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

Amira Houssam , Hassan Soubra , Donatien Koulla Moulla * , Alain Abran

Posted Date: 18 July 2024

doi: 10.20944/preprints202407.1498.v1

Keywords: IoT; Honeypots; Hardware Trojan; VHDL; FPGA; Raspberry Pi; Python; Socket Programming

Article

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

Amira Houssam [1], Hassan Soubra [2], Donatien Koulla Moulla [3,*] and Alain Abran [4]

[1] The German University in Cairo, New Cairo, Egypt; amira.h.roshdy@gmail.com
[2] 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; moulldk@unisa.ac.za
[4] Department of Software Engineering and Information Technology, Ecole de Technologie Supérieure, Montréal, 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 study 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 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. The 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

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 focused mainly on their functional aspects, cybersecurity threats have become a major concern. The proliferation of insecure IoT devices has created vulnerabilities that can be exploited in large-scale cyberattacks. Ensuring the security of these devices is becoming a top priority for preventing major incidents, such as the Mirai botnet attack in 2016 [2]. Intrusion Detection Systems (IDS) [3] and Intrusion Prevention Systems (IPS) [4] can be used to address these cyberattacks. Honeypots [5], decoy systems designed to mimic real systems, applications, or services, are gaining popularity as threat hunting tools for securing IoT systems. These deceptive lures attract attackers, allowing security professionals to study their tactics and techniques without compromising their real assets. Understanding and mitigating cyber threats are crucial and complex process [6]. The heterogeneous nature of IoT devices has resulted in serious security problems for users. A significant number of consumer-grade IoT devices lack robust security measures, increasing their susceptibility to privacy breaches and cyberattacks. Hence, it is crucial to understand the security issues and attack vectors in the IoT domain. Although significant research has been conducted to secure traditional computing systems, little focus has been placed on the IoT realm [7].

 doi:10.20944/preprints202407.1498.v1

The IoT paradigm has become omnipresent owing to its multiple applications, including Intelligent Transportation Systems, Smart Homes, Smart Cities. Securing IoT devices is essential [8]. Moreover, there has been significant growth in research aimed at securing connected cyber-physical systems, such as the smart autonomous bike in [9,10].

In this context, Hardware Trojans (HT) are malicious, unwanted, and deliberate modifications of the electronic circuits of IoT devices. They can alter functionality, degrade performance, and leak information. In this study, we propose a hardware defense layer for IoT devices against HT using honeypots.

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

2. Literature Review

This literature review provides a comprehensive analysis of the current state of research related to IoT security, particularly focusing 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 risks [13]. The unique architecture of IoT networks creates new vulnerabilities and cyber threats, demanding a deep understanding of these challenges and 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]. Additionally, 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 (IDS) 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

 doi:10.20944/preprints202407.1498.v1

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]. In addition, 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 (HT)

2.2.1. Definition and Classification of HTs

A HT is a malicious, unwanted, and deliberate modification of an electronic circuit. 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 the 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. As a result, 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 result in 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 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 Trojan makes detection even more difficult.
- **Analog (RF Trojan):** The trigger circuit is designed with capacitors that are activated by accumulating the charge from the toggling of a nearby victim wire that exceeds a specified threshold.

The design types for 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

 doi:10.20944/preprints202407.1498.v1

Detection mechanisms for HT encompass various approaches including 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 (GNN) [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 the 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 detect Trojans effectively without requiring a golden IC reference design, demonstrating high accuracy rates even on unseen Trojan benchmarks [32].

2.4. Honeypots

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 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 the model. This is intended to persuade the adversary that they have gained access to the actual system and encourage them to spend time within this restricted environment.

The researchers in [33] introduced a hardware honeypot called the Finite State Machine Honeypot (FSM-HP). The purpose is to imitate the original FSM by accepting identical inputs, aiming to appear highly realistic and evade detection. However, the FSM-HP is intentionally equipped with additional vulnerabilities to enhance its appeal to potential attackers. A honeypot can be reprogrammed in the event that 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 protecting against adversaries.

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

1. **Purpose-Based:**
    - **Research Honeypot:** 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 Honeypot:** The common type of honeypot is the production honeypot. Businesses use this decoy to gather information and intelligence regarding cyberattacks on their production networks. The collected data can include IP addresses, intrusion attempt times and dates, traffic volume, and other attributes. Production honeypots are relatively simple to design and deploy, but they produce less sophisticated intelligence than research honeypots.
2. **Levels of Interactivity-Based:**
    - **Low-Interaction Honeypot:** Low-interaction Honeypots use a small number of resources to simulate parts of a system's software or network services while gathering basic information about the attacker. Because of their limited capabilities, attackers cannot escape emulation; thus, no host system can 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

 doi:10.20944/preprints202407.1498.v1

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 is running on a real machine connected to the network 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 the lack of industry-wide security standards. While 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, few studies have focused on the detection and mitigation of HT through the use of honeypots. Table 1 presents some recent literature on IoT security, with a summary of their strengths and weaknesses.

Table 1. Summary of the related work on IoT security.

