya Nayak

Received: 16 July 2024

Revised: 16 October 2024

Accepted: 29 October 2024

Published: 31 October 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The number of Internet of Things (IoT) devices has grown exponentially and is expected to reach 80 billion by 2025 [1]. Although IoT device manufacturers have mainly focused on functional aspects, cybersecurity threats have become a major concern. The proliferation of insecure IoT devices has introduced significant vulnerabilities that can be exploited in large-scale cyberattacks. Ensuring the security of these devices is becoming a critical priority for preventing major incidents, such as the notorious Mirai botnet attack in 2016 [2].

Traditional security measures, such as Intrusion Detection Systems (IDSs) [3] and Intrusion Prevention Systems (IPS) [4], offer limited protection for IoT environments. Honeypots [5], decoy systems designed to mimic real systems, applications, or services, have emerged as a promising method for studying and mitigating these threats. These systems enable security experts to observe attacker behavior and techniques in a controlled environment without compromising real assets. However, the integration of honeypots into IoT systems requires further development, particularly for addressing hardware-level threats such as hardware Trojans (HTs) [6].

Although significant research has been conducted to secure traditional computing systems, limited attention has been paid to securing IoT devices [7]. Addressing the security of IoT devices is critical [8]. Additionally, there has been significant growth in research aimed at securing connected cyber-physical systems such as smart autonomous bikes [9,10].

HTs are malicious modifications to integrated circuits (ICs) that can cause performance degradation, functionality alterations, and data leakage. These modifications are often difficult to detect because they are triggered under specific conditions, making traditional software-based defenses inadequate. To address this issue, this study proposes a novel approach for detecting and mitigating HTs in IoT devices by leveraging hardware-based honeypots.

The proposed system is implemented on a Raspberry Pi and tested using a Field-Programmable Gate Array (FPGA) to emulate HTs. By integrating honeypots into the IoT hardware layer, the system effectively detects HTs without adding complexity to the IoT device operations. This solution is fully customizable and provides a flexible defense mechanism to meet the varying security requirements in IoT environments.

The remainder of this paper is organized as follows: Section 2 presents the background and related works. Section 3 presents the proposed IoT hardware Trojan honeypot (the proposed approach). Sections 4 and 5 describe the experimental setup (the experimental setup) and testing of the proposed approach, respectively (results and discussion). Finally, Section 6 concludes the study and outlines the potential for future research.

2. Literature Review

This literature review provides a comprehensive analysis of the current state of research related to IoT security, focusing particularly on the detection and mitigation of HTs using honeypots. This section explores the various challenges faced by IoT devices, nature and impact of HTs, existing detection mechanisms, and role of honeypots in enhancing security. Additionally, the review identifies gaps in current research, underscores the need for innovative approaches, and sets the stage for the proposed approach.

2.1. IoT Security Challenges

2.1.1. Overview of IoT Security Landscape

IoT security faces various challenges, including insufficiently upgraded devices, weak security measures, operator negligence, and unreliable device monitoring [11]. The integration of IoT devices into traditional networks introduces additional security complexities, necessitating the exploration of new solutions [12]. Furthermore, the rapid expansion of IoT activities has surpassed the ability of operators and asset owners to respond effectively, emphasizing the critical need for a focused approach to business risk [13]. The unique architecture of IoT networks creates new vulnerabilities and cyber threats, demanding a deep understanding of these challenges and the implementation of robust mitigation strategies [14]. Advances in securing IoT devices have been made by leveraging technologies such as blockchain and machine learning algorithms, showing promising results in enhancing data protection and overall security.

2.1.2. Common Vulnerabilities in IoT Devices

Common security vulnerabilities in IoT devices include weak communication protocols, inadequate authentication mechanisms, and unauthorized access facilitated by inherent design flaws and poor implementation of security standards [15,16]. These vulnerabilities can lead to cyberattacks, data breaches, and privacy concerns [17]. To address these issues, mitigation strategies involve the implementation of secure communication protocols, robust authentication mechanisms, and regular software updates [18,19]. In addition, raising user awareness of security risks, promoting responsible data collection practices, and incorporating privacy-by-design principles can help create a safer IoT environment. Intrusion Detection Systems (IDSs) based on new technologies such as blockchain and machine learning are also promising solutions for enhancing cybersecurity in IoT ecosystems.

2.1.3. Attacks on IoT Devices

Various attacks on IoT devices include Distributed Denial of Service (DDoS) [20], Denial of Service (DoS), reconnaissance, brute force, man-in-the-middle, and spoofing attacks [21]. These attacks pose significant security risks owing to the vulnerabilities present in IoT devices, which lack industry-wide security standards, making them susceptible to exploitation by cybercriminals [21]. In a study analyzing IoT device vulnerabilities [22], attacks such as DoS, man-in-the-middle, and brute force attacks were successfully executed on a Raspberry Pi 4 acting as an IoT device, highlighting the importance of identifying and mitigating these risks [22]. Additionally, a comprehensive IoT attack dataset was created, encompassing 33 attacks across seven categories, including DDoS, DoS, reconnaissance, brute force, spoofing, and Mirai attacks, to facilitate the development of security analytics applications for real IoT operations [23].

2.2. Hardware Trojans (HTs)

Definition and Classification of HTs

HT is a malicious, unwanted, and deliberate modification of an electronic circuit. However, these modifications can have several consequences:

- **Alteration of functionality:** This causes the circuit to perform unauthorized operations, such as encryption algorithm bypassing, privilege escalation, and denial of service.
- **Degraded performance:** This can potentially jeopardize the critical system through induced electromigration of wires caused by continuous DC stress, increased or decreased path delay, and fault injection.
- **Information leakage:** This undermines the security provided by cryptographic algorithms or directly leaks sensitive data handled by an integrated circuit (IC). This could include the disclosure of cryptographic keys or other sensitive information via debugging or I/O ports and side channels (e.g., delay, power).

An HT consists of two parts: trigger and payload. The HT is inserted into the original circuit. The user is usually unaware of its existence because the circuit behaves normally most of the time but starts behaving maliciously when the HT is triggered. Consequently, it is not sufficient to protect only the software layer of the device; it is also critical to protect the hardware layer of the device [24].

Hardware Trojans are classified into three classes [25]:

- **Combinational:** A trigger is taken from a circuit's primary inputs and/or internal nodes, and a payload can be activated once the trigger is asserted. Any Trojan design can be classified as a Type-p Trojan based on p-trigger inputs. An AND gate with p-inputs can be used to create the most basic trigger form. Any other combinational logic can also be used as a trigger to produce a different logic when activated.
- **Sequential:** The payload is delivered when a sequence of input patterns occurs or a period is triggered. To accomplish this goal, the trigger design of a sequential Trojan incorporates the state elements and combinational logic. The payload is delivered only when the counter reaches its maximum count or when the Finite State Machine (FSM) for the counter reaches its final state in the first approach. Because specific test patterns or inputs are unlikely to occur consecutively multiple times during testing or normal operations on IC, this property of sequential Trojans makes detection even more difficult.
- **Analog (RF Trojan):** The trigger circuit is designed with capacitors that are activated by accumulating charge from the toggling of a nearby victim wire that exceeds a specified threshold.

The design types of HT differ according to their triggering mechanisms. For instance, a counter-based Trojan, also known as a 'ticking time bomb', waits until the user has reached a preset number of executions of an instruction [26]. A cheat code Trojan is activated when a certain user input is entered into a circuit. Backdoor Trojans are malicious software

programs that allow unauthorized access to remote attacks [27]. Remote attackers can execute commands or achieve complete control of a compromised computer. Backdoor malware and viruses circumvent authentication procedures to gain access to systems and to avoid detection.

2.3. Detection Mechanisms for Hardware Trojans

The detection mechanisms for HT include side-channel information analysis [28], VGG-Net architecture-based detection [29], trigger signal testability assessment [30], and IC topology and behavior-aware detection using Graph Neural Networks (GNNs) [31].

Side-channel information analysis leverages dual discriminator-assisted conditional generation adversarial networks (D2ACGAN) to distinguish between side-channel data with and without Trojans, achieving high accuracy rates. The VGG-Net architecture-based method excels in detecting Trojans in Advanced Encryption Standard benchmarks. Trigger signal testability assessment focuses on the low testability of trigger signals within circuits to enhance detection speed and reduce false positives.

Additionally, the IC topology and behavior-aware approach utilize structural features and behavioral information extracted through graph learning to effectively detect Trojans without requiring a golden IC reference design, demonstrating high accuracy rates even on unseen Trojan benchmarks [32].

2.4. Honey pots

A honeypot is a cybersecurity mechanism that employs a simulated attack target to divert the attention of cybercriminals from legitimate targets [33]. They also gather intelligence on the identities, methods, and motivations of the adversaries. A honeypot can be designed to look like any digital asset, such as software applications, servers, or the network itself. It is purposefully designed to look like a legitimate target, with a structure, components, and content similar to that of the model. This is intended to persuade the adversary that they have gained access to the actual system and to encourage them to spend time within this restricted environment.

The researchers in [33] introduced a hardware honeypot called the Finite State Machine Honey pot (FSM-HP). The purpose was to imitate the original FSM by accepting identical inputs, aiming to appear highly realistic and evade detection. However, the FSM-HP was intentionally equipped with additional vulnerabilities to enhance its appeal to potential attackers. A honeypot can be reprogrammed if novel identification mechanisms are created. Accompanied by new characteristics, the generation of honeypots can be adjusted accordingly [33]. This makes a carefully constructed and designed honeypot an ideal mitigation mechanism for protection against adversaries.

Honey pots are classified into three major groups, each of which is further subdivided as follows [34]:

1. Purpose-Based:

- **Research Honey pot:** A research honeypot is designed to gather information about the specific methods and techniques used by adversaries as well as to identify potential vulnerabilities within the system related to these tactics. Typically, research honeypots are more complex than production honeypots are.
- **Production Honey pot:** The common type of honeypot is the production honeypot. Businesses use this decoy to gather information and intelligence regarding cyberattacks on production networks. The collected data include IP addresses, intrusion attempt times and dates, traffic volume, and other attributes. Production honeypots are relatively simple to design and deploy; however, they produce less sophisticated intelligence than research honeypots.

2. Levels of Interactivity-Based:

- **Low-Interaction Honey pot:** Low-interaction Honey pots use a small number of resources to simulate parts of a system's software or network services while gath-

ering basic information about the attacker. Because of their limited capabilities, attackers cannot escape emulation; thus, the host system cannot be compromised. The vast majority of production honeypots have low interaction. HoneyD is one of the most popular low-interaction honeypots.

- **High-Interaction Honeypot:** A high-interaction honeypot represents the other end of the spectrum in deception technology. Without a simulation, high-interaction honeypots use a comprehensive operating system. They are difficult to maintain and sophisticated and are designed to keep hackers occupied for long periods by utilizing a network of exploratory targets, such as several databases. This provides the cybersecurity team with a better understanding of how these attackers operate, their tactics, and even hints as to who they are [34,35].

3. Implementation Based:

- **Physical Honeypot:** The term “physical honeypot” or “hardware honeypot” refers to a honeypot powered by a physical mechanism. It runs on a real machine connected to the network by using its assigned IP address. Physical Honeypot generally suggests a high level of engagement, which allows the system to be entirely compromised.
- **Virtual Honeypot:** Virtual Honeypots can combine different levels of interactivity as they can host more than one honeypot and control the number and characteristics of the ones deployed. This can be more cost-efficient and less time-consuming [36].

Various honeypot systems, such as HoneyIoT [37], IoTZeroJar [38], and RIoTpot [39], have been developed to address security challenges in IoT environments. These systems utilize adaptive high-interaction techniques to analyze zero-day attacks and expose themselves to threats on the Internet. This approach allows for the collection of valuable data on attack patterns and attacker behavior [40]. By mimicking real IoT devices and applications, honeypots can effectively deceive attackers, gather information on emerging threats, and prevent unauthorized access to actual devices. This significantly enhances the overall security posture of IoT networks [41].

2.5. Analysis and Gaps

IoT devices face significant challenges, including insufficient updates, weak security, and complex integration into traditional networks, which necessitate innovative solutions [11–14]. Numerous studies have shared common drawbacks such as insecure communication, poor authentication, and design flaws, all of which lead to cyberattacks and data breaches. Proposed solutions include secure protocols, robust authentication, and user awareness [15–19]. IoT devices are susceptible to DDoS, DoS, brute force, and man-in-the-middle attacks. Studies highlight the importance of identifying and mitigating these risks [20–23].

Many IoT devices remain vulnerable owing to their inherent design flaws and lack of industry-wide security standards. Although promising, existing detection mechanisms for HTs often require complex implementations and may not cover all possible Trojan designs. Although effective, honeypot systems can be resource-intensive and require careful design to avoid detection by attackers. Research on IoT security requires continuous updating to keep pace with the rapidly evolving cyber threats and technologies.

In summary, researchers have proposed several methods and techniques to address the security issues in IoT devices. However, to the best of our knowledge, only a few studies have focused on the detection and mitigation of HT using honeypots. Table A1 in the Appendix A presents some recent literature on IoT security, with a summary of their strengths and weaknesses.

Given the extensive work presented in Table A1, it is evident that the existing solutions for HT detection in IoT devices often require additional complexity and significant modifications to the IoT system. IoT devices are typically built on less powerful chips, which make them more vulnerable than other devices within a network. To address these

vulnerabilities, we propose a novel approach that utilizes a hardware network-based honeypot that mimics IoT devices. This honeypot monitors and blocks any suspicious activity and effectively identifies an adversary’s intentions. This is achieved without introducing overhead to the IoT system or requiring knowledge of its internal working. Our approach is Trojan-agnostic and fully customizable to meet specific security needs, thereby providing a flexible and robust solution. Crucially, the logs generated by the honeypot can be analyzed to enhance network security, offering a practical means to address vulnerabilities and strengthen overall defenses.

3. Proposed Approach

Our approach involved designing and deploying a honeypot, creating an HT, and testing the system. The system was rigorously tested to ensure its effectiveness in identifying and analyzing security threats.

3.1. Honeypot Design Architecture

Our honeypot was designed to mimic vulnerable IoT devices, thereby attracting and recording cyberattacks. This enables the analysis of attack patterns and methodologies, helping to understand the strategies used by attackers to compromise IoT devices. The honeypot architecture is multilayered and comprises the following components:

- **Emulation Layer:** Simulates various IoT device vulnerabilities.
- **Monitoring Layer:** Tracks interactions with the honeypot.
- **Logging Layer:** Records data about the attacks for further analysis.

Figure 1 illustrates the honeypot system architecture. The emulation layer includes modules that mimic various IoT devices by hosting services identical to those offered by the actual IoT system. The monitoring layer captures the network traffic directed at these emulated devices. This layer uses socket programming to communicate with IoT devices in the LAN, determining whether incoming packets will be forwarded to the IoT device or dropped. The logging layer stores the captured data for analysis. The honeypot is strategically deployed within a controlled network environment to ensure that it appears to be a legitimate device, thus enabling attackers to interact with it. It monitors incoming traffic, captures malicious activities, and logs the details for further analysis. The honeypot uses Python 3.12 socket programming for network communication and integrates a packet sniffer to detect and log malicious packets.

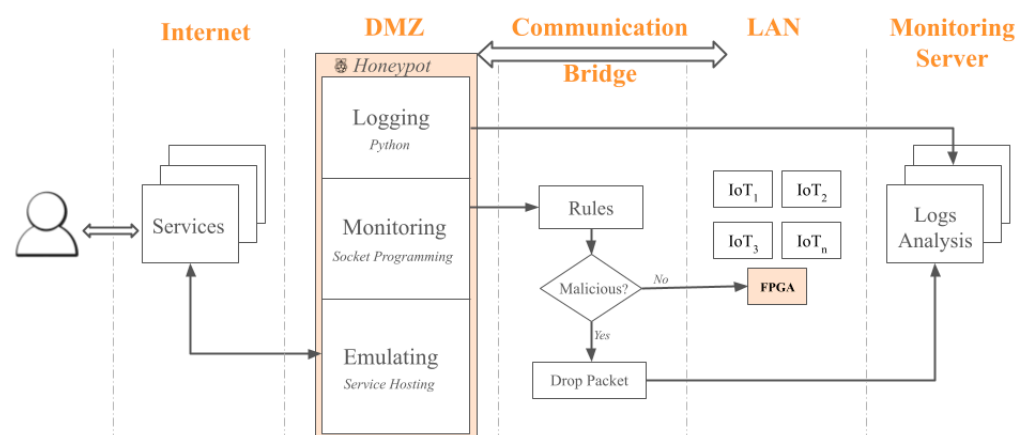


Figure 1. The Honeypot System architecture.

3.2. HT Design

Trigger-based HT is a type of malicious modification or addition to hardware components, particularly integrated circuits (ICs), and is designed to be activated under specific

conditions or triggers. These triggers can be based on various factors, such as a specific time, sequence of operations, signal pattern, or specific input sequence.

The mechanisms for activating trigger-based Trojans are divided into two categories:

- **Internal triggers:** These Trojans are conditional, such as a ticking time bomb, using counters to compute the execution of an instruction, or physical conditions, such as tampering with the internal temperature of the components.
- **External triggers:** These Trojans can be activated remotely by an attacker based on external factors, such as the user entering a specified input (cheat code) or a component returning a specific output.

This study employs a combination of a ticking time bomb and a backdoor Trojan. The counter-Trojan is activated if a specific command exceeds a predetermined value, thereby triggering the payload. This in turn activates the backdoor Trojan, creating a backdoor in the circuit and affecting its functionality.

4. Experimental Setup

4.1. Honeygot

The proposed honeypot is a low-interactive system implemented on a Raspberry Pi Model 3 B+, operating with the desktop version of the Raspbian 64-bit Lite OS. It redirects attackers to a fake admin login web page, prompting them to enter the username and password. The honeypot records all data, including the IP address, type of request, values entered, and breached port. It was created using a socket programming approach with a Python script directing users to a web page by listening to port 80, the port where the server anticipates receiving data from a web client. Multiple ports can be opened, allowing the honeypot to monitor changes or requests and respond accordingly. Socket programming facilitates communication between two nodes in a network: one socket (node) listens to a specific port at an IP address, whereas the other socket establishes a connection. When the client attempts to contact the server, the server creates a listener socket. The following Script demonstrates how the socket was created and configured for IPv4 addressing using a TCP connection. The socket is given a dedicated port to which it should listen, as illustrated in Script 1.

Script 1: Socket Connection Establishment

```
try:
    get_socket_con = socket(AF_INET, SOCK_STREAM)
    get_socket_con.bind((ip_add, port))
    get_socket_con.listen(10) #timeout after 10 s
while True:
    client_con, client_addr = get_socket_con.accept()
    print("Visitor Found! - {}".format(client_addr[0]))
```

The code snippet provided in Script 1 establishes a socket connection to create a honeypot system that listens to the incoming connections from potential attackers. The try block initiates the process by creating a new socket using the AF_INET address family and SOCK_STREAM socket type, which is suitable for TCP connections. The bind method associates the socket with a specific IP address (ip_add) and port (port). The listen method then sets the socket to listen to incoming connections with a timeout set to 10 s. Within an infinite loop (while True), the accept method waits for a client to establish a connection, returning a new socket object (client_con) and the address of the connected client (client_addr). Upon successful connection, the code prints a message indicating the detection of a visitor along with the visitor's IP address.

The session must then be ended upon completion, closing the endpoints.

The Hypertext Transfer Protocol (HTTP) is handled by port 80. Therefore, a request can be submitted by typing the honeypot's IP address '192.168.1.4' in any web browser. The attacker is then routed to a webpage that is designed using JavaScript and CSS. As

illustrated in Figure 2, two input text boxes represent the data needed to unlock the SmartLock system. This is normal behavior for the IoT system, giving the attackers a false sense of security and believing that they have infiltrated our IoT system. These inputs, along with other information about the attack and its location, were saved for the analysis.

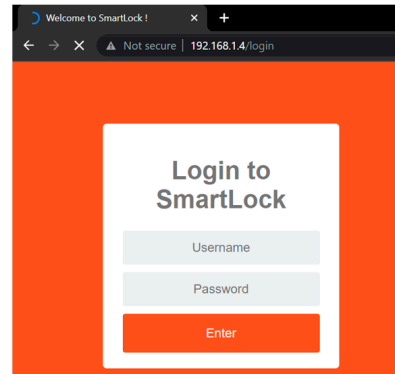


Figure 2. SmartLock Fake Webpage.