Reference	Method/Technique	Strengths of the Study	Weaknesses of the Study
[11] Joel et al., 2023	Analysis of security challenges in IoT smart homes	Comprehensive analysis of various IoT security challenges	Limited focus on mitigation strategies
[12] Blake et al., 2022	Security with blockchain technology	Innovative use of blockchain for IoT security	Does not address integration with other security measures
[13] Alawadhi et al., 2022	Analysis of IoT security risks for businesses	Focuses on business-related IoT risks	Lack of technical details on mitigation

 doi:10.20944/preprints202407.1498.v1

[14] Khanam, 2023	Review of IoT threats and solutions	Provides a broad overview of IoT threats	Lacks in-depth analysis of specific threats
[15] Haris et al., 2023	Discussion on IoT security and privacy issues	Highlights various security and privacy issues	Limited empirical data to support claims
[16] Gupta and Lingareddy, 2021	Security threats and mitigations in IoT	Discusses a range of security threats and solutions	Focuses mainly on theoretical aspects
[17] Bakshi, 2021	IoT architecture vulnerabilities	Detailed analysis of IoT architecture vulnerabilities	Limited focus on practical solutions
[18] Hromada et al., 2021	Security aspects of IoT	Comprehensive discussion on IoT security protocols	Limited real-world application examples
[19] Mallik and Jena, 2021	Analysis of IoT security vulnerabilities	Provides solutions for common IoT vulnerabilities	Focuses on general rather than specific vulnerabilities
[20] Amit et al., 2022	Study on DDoS attacks on IoT devices	Detailed analysis of DDoS attack methods	Limited focus on preventive measures
[21] Lightbody et al., 2023	Framework for intrusion detection in IoT	Innovative use of side-channel analysis	Limited scalability for large IoT networks
[22] Mohd Bakry et al., 2022	Security attack study using Raspberry Pi	Practical demonstration of IoT attacks	Limited scope with a single device model
[23] Neto et al., 2023	Real-time IoT attack dataset	Provides a comprehensive IoT attack dataset	Limited focus on mitigation strategies
[24] Kampel et al., 2022	Detection of HTs using combinatorial testing	Effective method for detecting HTs in cryptographic circuits	May not be applicable to all circuit types
[25] Jain et al., 2021	Survey of HT detection methods	Comprehensive survey of HT detection techniques	Limited practical application examples
[26] Liu et al., 2011	Design of counter-based HT	Innovative HT design method	Dated methodology, lacks modern context
[27] Shakya et al., 2017	Benchmarking of HTs	Provides a benchmarking framework for HTs	Limited focus on detection methods
[28] Tang et al., 2023	HT detection using adversarial networks	High accuracy in HT detection	Complex implementation
[29] Dakhale et al., 2023	Detection of HTs using VGG-Net	Effective use of neural networks for HT detection	Computationally intensive
[30] Mao et al., 2022	HT detection using suspicious circuit partition	Novel approach to HT detection	May produce false positives
[31] Hassan et al., 2023	GNN-based HT detection	High accuracy without a golden IC reference	Requires complex graph learning algorithms
[33] Brunner et al., 2024	FSM-based hardware honeypot	Realistic imitation of original FSM	May not cover all HT types
[34] Wegerer and Tjoa, 2016	MySQL database honeypot	Effective in deceiving database adversaries	Limited to MySQL databases

 doi:10.20944/preprints202407.1498.v1

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

Given the extensive work presented in Table 1, it is evident that 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 makes 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 the adversary's intentions. This is achieved without introducing overhead to the IoT system or requiring knowledge of its internal workings. Our approach is trojan-agnostic and fully customizable to meet specific security needs, 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 that are 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 on the LAN, determining whether incoming packets will be forwarded to the IoT device or dropped. The logging layer then stores the captured data for analysis. The honeypot is strategically deployed within a controlled network environment to ensure that it appears as a legitimate device, thus enabling attackers to interact with it. It monitors incoming traffic, captures any malicious activities, and logs the details for further analysis. The honeypot uses Python socket programming for network communication, and integrates a packet sniffer to detect and log malicious packets.

 doi:10.20944/preprints202407.1498.v1

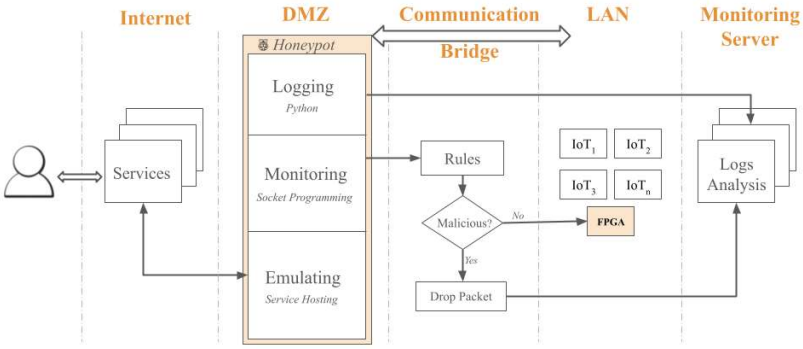


Figure 1. The Honeypot System architecture.

3.2. HT Design

Trigger-based HT are a type of malicious modification or addition to hardware components, particularly integrated circuits (ICs), 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. Honeypot

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 a username and password. The honeypot records all the 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 reaches out to establish a connection. When the client attempts to contact the server, the server creates a listener socket. The following scripts demonstrate how a 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

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 July 2024 doi:10.20944/preprints202407.1498.v1


```
try:
        get_socket_con = socket(AF_INET, SOCK_STREAM)
        get_socket_con.bind((ip_add, port))
        get_socket_con.listen(10) # timeout after 10 seconds
        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 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 for incoming connections with a timeout set to 10 seconds. 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 a 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, as shown in script 2.

Script 2