String manipulation techniques were applied to extract the username and password values. Data were retrieved from the HTML page using a socket programming approach that captures and decodes the response from the webpage. String manipulation was then performed on the decoded response to clean and filter data. These logs, generated by the honeypot, record any active connections established with the honeypot. The terminal output first announces that the honeypot is active and logs any device that connects to it within a specified time frame. Subsequently, it logs the values entered on the website, allowing us to track the attacker's activity on the webpage. The terminal output after entering the data into the webpage is shown in Figure 3.

```
"C:\Users\Amira_Housam\PycharmProjects\pythonProject\venv\Scripts\python.exe" "C:/Users/Amira_Housam/PycharmProjects/pythonProject/main.py"
[+] Honeypot Starting.....
Visitor Found! - [192.168.1.5]
Visitor Found! - [192.168.1.5]
Visitor Found! - [192.168.1.5]
Value a: 200
Value b: 150
```

Figure 3. Output of the terminal.

Next, an Excel spreadsheet was generated using a Python script to format the logged data. This sheet records the time a connection is made to the honeypot, the IP address of the device that made the request, the port accessed, and the type of connection, whether it is a GET request (when accessing the webpage) or a POST request (when sending a response to the webpage). In addition, it records the operating system and other information about the device that made the connection, as well as the actual values of the usernames and passwords that were entered.

Figure 4 illustrates the Excel spreadsheet "Honeypot Logs" automatically created locally, as previously mentioned. The sheet contains six columns with the following information:

- **Time Stamp:** The exact time a connection was made.
- **Visitor IP:** The IP address of the suspicious device.
- **Listening on Port:** The vulnerable port accessed on the honeypot.
- **Visitor Information:** Details about the type of connection made.
- **Username:** The username entered on the website.
- **Password:** The password entered on the website.

A	B	C	D	E	F
Time Stamp	Visitor IP	Listening on Port	Visitor Information	UserName	Password
2022-05-28 23:33:02	IP: - [192.168.1.5]	Port: - [80]	POST /login HTTP/1.1Host: 192.168.1.5Connectio 6	16	1
2022-05-28 23:54:20	IP: - [192.168.1.5]	Port: - [80]	POST /login HTTP/1.1Host: 192.168.1.5Connectio 1	1	1
2022-05-28 23:54:56	IP: - [192.168.1.9]	Port: - [80]	GET /login HTTP/1.1Host: 192.168.1.5Upgrade-Insecure-Requests: 1Accept: text/html,application/xhtml+xml,application/xml;		
2022-05-28 23:55:00	IP: - [192.168.1.5]	Port: - [80]	GET /favicon.ico HTTP/1.1Host: 192.168.1.5Connection: keep-aliveUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) A		
2022-05-28 23:55:20	IP: - [192.168.1.9]	Port: - [80]	POST /login HTTP/1.1Host: 192.168.1.5Origin: http	49	

Figure 4. Honeypot logs Excel sheet.

Additionally, the logging package in Python was used to create a logger file that presented the collected data in a different format. Figure 5 illustrates an example of a logger file.

```

2022-05-28 03:52:05,134 Visitor Found! - Time Stamp Logged
2022-05-28 03:52:05,135 IP: - [192.168.1.5]
2022-05-28 03:52:05,136 Port: - [80]
2022-05-28 03:52:05,140 Response Status: 200
2022-05-28 03:52:10,889 Visitor Found! - Time Stamp Logged
2022-05-28 03:52:10,890 IP: - [192.168.1.5]
2022-05-28 03:52:10,890 Port: - [80]
2022-05-28 03:52:10,896 Response Status: 200
2022-05-28 03:52:11,076 Data of Visitor Retrieved
    
```

Figure 5. Python logger file.

A port packet sniffer is used to log traffic from vulnerable open ports in the system. The TCPdump, a command-line-based sniffer, was employed to gather detailed information about incoming connections. The data captured by the TCPdump can be saved in a .pcap format, which is then imported into data analysis tools such as WireShark for further packet analysis. The collected data from the sniffer was appended to both a logger file and a .xlsx file for convenient access and comparison of the incoming connections across various ports. The last two lines in Figure 6 are added by the sniffer, containing the IP address initiating the connection, the accessed port, and two respective values, A and B. These values serve as the trigger inputs for the HT. Monitoring these values and connections helps mitigate attacks.

```

2022-05-28 21:00:28,108 Visitor Found! - Time Stamp Logged
2022-05-28 21:00:28,109 IP: - [192.168.1.11]
2022-05-28 21:00:28,109 Port: - [80]
2022-05-28 21:00:28,109 Response Status: 200
2022-05-28 21:03:14,656 Unknown Incoming data IP- 192.168.1.5 Port- 44444 - Value A: 23
2022-05-28 21:03:18,661 Unknown Incoming data IP- 192.168.1.5 Port- 44444 - Value B: 67
    
```

Figure 6. .pcap file additional logs.

These logs were also entered into a separate sheet .xlsx file in a separate tap “Attack Logs” for a clearer presentation as shown in Figure 7.

A	B	C	D	E
Time Stamp	Visitor IP	Listening on Port	Value A	Value B
2022-05-28 23:33:12	IP: - [192.168.1.5]	Port: - [37020]	1	2
2022-05-28 23:33:20	IP: - [192.168.1.5]	Port: - [37020]	3	4
2022-05-28 23:54:31	IP: - [192.168.1.5]	Port: - [37020]	12	12
2022-05-28 23:54:40	IP: - [192.168.1.5]	Port: - [37020]	78	79
2022-05-28 23:54:48	IP: - [192.168.1.5]	Port: - [37020]	9	0
2022-05-28 23:55:39	IP: - [192.168.1.5]	Port: - [37020]	1	2

Figure 7. Attack logs.

4.2. Hardware Trojan

The hardware Trojan employed in this study was a hybrid design consisting of two separate Trojans based on references [26,42]. The newly established Trojan was then simulated on Logisim and emulated on an FPGA Cyclone IV (D2-115) Intel board.

Figure 8 illustrates a counter-based Trojan, commonly referred to as a “ticking time-bomb” [26]. This architecture served as a case study for our project. The activation mechanism consists of an XOR gate and a k-bit counter. When $a = b = 1$, the k-bit counter is decremented. When $a \neq b$, the counter is incremented. When the counter value exceeds a predefined number N , the output signal is altered.

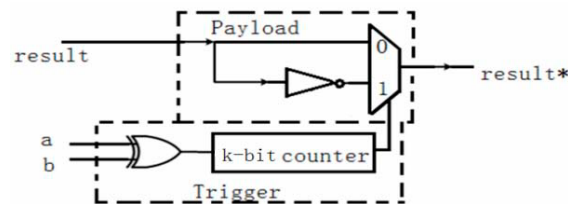


Figure 8. Counter-based Trojan circuit architecture [26].

According to [42], for a system to authenticate user-password pairs x and f , the function $f(x)$ needs to be constructed. With $f(x) = x^2$, the design specifies ten users, I_0 through I_9 . Because there are ten users, the designer utilizes four bits, x_1 to x_4 , to encode them. Because the greatest function output is 81, seven bits, z_1 to z_7 , are required. Figure 9 shows the final circuit.

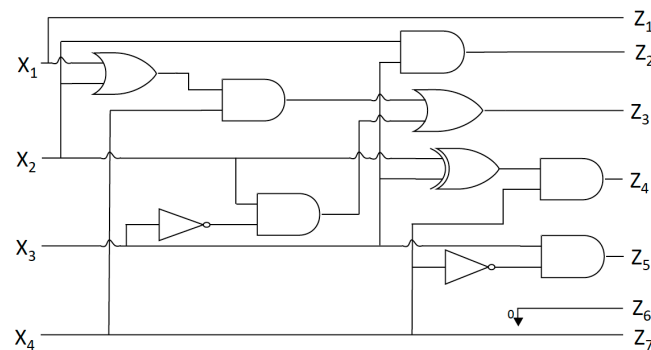


Figure 9. Circuit (a) without the hardware Trojan.

Because the input is 4 bits, we can have 16 users. However, this design does not utilize all of these. Consequently, 6 users are not in use and are in the “don’t care” condition. The designer asserts that the circuit correctly performs the function $f(x) = x^2$ despite this incomplete specification. If an attacker makes the minor modification shown in Figure 10 (in red), the output for the ten valid inputs remains unchanged compared to the original circuit. For inputs 10 and 11, there are two additional correct outputs. Thus, the modified circuit included a backdoor Trojan.

Table 1 presents the circuit outputs. In this study, we designed an HT that combines the two types using the counter-based Trojan result as a selection line for a multiplexer, which determines whether the backdoor Trojan is activated. Figure 11 illustrates the modified circuit, including the counter-based hardware Trojan (HT), as modeled in Logisim. This diagram provides a clearer visualization of the proposed circuit design, which was subsequently implemented using Quartus. The circuit consists of four inputs (x_0 to x_3) representing the binary value of the user number and seven outputs (z_0 to z_6) corresponding to the binary value of the password for the respective user. The K-bit counter represents the counter-based HT integrated into our circuit.

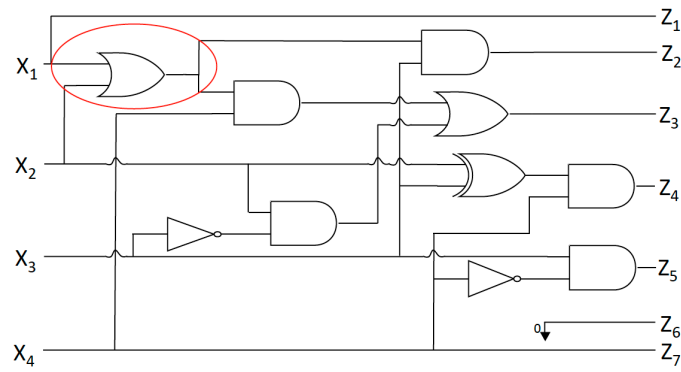


Figure 10. Circuit (b) with the hardware Trojan.

Table 1. Outputs of the circuits.

	Inputs				X	Circuit (a)							F(x)	Circuit (b)							F(x)		
	x ₁	x ₂	x ₃	x ₄		z ₁	z ₂	z ₃	z ₄	z ₅	z ₆	z ₇		z ₁	z ₂	z ₃	z ₄	z ₅	z ₆	z ₇			
Inputs	I ₀	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	I ₁	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1
	I ₂	0	0	1	0	2	0	0	0	0	1	0	0	4	0	0	0	0	1	0	0	0	4
	I ₃	0	0	1	1	3	0	0	0	1	0	0	1	9	0	0	0	1	0	0	1	9	
	I ₄	0	1	0	0	4	0	0	1	0	0	0	0	16	0	0	1	0	0	0	0	16	
	I ₅	0	1	0	1	5	0	0	1	1	0	0	1	25	0	0	1	1	0	0	1	25	
	I ₆	0	1	1	0	6	0	1	0	0	1	0	0	36	0	1	0	0	1	0	0	36	
	I ₇	0	1	1	1	7	0	1	1	0	0	0	1	49	0	1	1	0	0	0	1	49	
	I ₈	1	0	0	0	8	1	0	0	0	0	0	0	64	1	0	0	0	0	0	0	64	
	I ₉	1	0	0	1	9	1	0	1	0	0	0	1	81	1	0	1	0	0	0	1	81	
Undefined	I ₁₀	1	0	1	0	10	1	0	0	0	1	0	0	68	1	1	0	0	1	0	0	100	
	I ₁₁	1	0	1	1	11	1	0	1	1	1	0	1	89	1	1	1	1	0	0	1	121	
	I ₁₂	1	1	0	0	12	1	1	1	0	0	0	0	112	1	0	1	0	0	0	0	80	
	I ₁₃	1	1	0	1	13	1	1	1	1	0	0	1	121	1	0	1	1	0	0	1	89	
	I ₁₄	1	1	1	0	14	1	1	0	0	1	0	0	100	1	1	0	0	1	0	0	100	
	I ₁₅	1	1	1	1	15	1	1	1	0	0	0	1	113	1	1	1	0	0	0	1	113	

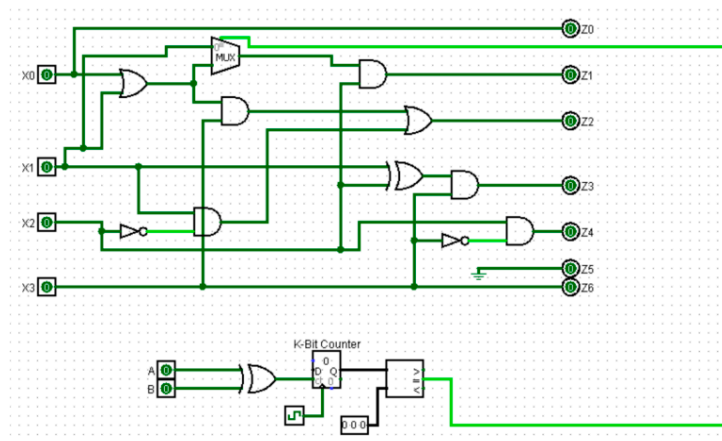


Figure 11. Proposed System Architecture with a counter HT.

The proposed IoT system architecture was simulated using serial programming for FPGA-Raspberry Pi communication. General Purpose Input Output (GPIO) in the FPGA and the Raspberry Pi were used to achieve serial communication between them. Data can be sent from one board to another to facilitate the control of the FPGA using Raspberry Pi. Algorithm A1 in Appendix A describes the Python logic circuit for HT.

Next, Quartus was used to synthesize the Trojan on the FPGA board. The VHSIC Hardware Description Language (VHDL) was used, and the project consisted of four files. Three of these files were for different circuit components, and the main file included all of them. The FPGA accepts inputs a , b , x_0 , x_1 , x_2 and x_3 , which are used by the counter-based and backdoor Trojans, respectively. An LED is assigned to each output from z_0 to z_7 .

If the counter does not exceed a preset value, the circuit operates normally and does not create a backdoor that allows hackers to breach the system. Only ten users can successfully generate a password in this circuit. However, those ten users require four bits to encode, which allows for the creation of six additional undesirable users. When the HT is not enabled, the circuit continues to function normally; however, when triggered, the HT allows two more users to obtain the correct password. This circuit can be assumed to be part of a smart-lock IoT device that only permits ten users to unlock the SmartLock. The generation function for the passwords is $f(x) = x^2$.

Algorithm A2 in Appendix A shows the implementation of the HT in VHDL language on Quartus.

As shown in Figure 12 in Modelsim, the value of the counter changed as a result of XORing A and B together. The Trojan will not be activated unless the counter reaches 7 '111'. A value of 7 is predefined in the code and can be changed as needed. Once this state is reached, the circuit cannot return to its previous configuration because a backdoor has been created.



Figure 12. Hardware Trojan Simulation in ModelSim.

4.3. System Integration

The system schematic in Figure 13 illustrates how the hardware components, specifically the FPGA and Raspberry Pi, are connected for seamless communication between them. A common ground was established between them as well as two 330-ohm resistors due to the DE2-115 board I/O standard mismatch.

Ground (GRD): The GRD pin '39' on the Raspberry Pi and the GRD pin in JP5 on the FPGA were connected to create a common ground.

- **Inputs:**

1. **FPGA:**

1. Pin 'AB 22' is used for transmitting the result of XORing values A and B.
2. Pin 'Y 17' determines whether access is authorized. Authorized access occurs when the password entered on the website matches the correct password

generated for the username and the Trojan has not been triggered. Failure to meet any of these conditions results in unauthorized access.

1. **Raspberry Pi:**
 1. Pin 37 is used by the FPGA to alert the honeypot of a malicious packet if a Trojan has been triggered.
- **Outputs:**
 1. **FPGA:**
 1. Pin 'AC 215' is used to report back to the honeypot that the Trojan is activated.
 1. **Raspberry Pi:**
 1. Pin 40 is used to send the result of the XOR function to the FPGA.
 2. Pin 35 is used to compare the password entered on the website with the correct password generated for the username.
- **Resistors:** Because of the DE2-115 board I/O standard mismatch, 330 Ω pull-down resistors were used to compensate for the mismatched GPIO pin differences.

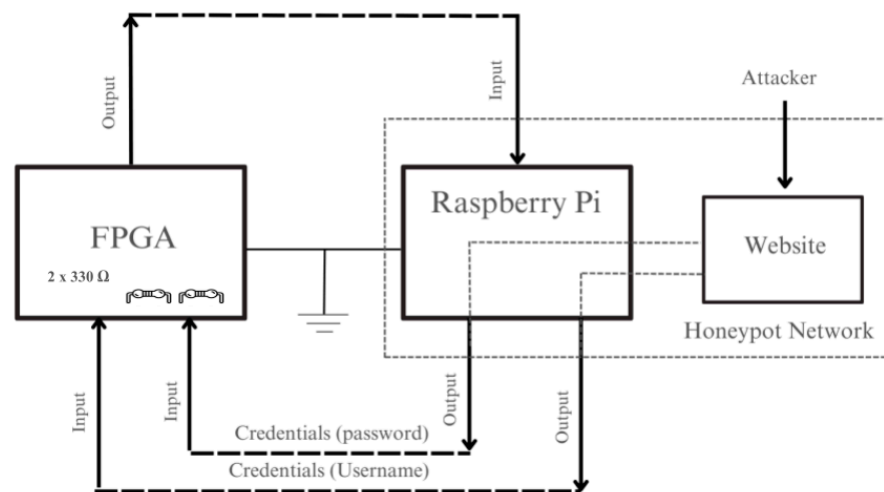


Figure 13. System hardware components schematic.

The Raspberry Pi has two threads running with a time-slicing round-robin scheduling method. To achieve this, the threading and time packages were imported. The two threads running are the main thread and the listener thread. The main function operates as a honeypot, logging usernames and passwords from the website, among other activities. The listener function, which logs the values A and B, operates as a sniffer. Both functions have different vulnerable ports open to lure attackers.

Figure 14 illustrates the process by which an incoming connection undergoes verification to determine whether it constitutes an attack. Any incoming connection is malicious because it is a connection to a decoy and not a real system in the network. However, our honeypot is designed to keep the adversary engaged until undesired behavior is observed. This approach allows us to confirm the presence of a hardware Trojan and identify the key to triggering it. This process begins with the collection and logging of information, such as username and password values, entered on the website using a honeypot. Because the honeypot's IP address is not genuine, any input to the website is automatically flagged as malicious and not intended for legitimate access. When values A and B are entered, their XORing results are transmitted to an emulated FPGA device to monitor their impact on the IoT device's behavior. The emulated device then reports back to the honeypot, indicating whether the Trojan has been activated. If the Trojan is activated, the packet is dropped, and the system proceeds with a dummy login. Otherwise, the honeypot logs the entered data to gather more information regarding the attacker's objectives.

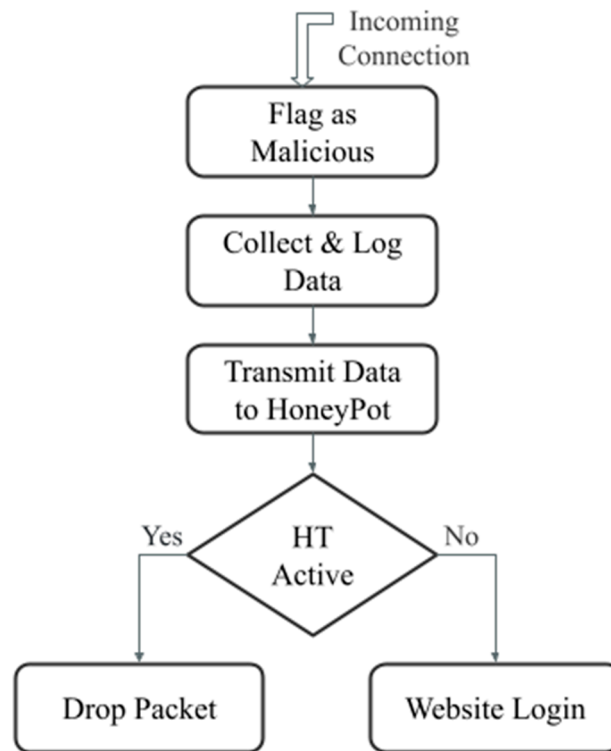


Figure 14. Flowchart of Our HoneyPot System’s Processes.