```
except KeyboardInterrupt as key:
    print("[-] process stopped !")
finally:
    get_socket_con.close()
get_socket_con.close()
```

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' into any web browser. The attacker is then routed to a webpage designed using JavaScript and CSS. As illustrated in Figure 2, there are two input text boxes representing 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.

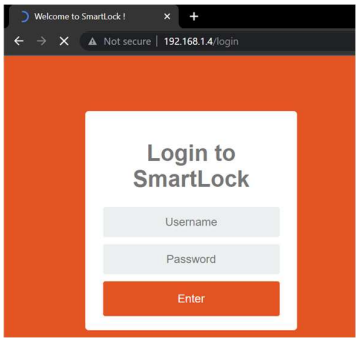


Figure 2. SmartLock Fake Webpage.

String manipulation techniques were applied to extract the values of the username and password. Data are 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 the 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 every 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.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 July 2024 doi:10.20944/preprints202407.1498.v1

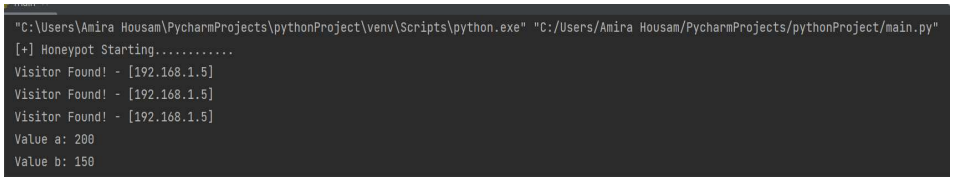


Figure 3. Output of the terminal.

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). Additionally, 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 mentioned above. 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.

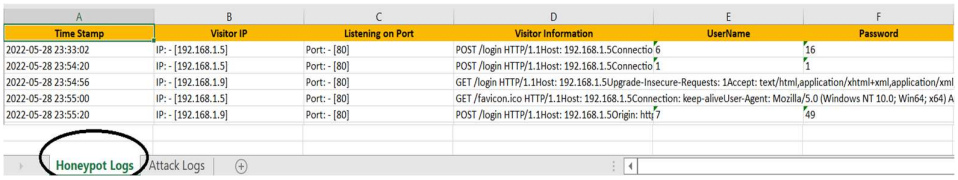


Figure 4. Honeypot logs Excel sheet.

In addition, 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.

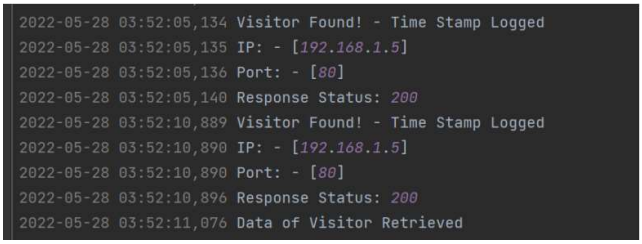


Figure 5. Python logger file.

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 the 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 of the sniffer are 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, accessed port, and two respective values, A and B. These values serve as trigger inputs for the HT. Monitoring these values and connections helps to mitigate attacks.

 doi:10.20944/preprints202407.1498.v1


```
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.

Time Stamp	Visitor IP	Lisenting 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

Honeypot Logs | Attack Logs

Figure 7. Attack logs.

4.2. Hardware Trojan

The hardware Trojan employed in this study is 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 serves 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.

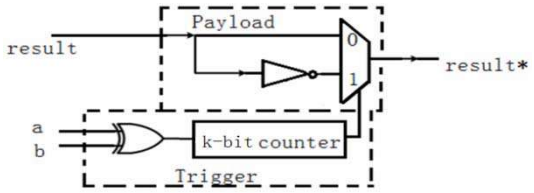


Figure 8. Counter-based Trojan Circuit Architecture [26].

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.

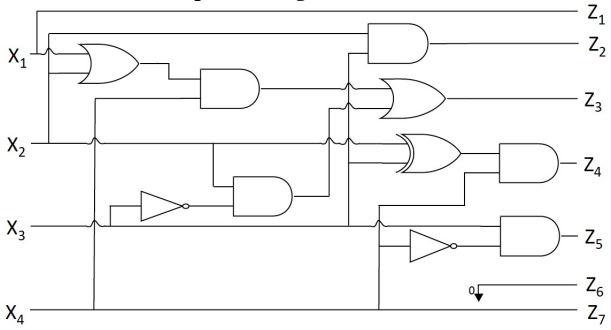


Figure 9. Circuit (a) without the Hardware Trojan.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 July 2024 doi:10.20944/preprints202407.1498.v1

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.

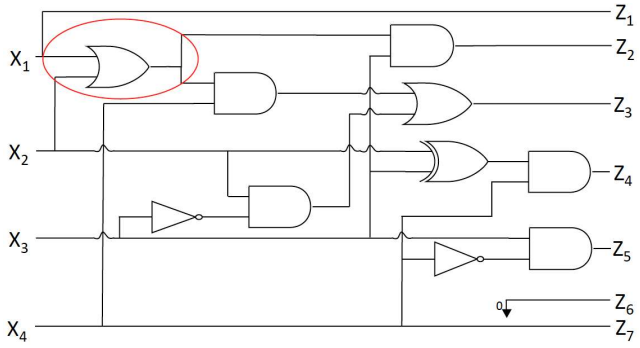


Figure 10. Circuit (b) with the Hardware Trojan.

Figure 11 presents the outputs of the circuits.

		Inputs					Circuit (a)								Circuit (b)							
		X_1	X_2	X_3	X_4	X	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	$F(X)$	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	$F(X)$
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	1	1
	I_2	0	0	1	0	2	0	0	0	0	1	0	0	4	0	0	0	0	1	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	0	0	0	0	1	81	1	0	0	1	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	0	1	0	0	1	89	1	1	0	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. Outputs of the circuits.