5. Results and Discussion

5.1. Emulating the Hardware Trojan

The initial proof of concept involved testing the activation of the HT and assessing its impact on an IoT device. The circuit was modified to receive the XOR results for A and B from the honeypot. When the counter reached ‘111’ (seven in decimal), the generated password for the user ‘10’ was altered, indicating Trojan activation. This scenario is depicted in Figures 15 and 16. Figures 17 and 18 show that for input ‘9’, the generated password remains unaffected by the Trojan, demonstrating a selective impact based on specific conditions.

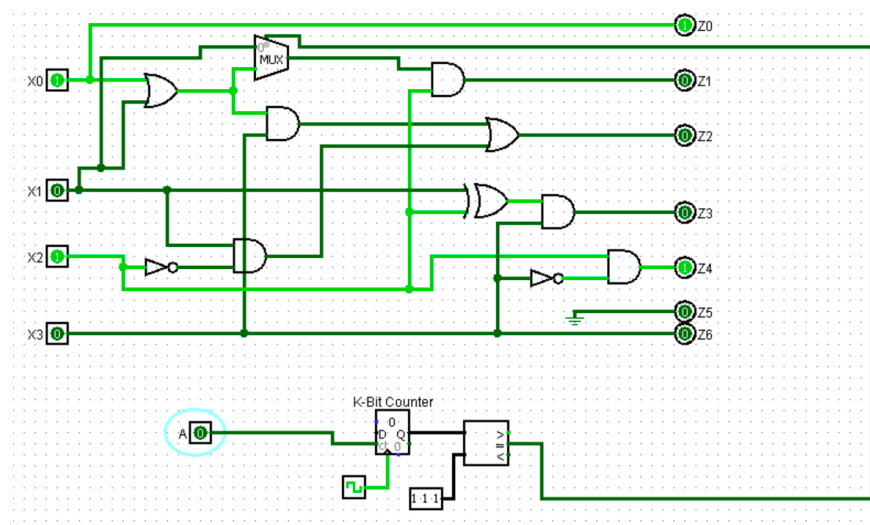


Figure 15. Generated password with trigger off.

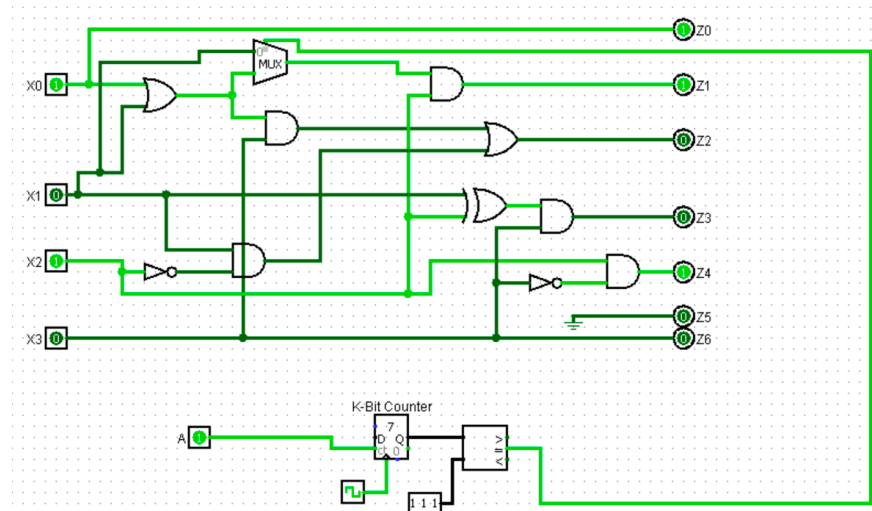


Figure 16. Generated password with trigger on.

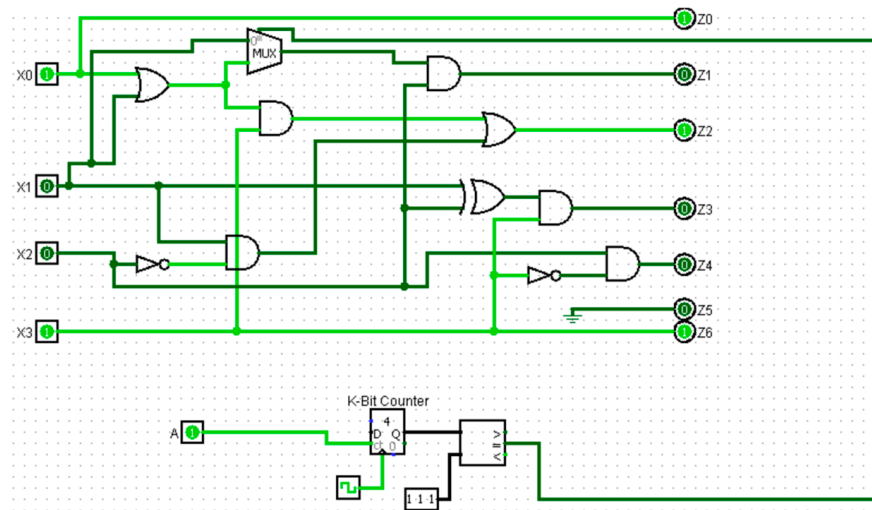


Figure 17. Generated password with trigger off.

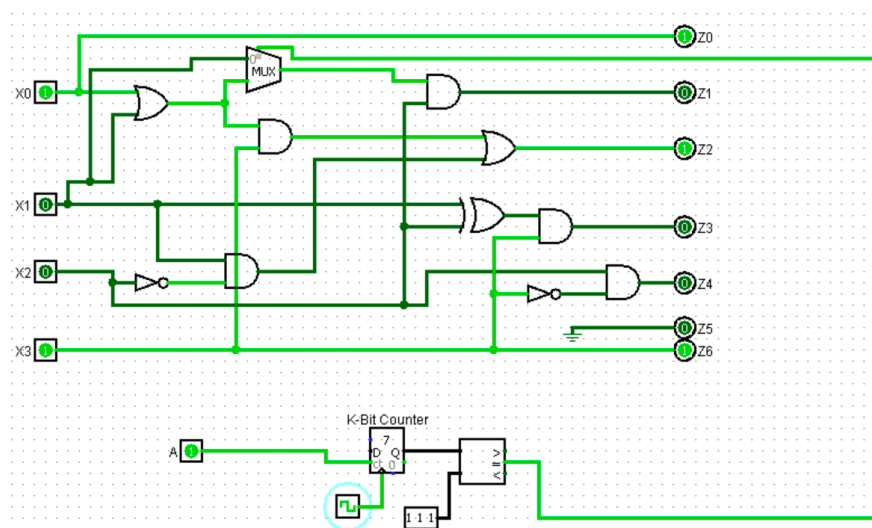


Figure 18. Generated password with trigger on.

5.2. Evasion Testing

Various scenarios were considered to test the proposed system. We categorized the types of attacks into four groups: authorized, unauthorized, triggering, and non-triggering attacks, each with their own testing scenarios.

5.2.1. Authorized Attacks

Authorized attacks involve accessing a website and attempting to log in with valid credentials. The honeypot, which mimics the admin login page of the IoT device ‘Smart-Lock’, logs the entered data and prevents intrusion by reporting to the FPGA. This scenario is illustrated in Figure 19.

```

Visitor Found ! - [ 192.168.1.5 ]
UserName: 2
Password Entered: 4
Password correct !
Safe to Transmit
    
```

Figure 19. Authorized attacks.

5.2.2. Unauthorized Attacks

Unauthorized attacks involve attempts to log in with incorrect credentials. When such an attempt is detected, the honeypot sends a signal to the FPGA, which is displayed as ‘U’ on a 7-segment display, as shown in Figure 20. Figure 21 illustrates the various inputs to the website, highlighting both successful logins with correct credentials and unsuccessful attempts with incorrect ones.

```

Visitor Found ! - [ 192.168.1.5 ]
UserName: 1
Password Entered: 2
Password Incorrect !
Safe to Transmit
    
```

Figure 20. Unauthorized Attack.

A	B	C	D	E	F
Time Stamp	Visitor IP	Listening on Port	Visitor Information	UserName	Password
2022-06-08 18:44:52	IP: - [192.168.1.5]	Port: - [80]	GET / HTTP/1.1 Host: 192.168.1.5 Connection: keep-alive Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9 Accept-Encoding: gzip, deflate Accept-Language: en-US,en;q=0.9		
2022-06-08 18:44:59	IP: - [192.168.1.5]	Port: - [80]	5 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36	2	4
2022-06-08 18:45:34	IP: - [192.168.1.5]	Port: - [80]	HTML, like Gecko) Chrome/102.0.0.0 Safari/537.36	2	3
2022-06-08 18:47:13	IP: - [192.168.1.5]	Port: - [80]	HTML, like Gecko) Chrome/102.0.0.0 Safari/537.36	3	9
2022-06-08 18:47:50	IP: - [192.168.1.5]	Port: - [80]	HTML, like Gecko) Chrome/102.0.0.0 Safari/537.36	6	6
2022-06-08 18:48:12	IP: - [192.168.1.5]	Port: - [80]	HTML, like Gecko) Chrome/102.0.0.0 Safari/537.36	10	100

Figure 21. Logs Gathering.

5.2.3. Triggering Attacks

Triggering attacks occur when the counter reaches its predetermined value, which is seven in this case. Consecutive inputs of values A and B, resulting in an XORing output of 1, trigger the Trojan. The counter is displayed on a 7-segment display, and whenever the Trojan is triggered, it is also considered an unauthorized access, and a ‘U’ will be displayed on the neighboring 7-segment display.

Figure 22 shows the values of A and B entered through the website until the Trojan is activated upon reaching the counter limit of 7. Figure 23 illustrates the process of a packet being dropped when the correct credentials are provided. Figure 24 displays an unauthorized trigger, where incorrect credentials are entered, yet the packet is still dropped. This is because the HT is activated in our honeypot, causing any subsequent packets to be automatically considered malicious and subsequently dropped.

```
message sent!  
Enter you input B:13  
message sent!  
Enter you input A:1  
message sent!  
Enter you input B:2  
message sent!  
Enter you input A:3  
message sent!  
Enter you input B:4  
message sent!  
Enter you input A:67  
message sent!  
Enter you input B:69  
message sent!  
Enter you input A:90  
message sent!  
Enter you input B:100  
message sent!  
Enter you input A:12  
message sent!  
Enter you input B:11  
message sent!  
Enter you input A:0  
message sent!  
Enter you input B:6
```

Figure 22. Triggering The HT.

```
Visitor Found ! - [ 192.168.1.5 ]  
UserName: 2  
Password Entered: 4  
Password correct !  
Trojan is Activated - Drop Packet
```

Figure 23. Triggered Trojan with correct credentials.

```
Visitor Found ! - [ 192.168.1.5 ]  
UserName: 2  
Password Entered: 3  
Password Incorrect !  
Trojan is Activated - Drop Packet
```

Figure 24. Unauthorized triggered.

5.2.4. Non-Triggering Attacks

Non-triggering attacks involve the values of A and B, which do not cause the counter to reach seven. Every entered value is logged, and the attack can be either authorized or unauthorized. Figure 25 shows attempts to trigger the Trojan. If the attacker continues with this pattern, they may eventually reach the triggering stage. However, with the counter currently at two, the Trojan has not yet been activated.

```

20:46:38,642 Visitor Found! - Time Stamp Logged
20:46:38,644 IP: - [192.168.1.5]
20:46:38,644 Port: - [80]
20:46:38,648 Response Status: 200
20:46:48,761 Visitor Found! - Time Stamp Logged
20:46:48,761 IP: - [192.168.1.5]
20:46:48,762 Port: - [80]
20:46:48,766 Response Status: 200
20:46:49,022 Data of Visitor Retrieved
20:46:58,943 Unknown Incoming data IP- 192.168.1.5 Port- 44444 - Value A: 12
20:47:03,036 Unknown Incoming data IP- 192.168.1.5 Port- 44444 - Value B: 35
20:47:07,123 Unknown Incoming data IP- 192.168.1.5 Port- 44444 - Value A: 66
20:47:11,201 Unknown Incoming data IP- 192.168.1.5 Port- 44444 - Value B: 66
    
```

Figure 25. Attempt in triggering the Trojan—logger view.

5.3. TCPdump

A TCPdump was utilized to monitor incoming HTTP POST requests on port 80. After installation, it was tested by pinging the Raspberry Pi from a laptop. Identifying the correct network interface is crucial for accurately sniffing the network traffic, as illustrated in Figure 26.

Below is the final command used after determining the correct interface:

```

$ tcpdump -i wlan0 -s 0 -A 'tcp dst port 80 and tcp[((tcp [12:1] & 0xf0) >> 2):4] = 0x504F5354'
    
```

The results of capturing POST requests are displayed in Figure 27.

```

pi@raspberrypi:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether b8:27:eb:9d:93:43 brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b8:27:eb:c8:c6:16 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.6/24 brd 192.168.1.255 scope global dynamic noprefixroute wlan0
        valid_lft 75371sec preferred_lft 64571sec
    inet6 fd9c:69d1:61c7:b00:6536:38d2:fd69:dff0/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 7145sec preferred_lft 3545sec
    inet6 fe80::c562:afa6:78a1:2aee/64 scope link
        valid_lft forever preferred_lft forever
    
```

Figure 26. Fetching the Correct Interface.

```

pi@raspberrypi:~$ sudo tcpdump -i wlan0 -s 0 -A 'tcp dst port 80 and tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x504F5354'
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
22:50:12.945420 IP 192.168.1.5.57774 > 192.168.1.6.http: Flags [P.], seq 1371962159:1371962763, ack 2020611439, win 513, length 604: HTTP: POST /login HTTP/1.1
E.....PQ.{/xp.oP...B...POST /login HTTP/1.1
Host: 192.168.1.6
Connection: keep-alive
Content-Length: 19
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.1.6
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://192.168.1.6/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

value_a=3&value_b=9
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel
pi@raspberrypi:~$
    
```

Figure 27. TCPdump Capturing Incoming Packets.

5.4. Analysis

Honeypots offer a compelling defense strategy for IoT systems, particularly for addressing the challenges posed by hardware Trojans. By acting as decoys, honeypots lure attackers away from actual devices, facilitating early threat detection and valuable data collection on attack vectors and adversary behavior. They significantly reduce false positives, provide insights into attacker motivations, and enhance incident-response strategies. Importantly, honeypots allow for effective security without overloading IoT devices, whose small controllers struggle with complex security measures, thereby maintaining their performance. This lightweight approach not only protects the network but also enables organizations to implement cost-effective targeted defenses. In this section, we elaborate on the effectiveness of the proposed solution in various attack scenarios, the scalability of the solution, and a comparative analysis with existing work.

5.4.1. System Effectiveness in Various Attack Scenarios

The proposed system was evaluated across multiple scenarios, including authorized, unauthorized, triggering, and non-triggering attacks. The results demonstrate that the honeypot system effectively distinguishes between legitimate and malicious activities, thereby providing a robust security layer for IoT devices.

1. **Authorized Attacks:** The system successfully validated legitimate login attempts, logging critical data for future analysis, and ensuring that no intrusions occurred. In addition, the system employed a fallback mechanism to prevent unauthorized access, even if attackers used valid credentials inappropriately.
2. **Unauthorized Attacks:** In cases of invalid credentials, the system effectively logged all attempts and redirected attackers to a decoy page. This allows the system to gather valuable intelligence on potential threat actors while maintaining system security.
3. **Triggering Attacks:** The hardware Trojan (HT) mechanism was successfully triggered under specific conditions, demonstrating the capability of the system to detect sophisticated HT attacks. The counter-based Trojan activation mechanism ensures selective triggering, thereby minimizing false positives, which is often a challenge in static detection systems.
4. **Non-Triggering Attacks:** The system logged non-triggering attacks without activating the Trojan, ensuring that the IoT device maintained normal operation. This aspect ensures that only real threats are flagged, reducing system noise and unnecessary alerts while enhancing reliability.

5.4.2. Real-Time Packet Analysis and Behavioral Insights

By capturing HTTP POST requests on port 80 and analyzing the incoming traffic, the system provides comprehensive insights into network activity, adding an additional layer of monitoring that helps identify malicious actions as they occur. The detailed logs generated by the honeypot recorded critical data such as IP addresses, ports accessed, and attack attempts. These data offer valuable insights into attacker behavior, which is essential for understanding how future attacks may evolve.

5.4.3. Resource Efficiency and Scalability

The system, implemented on a Raspberry Pi, requires minimal resource consumption. By offloading computationally intensive tasks to an FPGA, the IoT device itself is not burdened by complex detection mechanisms, allowing for efficient operation even in resource-constrained environments. Owing to its low interaction design, the honeypot architecture allows for the deployment of distributed honeypots across various devices in large IoT networks. This design ensures that the system scales easily while maintaining its detection capabilities, making it suitable for wide-scale IoT environments, from smart homes to industrial IoT systems.

5.4.4. Comparative Analysis with Existing Studies

The results of the proposed system were compared with existing research, demonstrating superior performance across critical parameters, such as detection accuracy, resource efficiency, adaptability, and scalability.

1. **Detection Accuracy:** Side-channel analysis [28] achieves high accuracy by analyzing side-channel data such as power consumption and signal delays. However, this approach is computationally expensive and not practical for resource-limited IoT devices. Graph Neural Networks (GNNs) [31], which analyze circuit topologies for HT detection, are effective but require golden IC reference designs, making them complex and difficult to apply across heterogeneous IoT environments. The trigger-based detection mechanism in our system, integrated with a honeypot, offers real-time detection of HTs without adding significant complexity to the IoT device. Unlike side-channel analysis, this approach does not require monitoring physical characteristics, providing a more streamlined solution for IoT environments while still ensuring high detection accuracy.
2. **Resource Efficiency:** Techniques using deep learning models (e.g., VGG-Net) [29] and GNN [31] are effective but are resource-intensive and require significant computational power and memory, which are often unavailable in IoT environments. Our solution, which is designed to operate on low-resource platforms such as Raspberry Pi, is highly efficient. It uses lightweight socket programming for monitoring and logging attacker interactions while offloading more demanding tasks to an FPGA. This ensures that IoT devices are not overwhelmed by complex detection processes, thereby providing a scalable and cost-effective solution.
3. **Adaptability to Different Attack Types:** Static detection methods such as combinatorial testing [24] are effective for specific HT designs but are not adaptable to new or emerging attack vectors. Machine learning-based systems can adapt to evolving threats but require continuous retraining, which is resource-heavy and not ideal for IoT environments. The honeypot-based architecture is Trojan-agnostic, making it adaptable to a wide range of HT designs and attack types. The system is capable of dynamically detecting and logging abnormal behaviors without the need for constant retraining, thereby offering a more flexible and efficient solution for rapidly evolving IoT environments.
4. **Scalability:** High-interaction honeypots, such as HoneyIoT [37] and RIoTPot [39], provide deep engagement with attackers but are resource-intensive and difficult to scale across large IoT networks. Blockchain-based security solutions offer secure communication but struggle with scalability because of their computational requirements. The low-interaction honeypot design of our system ensures scalability without the need for significant resources. It can be deployed across large IoT networks while maintaining high detection capabilities, making it ideal for both small- and large-scale IoT systems.

6. Conclusions and Future Work

6.1. Main Contribution

This study aims to enhance the security of IoT devices by proposing a novel honeypot-based architecture for detecting and mitigating hardware Trojans (HTs). The proposed system, successfully implemented on a Raspberry Pi and integrated with an FPGA, demonstrated an effective hybrid solution for detecting both triggering and non-triggering HT attacks while maintaining the operational integrity of IoT devices.

Key findings from the study include the following:

- The honeypot system effectively identified and logged both authorized and unauthorized access attempts.

- The HT detection mechanism proved to be resource-efficient, making it suitable for deployment in resource-constrained IoT environments.
- Using a low-interaction honeypot design, the system successfully diverted malicious attempts without imposing a significant overhead on IoT devices.

The implications of this research suggest that hardware honeypots provide a flexible and robust defense layer against HT attacks, particularly in environments where traditional detection mechanisms may be too complex or resource-intensive. This study contributes to the broader field of IoT security by offering an adaptable solution for addressing emerging hardware-based threats.

6.2. System Limitations and Future Work

Although the implemented system demonstrated significant strengths, it also highlights several areas for improvement.