In this study, we designed an HT that combines the two types by using the counter-based Trojan result as a selection line for a multiplexer, which determines whether the backdoor Trojan is activated. Figure 12 shows the circuit of the HT in Logisim. The switches in Box 2 were used to enter the user number to generate the corresponding password. Box 1 contains LEDs that represent the values of z_1 to z_7 from left to right. In addition, the 7-segment display in Box 3 shows the current value of the counter.

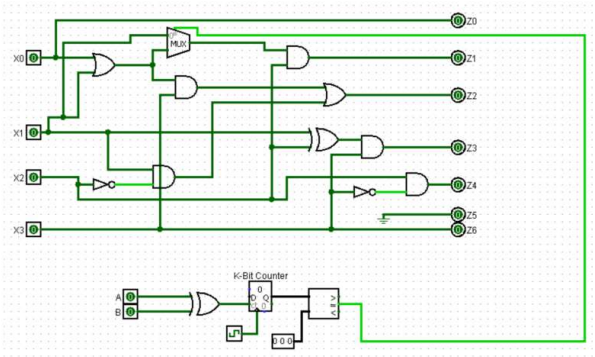


Figure 12. Proposed System Architecture with a counter HT.

 doi:10.20944/preprints202407.1498.v1

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 the other to facilitate control of the FPGA using Raspberry Pi (see script 3).

Script 3. Hardware Trojan on Raspberry Pi in Python

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(10, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(12, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(18, GPIO.OUT)

input_a = 0
input_b = 0
Counter = 0
##############################
# CASE Hardware Trojan
##############################

def AND(a, b):
    if a == 1 and b == 1:
        return True
    else:
        return False

def XOR(a, b):
    if a != b:
        return 1
    else:
        return 0

def Trigger():
    GPIO.output(18, GPIO.HIGH) # Payload

while True: # Run forever
    if GPIO.input(10) == GPIO.HIGH:
        input_a = 1
    else:
        input_a = 0
    if GPIO.input(12) == GPIO.HIGH:
        input_b = 1
    else:
        input_b = 0
    if XOR(input_a, input_b):
        Counter = Counter + 1
    if AND(input_a, input_b):
        Counter = Counter - 1
    if Counter == 20:
        Trigger()
```

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 the 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 will continue to function normally; however, when triggered, the HT will allow 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 $f(x) = x^2$.

Script 4 shows the implementation of the HT in VHDL language on Quartus.

Script 4. Hardware Trojan Architecture in VHDL

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 July 2024 doi:10.20944/preprints202407.1498.v1


```
stage0: updowncounter PORT MAP (A,B,CLK,count);
stage1: compare PORT MAP (count , v , equal);
stage2: mux2to1 PORT MAP (equal,XOOrX1,x(1),X1Mux);

XOOrX1 <= x(0) OR x(1);
z(0) <= x(0);
z(1) <= X1Mux AND x(2);
z(2) <= ( (x(0) OR x(1)) AND ( x(3) ) ) OR ( NOT x(2) AND x(1) );
z(3) <= (x(1) XOR x(2)) AND ( x(3) );
z(4) <= NOT x(3) AND x(2) ;
z(5) <= '0'; --ground
z(6) <= x(3);
LEDS <= z;

Process(A,B)
    BEGIN
       case count is         --abcdefg
          when "000" =>leds1<= "1000000";--0
          when "001" =>leds1<= "1111001";--1
          when "010" =>leds1<= "0100100";--2
          when "011" =>leds1<= "0110000";--3
          when "100" =>leds1<= "0011001";--4
          when "101" =>leds1<= "0010010";--5
          when "110" =>leds1<= "0000010";--6
          when "111" =>leds1<= "1111000";--7
          when others =>leds1<= "1111000";
       end case;
    end process;
```

As shown in Figure 13 in Modelsim, the value of the counter changes 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 13. Hardware Trojan Simulation in ModelSim.

4.2. System Integration.

The system schematic in Figure 14 illustrates how the hardware components, specifically the FPGA and Raspberry Pi, were connected for seamless communication between them.

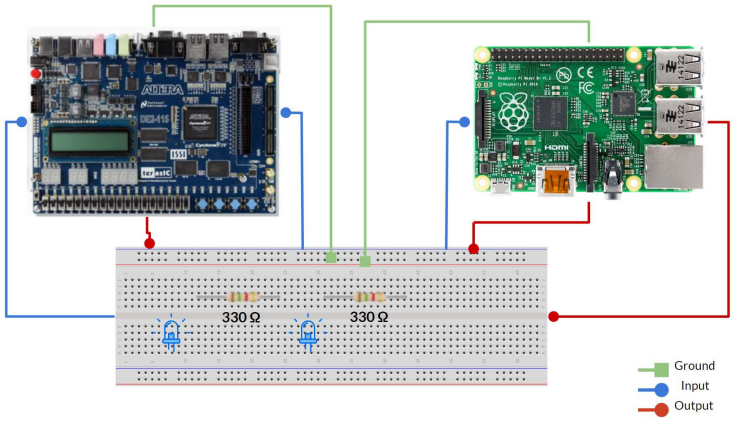


Figure 14. System Hardware Components Schematic.

 doi:10.20944/preprints202407.1498.v1

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:**
  - **FPGA:**
    - Pin 'AB 22' is used for transmitting the result of XORing values A and B.
    - 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.
  - **Raspberry Pi:**
    - Pin 37 is used by the FPGA to alert the honeypot of a malicious packet if a Trojan has been triggered.
- **Outputs:**
  - **FPGA:**
    - Pin 'AC 215' is used to report back to the honeypot that the Trojan is activated.
  - **Raspberry Pi:**
    - Pin 40 is used to send the result of the XOR function to the FPGA.
    - Pin 35 is used to compare the password entered on the website with the correct password generated for the username.
- **Resistors**: As a result of the DE2-115 board I/O standard mismatch, 330 Ω pull-down resistors were used to compensate for the mismatched GPIO pin differences.
- **LEDs:** Used for testing and debugging purposes.

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 15 illustrates the process by which an incoming connection undergoes verification to determine whether it constitutes an attack. 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 is not intended for legitimate access.

 doi:10.20944/preprints202407.1498.v1

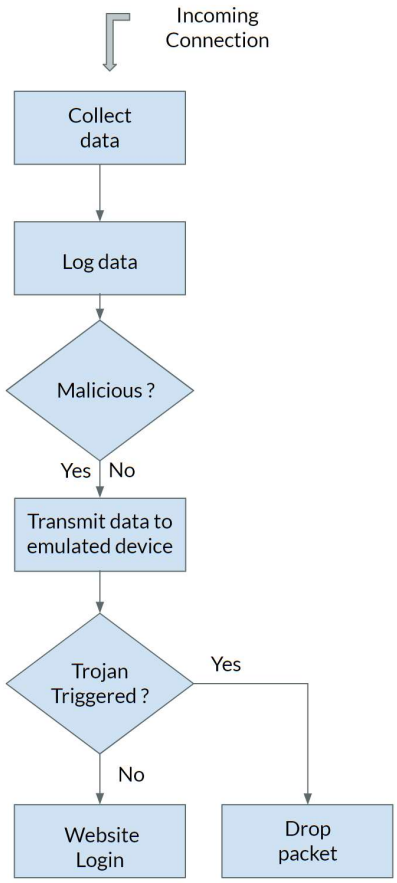


Figure 15. Flowchart of Our Honeypot System's Processes.

When values A and B are entered, their XORing results are transmitted to an emulated FPGA device to monitor its impact on the IoT device's behavior. The emulated device then reports back to the honeypot, indicating whether 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 about the attacker's objectives.

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 the IoT device. The circuit, as illustrated in Figure 16, 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 17 and 18. Figures 19 and 20 show that for input '9', the generated password remains unaffected by the Trojan, demonstrating a selective impact based on specific conditions.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 July 2024 doi:10.20944/preprints202407.1498.v1