- **Scalability:** Although the current system performs effectively in the tested scenarios and is suitable for adaptation to a larger scale, it may encounter challenges when deployed in larger networks with multiple IoT devices. Future studies should explore the integration of distributed honeypots and scalable logging mechanisms to manage larger data volumes and complex attack patterns.
- **Machine Learning Integration:** Incorporating machine learning algorithms can enhance a system's ability to respond adaptively to evolving attack patterns. By analyzing historical attack data, machine learning models can predict and preemptively mitigate new types of attacks, thereby improving the system's proactive defense capabilities.
- **High-Interaction Honeypots:** Transitioning to high-interaction honeypots can provide deeper insights into attacker behavior. These honeypots, which simulate full operating systems and applications, can capture more detailed data on sophisticated attacks, thereby providing more comprehensive security strategies.
- **User Experience and Usability:** Enhancing the user interface to monitor and manage the honeypot system can improve its usability. Providing intuitive dashboards and real-time alerts can help security teams respond quickly and effectively to detected threats.
- **Interoperability with Existing Security Systems:** Ensuring that the honeypot system can seamlessly integrate with existing security infrastructure (e.g., firewalls and intrusion detection/prevention systems) will enhance its effectiveness and facilitate broader adoption in enterprise environments.

Future research should focus on refining the scalability of the system, integrating more advanced logging mechanisms, and exploring the application of artificial intelligence to automate threat detection and response.

Author Contributions: Conceptualization, A.H.E.O. and H.S.; methodology, A.H.E.O. and H.S.; software, A.H.E.O. and H.S.; validation, A.H.E.O., H.S., D.K.M. and A.A.; formal analysis, A.H.E.O., H.S., D.K.M. and A.A.; investigation, A.H.E.O., H.S. and D.K.M.; resources, A.H.E.O., H.S. and D.K.M.; data curation, A.H.E.O. and H.S.; writing—original draft preparation, A.H.E.O., H.S. and D.K.M.; writing—review and editing, A.H.E.O., H.S., D.K.M. and A.A.; visualization, A.H.E.O. and H.S.; supervision, H.S., D.K.M. and A.A.; project administration, H.S., D.K.M. and A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used to support the findings of the study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1. Summary of the related work on IoT security.

Reference	Method/Technique	Strengths of the Study	Weaknesses of the Study
Joel et al., 2023 [11]	Analysis of security challenges in IoT smart homes	Comprehensive analysis of various IoT security challenges	Limited focus on mitigation strategies
Blake et al., 2022 [12]	Security with blockchain technology	Innovative use of blockchain for IoT security	Does not address integration with other security measures
Alawadhi et al., 2022 [13]	Analysis of IoT security risks for businesses	Focuses on business-related IoT risks	Lack of technical details on mitigation
Khanam, 2023 [14]	Review of IoT threats and solutions	Provides a broad overview of IoT threats	Lacks in-depth analysis of specific threats
Haris et al., 2023 [15]	Discussion on IoT security and privacy issues	Highlights various security and privacy issues	Limited empirical data to support claims
Gupta and Lingareddy, 2021 [16]	Security threats and mitigations in IoT	Discusses a range of security threats and solutions	Focuses mainly on theoretical aspects
Bakshi, 2021 [17]	IoT architecture vulnerabilities	Detailed analysis of IoT architecture vulnerabilities	Limited focus on practical solutions
Hromada et al., 2021 [18]	Security aspects of IoT	Comprehensive discussion on IoT security protocols	Limited real-world application examples
Mallik and Jena, 2021 [19]	Analysis of IoT security vulnerabilities	Provides solutions for common IoT vulnerabilities	Focuses on general rather than specific vulnerabilities
Amit et al., 2022 [20]	Study on DDoS attacks on IoT devices	Detailed analysis of DDoS attack methods	Limited focus on preventive measures
Lightbody et al., 2023 [21]	Framework for intrusion detection in IoT	Innovative use of side-channel analysis	Limited scalability for large IoT networks
Mohd Bakry et al., 2022 [22]	Security attack study using Raspberry Pi	Practical demonstration of IoT attacks	Limited scope with a single device model
Neto et al., 2023 [23]	Real-time IoT attack dataset	Provides a comprehensive IoT attack dataset	Limited focus on mitigation strategies
Kampel et al., 2022 [24]	Detection of HTs using combinatorial testing	Effective method for detecting HTs in cryptographic circuits	May not be applicable to all circuit types
Jain et al., 2021 [25]	Survey of HT detection methods	Comprehensive survey of HT detection techniques	Limited practical application examples
Liu et al., 2011 [26]	Design of counter-based HT	Innovative HT design method	Dated methodology, lacks modern context
Shakya et al., 2017 [27]	Benchmarking of HTs	Provides a benchmarking framework for HTs	Limited focus on detection methods
Tang et al., 2023 [28]	HT detection using adversarial networks	High accuracy in HT detection	Complex implementation
Dakhale et al., 2023 [29]	Detection of HTs using VGG-Net	Effective use of neural networks for HT detection	Computationally intensive
Mao et al., 2022 [30]	HT detection using suspicious circuit partition	Novel approach to HT detection	May produce false positives
Hassan et al., 2023 [31]	GNN-based HT detection	High accuracy without a golden IC reference	Requires complex graph learning algorithms
Brunner et al., 2024 [33]	FSM-based hardware honeypot	Realistic imitation of original FSM	May not cover all HT types

Table A1. Cont.

Reference	Method/Technique	Strengths of the Study	Weaknesses of the Study
Wegerer and Tjoa, 2016 [34]	MySQL database honeypot	Effective in deceiving database adversaries	Limited to MySQL databases
Piggin and Buffey, 2016 [35]	Operational technology honeypot	Provides insights into attacker methods	Limited scope, focused on specific technologies
Kibret and Yong, 2013 [36]	Dynamic hybrid virtual honeypot	Combines multiple honeypot types	Complex implementation
Guan et al., 2023 [37]	Adaptive honeypot for IoT using RL	Adapts to evolving threats using RL	High resource requirements
Ellouh et al., 2022 [38]	IoT honeypot for zero-day attacks	Effective against zero-day attacks	Limited real-world deployment
Srinivasa et al., 2021 [39]	Modular hybrid-interaction honeypot	Flexible and modular design	Limited long-term studies
Srinivasa et al., 2022 [40]	Comprehensive honeypot analysis and dataset	Provides extensive data on attack patterns	High complexity in analysis
Xiaoming et al., 2022 [41]	Lightweight honeynet for IoT	Cost-effective and scalable	Limited to lightweight applications

Algorithm A1. Python Logic Circuit for Hardware Trojan

Input: GPIO pin 10, GPIO pin 12 ▷ Input pins

GPIO pin 18 ▷ Output pin (payload control)

Output: Trigger activation (GPIO pin 18 set to HIGH)

Initialize variables:

input_a, input_b, Counter ▷ 0.

Define logic gates:

Define *ANDGate(a, b)* ▷ Returns True if a and b are both 1.

Define *XORGate(a, b)* ▷ Returns True if exactly one of a or b is 1.

While True do:

 Check GPIO pin 10

if pin 10 is HIGH **then**

 set *input_a* to 1

else

 set *input_a* to 0

endif

 Check GPIO pin 12

if pin 12 is HIGH **then**

 set *input_b* to 1

else

 set *input_b* to 0

endif

 Logic Operations

if *XORGate(input_a, input_b)* is True **then**

 increment *Counter* by 1.

endif

if *ANDGate(input_a, input_b)* is True **then**

 decrement *Counter* by 1.

endif

 Trigger condition

if *Counter* is 20 **then**

 call *Trigger()*

endif

Algorithm A2. Hardware Trojan Architecture in VHDL**Input:** A, B ▷ Control signals

CLK ▷ Clock signal

count ▷ Counter value

 v ▷ Counter Limit value**Output:** LEDs, leds ▷ 7-bits binary value**Stage 0:** Up-Down Counter ▷ controlled by inputs A and B and synchronized with the CLK signal. The counter produces the value count.**Stage 1:** Comparator ▷ Compare the count value with v **Stage 2:** Multiplexer ▷ 2-to-1 multiplexer controls the LEDs.**Logic Computations:**Assign $Z = \{z0, z1, \dots, z6\}$ ▷ with the corresponding circuit design

Assign LEDs ▷ Z

Begin Process

#based on count, the corresponding LED segment are activated

When count is "000"

Activate leds ▷ "1000000"

When count is "001"

Activate leds ▷ "1111001"

When count = "010"

Activate leds ▷ "0100100"

When count = "011"

Activate leds ▷ "0110000"

When count = "100"

Activate leds ▷ "0011001"

When count = "101"

Activate leds ▷ "0010010"

When count = "110"

Activate leds ▷ "0000010"

When count = "111"

Activate leds ▷ "1111000"

Otherwise

Activate leds ▷ "1111000"

References

1. Alyasiri, H.; Clark, J.A.; Malik, A.; de Fréin, R. Grammatical evolution for detecting cyberattacks in internet of things environments. In Proceedings of the International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 19–22 July 2021; pp. 1–6. [\[CrossRef\]](#)
2. Suber, J.G.; Zantua, M. Intelligent Interaction Honey pots for Threat Hunting within the Internet of Things. *J. Colloq. Inf. Syst. Secur. Educ.* **2022**, *9*, 1–5. [\[CrossRef\]](#)
3. Khraisat, A.; Alazab, A. A critical review of intrusion detection systems in the internet of things: Techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity* **2021**, *4*, 18. [\[CrossRef\]](#)
4. Kumar, A.; Abhishek, K.; Ghalib, M.R.; Shankar, A.; Cheng, X. Intrusion detection and prevention system for an IoT environment. *Digit. Commun. Netw.* **2022**, *8*, 540–551. [\[CrossRef\]](#)
5. Pa, Y.M.P.; Suzuki, S.; Yoshioka, K.; Matsumoto, T.; Kasama, T.; Rossow, C. IoTPOT: A novel honeypot for revealing current IoT threats. *J. Inf. Process.* **2016**, *24*, 522–533. [\[CrossRef\]](#)
6. Almohannadi, H.; Awan, I.; Al Hamar, J.; Cullen, A.; Disso, J.P.; Armitage, L. Cyber Threat Intelligence from Honeypot Data Using Elasticsearch. In Proceedings of the 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA), Krakow, Poland, 16–18 May 2018; pp. 900–906. [\[CrossRef\]](#)
7. Hakim, M.A.; Aksu, H.; Uluagac, A.S.; Akkaya, K. U-PoT: A Honeypot Framework for UPnP-Based IoT Devices. In Proceedings of the 2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC), Orlando, FL, USA, 17–19 November 2018; pp. 1–8. [\[CrossRef\]](#)
8. Shahin, S.; Soubra, H. An IoT Adversary Emulation prototype tool. In Proceedings of the 2022 5th International Conference on Information and Computer Technologies (ICICT), New York, NY, USA, 4–6 March 2022; pp. 7–12. [\[CrossRef\]](#)
9. Sabry, N.; Abobkr, M.; ElHayani, M.; Soubra, H. A Cyber-Security Prototype Module for Smart Bikes. In Proceedings of the 2021 16th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 15–16 December 2021; pp. 1–5. [\[CrossRef\]](#)

10. Elhousseiny, N.; Sabry, M.; Soubra, H. B2X Multiprotocol Secure Communication System for Smart Autonomous Bikes. In Proceedings of the 2023 IEEE Conference on Power Electronics and Renewable Energy (CPERE), Luxor, Egypt, 19–21 February 2023; pp. 1–6. [\[CrossRef\]](#)
11. Joel, M.R.; Manikandan, G.; Bhuvaneswari, G. An Analysis of Security Challenges in Internet of Things (IoT) based Smart Homes. In Proceedings of the 2023 Second International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 2–4 March 2023; pp. 490–497. [\[CrossRef\]](#)
12. Bryant, B.; Saiedian, H. Key challenges in security of IoT devices and securing them with the blockchain technology. *Secur. Priv.* **2022**, *5*, 5. [\[CrossRef\]](#)
13. Alawadhi, J.; AlJanabi, A.M.; Khder, M.A.; Ali, B.J.A.; Al-Shalabi, R.F. Internet of Things (IoT) Security Risks: Challenges for Business. In Proceedings of the 2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), Manama, Bahrain, 22–23 June 2022; pp. 450–456. [\[CrossRef\]](#)
14. Khanam, R. Review of Threats in IoT Systems: Challenges and Solutions. *Int. J. Sci. Technol. Eng.* **2023**, *11*, 325–341. [\[CrossRef\]](#)
15. Haris, M.A.H.B.; Yahya, M.I.H.B.; Ibrahim, M.N.B. Security and Privacy Issues in Internet of Things (IoT). *TechRxiv* **2023**. [\[CrossRef\]](#)
16. Gupta, S.; Lingareddy, N. Security Threats and Their Mitigations in IoT Devices. In *Techno-Societal 2020*; Pawar, P.M., Balasubramaniam, R., Ronge, B.P., Salunkhe, S.B., Vibhute, A.S., Melinamath, B., Eds.; Springer: Cham, Switzerland, 2021. [\[CrossRef\]](#)
17. Bakshi, G. IoT Architecture Vulnerabilities and Security Measures. In *Security Incidents & Response Against Cyber Attacks*; Bhardwaj, A., Sapra, V., Eds.; EAI/Springer Innovations in Communication and Computing; Springer: Cham, Switzerland, 2021. [\[CrossRef\]](#)
18. Hromada, D.; Costa, R.L.; Santos, L.; Rabadão, C. Security Aspects of the Internet of Things. In *IoT Protocols and Applications for Improving Industry, Environment, and Society*; González García, C., García-Díaz, V., Eds.; IGI Global: Harrisburg, PA, USA, 2021; pp. 207–233. [\[CrossRef\]](#)
19. Mallik, P.; Jena, O.P. Analysis of Security Vulnerabilities of Internet of Things and It's Solutions. In *Intelligent Systems*; Udgata, S.K., Sethi, S., Srirama, S.N., Eds.; Lecture Notes in Networks and Systems; Springer: Singapore, 2021; Volume 185. [\[CrossRef\]](#)
20. Amit; Dhingra, A.; Sangwan, A.; Sindhu, V. DDOS Attack on IOT devices. *Int. J. Circuit. Comput. Netw.* **2022**, *3*, 33–42. [\[CrossRef\]](#)
21. Lightbody, D.; Ngo, D.-M.; Temko, A.; Murphy, C.C.; Popovici, E. Attacks on IoT: Side-Channel Power Acquisition Framework for Intrusion Detection. *Future Internet* **2023**, *15*, 187. [\[CrossRef\]](#)
22. Mohd Bakry, B.B.; Bt Adenan, A.R.; Mohd Yusoff, Y.B. Security Attack on IoT Related Devices Using Raspberry Pi and Kali Linux. In Proceedings of the 2022 International Conference on Computer and Drone Applications (IconDA), Kuching, Malaysia, 28–29 November 2022; pp. 40–45. [\[CrossRef\]](#)
23. Neto, E.C.P.; Dadkhah, S.; Ferreira, R.; Zohourian, A.; Lu, R.; Ghorbani, A.A. CICIOT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors* **2023**, *23*, 5941. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Kampel, L.; Kitsos, P.; Simos, D.E. Locating Hardware Trojans Using Combinatorial Testing for Cryptographic Circuits. *IEEE Access* **2022**, *10*, 18787–18806. [\[CrossRef\]](#)
25. Jain, A.; Zhou, Z.; Guin, U. Survey of Recent Developments for Hardware Trojan Detection. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 22–28 May 2021; pp. 1–5. [\[CrossRef\]](#)
26. Liu, H.; Luo, H.; Wang, L. Design of hardware trojan horse based on counter. In Proceedings of the 2011 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering, Xi'an, China, 17–19 June 2011; pp. 1007–1009. [\[CrossRef\]](#)
27. Shakya, B.; He, T.; Salmani, H.; Forte, D.; Bhunia, S.; Tehranipoor, M. Benchmarking of Hardware Trojans and Maliciously Affected Circuits. *J. Hardw. Syst. Secur.* **2017**, *1*, 85–102. [\[CrossRef\]](#)
28. Tang, W.; Su, J.; Gao, Y. Hardware Trojan Detection Method Based on Dual Discriminator Assisted Conditional Generation Adversarial Network. *J. Electron. Test.* **2023**, *39*, 129–140. [\[CrossRef\]](#)
29. Dakhale, B.; Vipinkumar, K.; Narotham, K.; Kadam, S.; Bhurane, A.A.; Kothari, A.G. Automated Detection of Hardware Trojans using Power Side-Channel Analysis and VGG-Net. In Proceedings of the 2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS), Nagpur, India, 5–6 April 2023; pp. 1–5. [\[CrossRef\]](#)
30. Mao, J.; Jiang, X.; Liu, D.; Chen, J.; Huang, K. A Hardware Trojan-Detection Technique Based on Suspicious Circuit Block Partition. *Electronics* **2022**, *11*, 4138. [\[CrossRef\]](#)
31. Hassan, R.; Meng, X.; Basu, K.; Dinakarao, S.M.P. Circuit Topology-Aware Vaccination-Based Hardware Trojan Detection. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 2852–2862. [\[CrossRef\]](#)
32. Yasaei, R.; Chen, L.; Yu, S.-Y.; Faruque, M.A.A. Hardware trojan detection using graph neural networks. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**; 1–14. [\[CrossRef\]](#)
33. Brunner, M.; Lee, H.H.; Hepp, A.; Baehr, J.; Sigl, G. Hardware HoneyPot: Setting Sequential Reverse Engineering on a Wrong Track. In Proceedings of the 2024 27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS), Kielce, Poland, 3–5 April 2024; pp. 47–52. [\[CrossRef\]](#)
34. Wegerer, M.; Tjoa, S. Defeating the Database Adversary Using Deception—A MySQL Database HoneyPot. In Proceedings of the 2016 International Conference on Software Security and Assurance (ICSSA), Saint Pölten, Austria, 24–25 August 2016; pp. 6–10. [\[CrossRef\]](#)
35. Piggini, R.; Buffey, I. Active defence using an operational technology honeypot. In Proceedings of the 11th International Conference on System Safety and Cyber-Security (SSCS 2016), London, UK, 11–13 October 2016; pp. 1–6. [\[CrossRef\]](#)

36. Kibret, Y.; Yong, W. Design and Implementation of Dynamic Hybrid Virtual HoneyPot Architecture for Attack Analysis. *Int. J. Netw. Distrib. Comput.* **2013**, *1*, 108–123. [[CrossRef](#)]
37. Guan, C.; Liu, H.; Cao, G.; Zhu, S.; La Porta, T. HoneyIoT: Adaptive High-Interaction HoneyPot for IoT Devices Through Reinforcement Learning. In Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23), Guildford, UK, 29 May–1 June 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 49–59. [[CrossRef](#)]
38. Ellouh, M.; Ghaleb, M.; Felemban, M. IoTZeroJar: Towards a HoneyPot Architecture for Detection of Zero-Day Attacks in IoT. In Proceedings of the 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 4–6 December 2022; pp. 765–771. [[CrossRef](#)]
39. Srinivasa, S.; Pedersen, J.M.; Vasilomanolakis, E. RIOTPot: A Modular Hybrid-Interaction IoT/OT HoneyPot. In Proceedings of the the 26th European Symposium on Research in Computer Security (ESORICS), Darmstadt, Germany, 4–8 October 2021; pp. 1–7. [[CrossRef](#)]
40. Srinivasa, S.; Pedersen, J.M.; Vasilomanolakis, E. Interaction matters: A comprehensive analysis and a dataset of hybrid IoT/OT honeypots. In Proceedings of the 38th Annual Computer Security Applications Conference (ACSAC '22), Austin, TX, USA, 5–9 December 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 742–755. [[CrossRef](#)]
41. Lu, X.-X.; Yu, X.-N.; Liu, Y.-Z.; Zhang, M. Dynamic Deployment Model of Lightweight HoneyNet for Internet of Things. In Proceedings of the 2022 International Conference on 6G Communications and IoT Technologies (6GIoTT), Fuzhou, China, 23–25 December 2022; pp. 30–34. [[CrossRef](#)]
42. Moein, S.; Gulliver, T.A.; Gebali, F.; Alkandari, A. A New Characterization of Hardware Trojans. *IEEE Access* **2016**, *4*, 2721–2731. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.