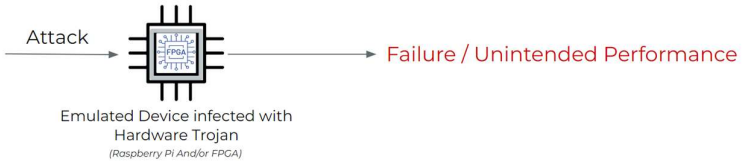


Figure 16. The modified circuit.

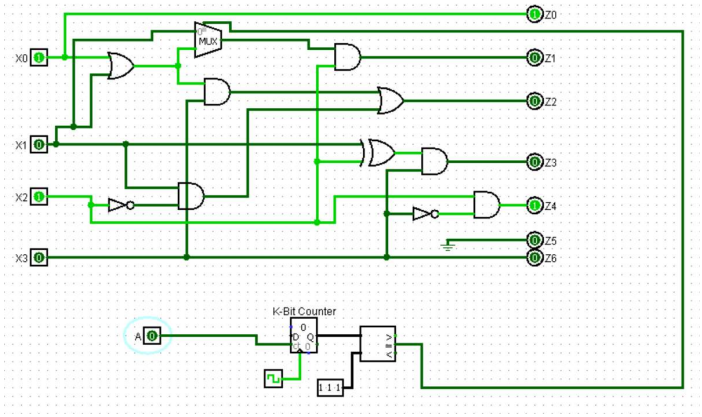


Figure 17. (a) Generated password with trigger off.

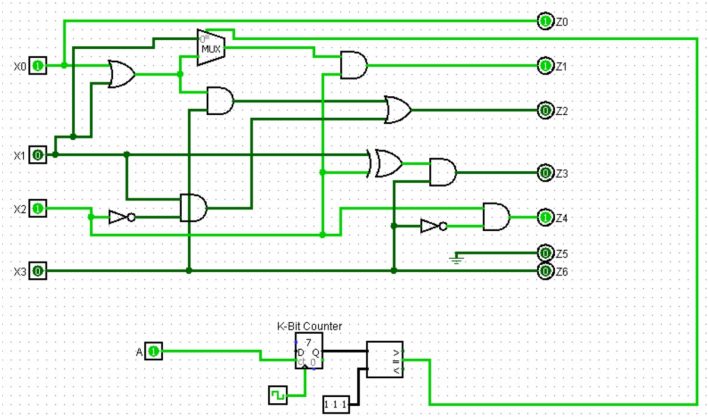


Figure 18. (b) Generated password with trigger on.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 July 2024 doi:10.20944/preprints202407.1498.v1

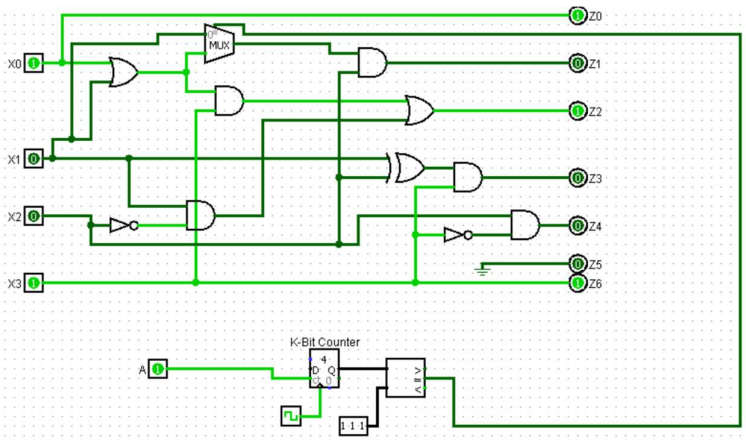


Figure 19. (c) Generated password with trigger off.

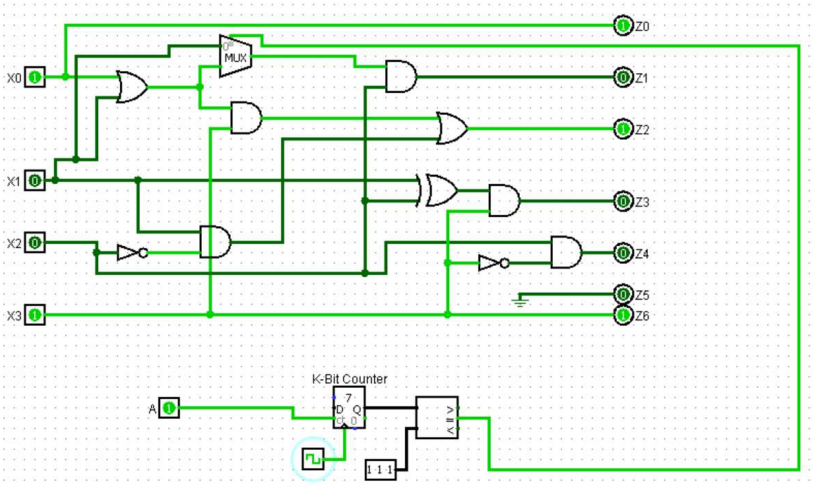


Figure 20. (d) Generated password with trigger on.

5.2. Evasion Testing

Various scenarios were considered to test the proposed system. We categorized the types of attacks into four groups: authorized attacks, unauthorized attacks, triggering attacks, 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 'SmartLock', logs the entered data and prevents intrusion by reporting to the FPGA. This scenario is illustrated in Figure 21.

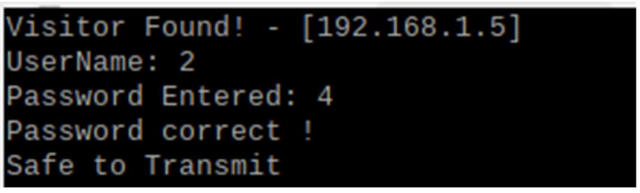


Figure 21. Authorized attacks.

5.2.2. Unauthorized Attacks

 doi:10.20944/preprints202407.1498.v1

Unauthorized attacks involve attempts to log in to 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 22. Figure 23 illustrates the various inputs to the website, highlighting both successful logins with correct credentials and unsuccessful attempts with incorrect ones.

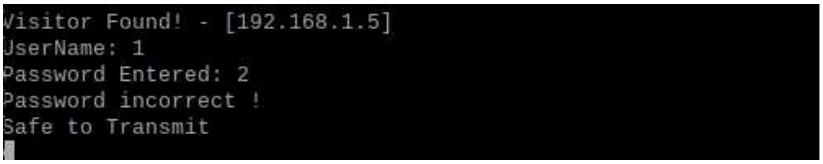
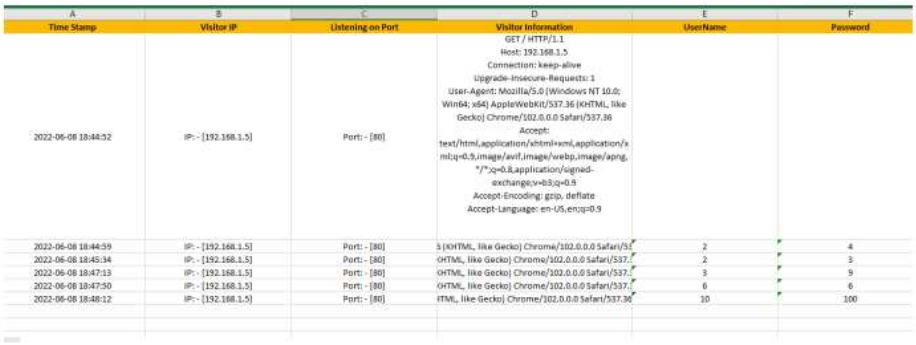


Figure 22. Unauthorized Attack.



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/53	2	4
2022-06-08 18:45:34	IP: - [192.168.1.5]	Port: - [80]	OHTML, like Gecko) Chrome/102.0.0.0 Safari/537.	2	3
2022-06-08 18:47:13	IP: - [192.168.1.5]	Port: - [80]	OHTML, like Gecko) Chrome/102.0.0.0 Safari/537.	3	9
2022-06-08 18:47:50	IP: - [192.168.1.5]	Port: - [80]	OHTML, like Gecko) Chrome/102.0.0.0 Safari/537.	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 23. 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 to be an unauthorized access, and a 'U' will be displayed on the neighboring 7-segment display.

Figure 24 shows the values of A and B entered through the website until the Trojan is activated upon reaching the counter limit of 7. Figure 25 illustrates the process of a packet being dropped when the correct credentials are provided. Figure 26 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.

 doi:10.20944/preprints202407.1498.v1

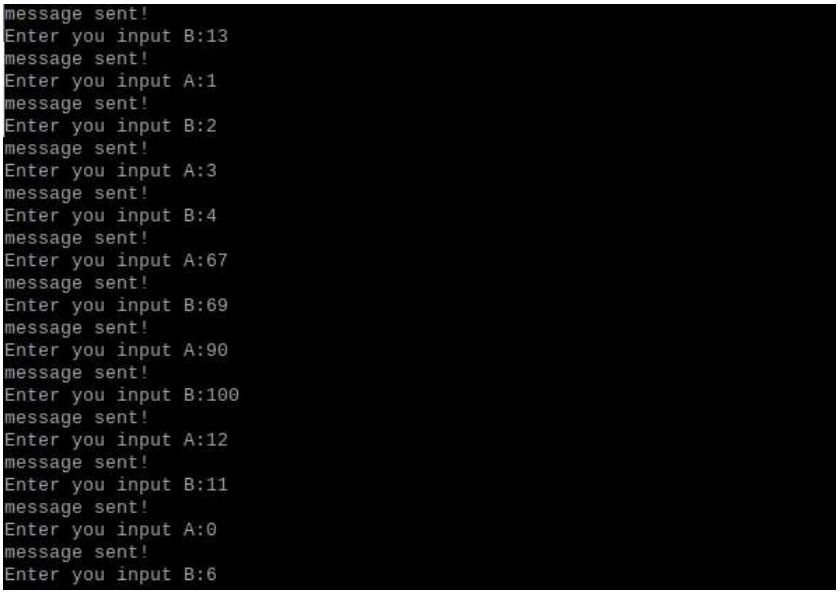


Figure 24. Triggering The HT.

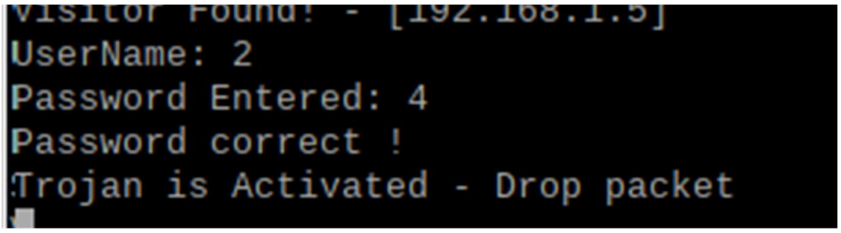


Figure 25. Triggered Trojan with correct credentials.

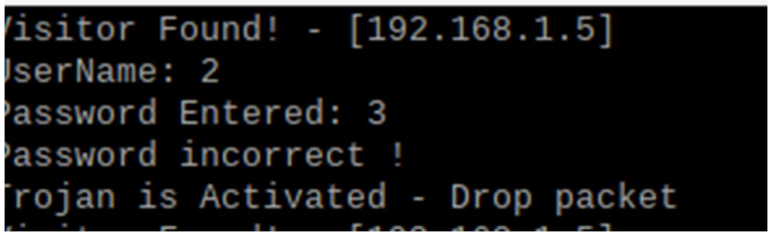


Figure 26. Unauthorized triggered.

5.2.4. Non-Triggering Attacks

Non-triggering attacks involve the values of A and B that do not cause the counter to reach seven. Every entered value is logged and the attack can be either authorized or unauthorized. Figure 27 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.

 doi:10.20944/preprints202407.1498.v1


```
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 27. 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 network traffic, as illustrated in Figure 28.

```
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 100
0
    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 28. Fetching the Correct Interface.

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 29.

```
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, le
ngth 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 29. TCPdump Capturing Incoming Packets.

5.4. Analysis

Testing the proposed system under various scenarios demonstrated its ability to effectively differentiate between authorized and unauthorized access attempts. The honeypot's capability to

 doi:10.20944/preprints202407.1498.v1

mimic the admin login page, log data, and prevent intrusion highlights its effectiveness in handling unauthorized access attempts. In addition, the system's response to unauthorized access attempts by logging and displaying unauthorized entries further validates its strength. The triggering attacks highlight the resilience of the system to activate the HT through repeated inputs. The system successfully logs and prevents unauthorized triggering, thereby ensuring the security of the IoT device. The TCPdump analysis provides additional verification of the system's ability to monitor and capture malicious HTTP POST requests, thereby strengthening the overall security framework.

These results confirm that the proposed system, incorporating honeypots and packet sniffers, adds a significant layer of defense to IoT devices against various types of cyber-attacks. Future work may focus on enhancing the sophistication of honeypots and integrating machine learning algorithms to adaptively respond to evolving attack patterns.

6. Conclusion and Future Work

6.1. Main Contribution

This study aimed to enhance the security of IoT devices by implementing a hybrid hardware honeypot system to detect, prevent, and log cyber-attacks. The system was tested under various scenarios, including both authorized and unauthorized access attempts, to validate its effectiveness. The results and subsequent analysis offer critical insights into the system's performance and suggest potential improvements for future implementation.

6.1.1. Effectiveness of Hardware Honeypots

The utilization of hardware honeypots in this study proved to be a robust method for securing IoT devices. By mimicking an admin login page and logging all data entries, the honeypot successfully diverted and captured the unauthorized access attempts. This approach not only prevents actual intrusions but also provides valuable data for analyzing attack patterns. The implementation of a low-interaction honeypot was effective in this context, balancing the complexity and resource consumption. However, for high-security environments, transitioning to high-interaction honeypots could offer deeper insights into attacker behavior and more sophisticated logging capabilities.

6.1.2. Hardware Trojan Detection and Prevention

The integration of HT detection and prevention into the system introduces an additional security layer. The proof of concept demonstrates that the hybrid Trojan design can selectively impact an IoT device based on specific conditions, such as consecutive inputs leading to a triggered state. The selective nature of the Trojan's activation, triggered by a counter reaching a predetermined value, showcases the system's ability to withstand repeated intrusion attempts without compromising the overall functionality of the IoT device. This approach significantly reduces the risk of unauthorized access, while maintaining the operational integrity of the device.

6.1.3. Analysis of Attack Scenarios

- **Authorized Attacks:** The system's response to authorized attacks is both effective and efficient. Logging every attempt and verifying credentials ensured that only valid access was granted. This verification process also included a fallback mechanism that prevented unauthorized access, even with valid credentials, highlighting the robustness of the security measures in place.
- **Unauthorized Attacks:** In scenarios involving unauthorized access attempts, the honeypot's logging and redirecting capabilities are critical. Immediate feedback and logging of unauthorized entries allowed real-time monitoring and a potential rapid response. This capability is crucial for identifying and mitigating potential threats before vulnerabilities can be exploited.
- **Triggering Attacks:** The results of triggering attacks are particularly noteworthy. The system's ability to activate the HT only when a specific condition was met (e.g., a counter reaching seven) ensured that false positives were minimized. This selective activation mechanism is crucial for

 doi:10.20944/preprints202407.1498.v1

maintaining the reliability and availability of IoT devices that are often subjected to numerous benign inputs that do not trigger security measures.

- **Non-Triggering Attacks:** For non-triggering attacks, the system effectively logged all attempts and maintained normal operation without activating the Trojan. This capability underscores the system's reliability in distinguishing between benign and malicious inputs, further enhancing its overall security.
- **TCPdump and Packet Analysis:** The deployment of TCPdump for capturing incoming HTTP POST requests provides an additional layer of network security monitoring. The successful installation and testing of the TCPdump demonstrated its capability to log and analyze network traffic in real-time. This tool proved invaluable in identifying and documenting malicious attempts to access IoT devices, thereby providing a comprehensive view of the network's security landscape. The logs captured from ports 80 and 44444 offered detailed insights into the nature and frequency of access attempts, informing further security enhancements.

6.2. System Limitations and Future Work

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

- **Scalability:** Although the current system is effective for the tested scenarios, it may face challenges when scaled to 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 more complex attack patterns.
- **Machine Learning Integration:** Incorporating machine learning algorithms can enhance a system's ability to adaptively respond 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 informing 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.

Author Contributions: Conceptualization, A.H. and H.S.; methodology, A.H. and H.S.; software, A.H. and H.S.; validation, A.H., H.S., D.K.M. and A.A.; formal analysis, A.H., H.S., D.K.M. and A.A.; investigation, A.H., H.S. and D.K.M.; resources, A.H., H.S. and D.K.M.; data curation, A.H. and H.S.; writing—original draft preparation, A.H., H.S. and D.K.M.; writing—review and editing, A.H., H.S., D.K.M. and A.A.; visualization, A.H. 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.

References

1. Alyasiri, H.; Clark, J. A.; Malik, A.; Fréin, R. d. Grammatical evolution for detecting cyberattacks in internet of things environments. In Proceedings of the International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 2021, pp. 1-6. doi: 10.1109/ICCCN52240.2021.9522283.

2. Suber, J.G; Zantua, M. Intelligent Interaction Honeypots for Threat Hunting within the Internet of Things. Journal of The Colloquium for Information Systems Security Education. **2022**, 9, 1, 1–5. doi:10.53735/cisse.v9i1.147
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. Cybersecur. **2021**, 4, 18, 1-27. doi:10.1186/s42400-021-00077-7
4. Kumar, A.; Abhishek, K.; Ghalib, M.R.; Shankar, A.; Cheng, X. Intrusion detection and prevention system for an IoT environment. Digital Communications and Networks. **2022**, 8, 4, 540-551. doi:10.1016/j.dcan.2022.05.027.
5. Pa, Y. M. P.; Suzuki, S.; Yoshioka, K.; Matsumoto, T.; Kasama, T.; Rossow, C. IoTPOT: A novel honeypot for revealing current IoT threats. Journal of Information Processing. **2016**, 24, 3, 522-533. doi:10.2197/ipsjjip.24.522
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, 2018, pp. 900-906. doi: 10.1109/AINA.2018.00132.
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, 2018, pp. 1-8. doi: 10.1109/PCCC.2018.8711321.
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, 2022, pp. 7-12. doi: 10.1109/ICICT55905.2022.00009.
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, Egypt, 2021, pp. 1-5. doi: 10.1109/ICCES54031.2021.9686133.
10. Elhusseiny, 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, 2023, pp. 1-6. doi: 10.1109/CPERE56564.2023.10119631.
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, 2023, pp. 490-497. doi: 10.1109/ICEARS56392.2023.10085106.
12. Bryant, B.; Saiedian, H. Key challenges in security of IoT devices and securing them with the blockchain technology. Security and privacy. **2022**, 5, 5. doi: 10.1002/spy2.251.
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, 2022, pp. 450-456. doi: 10.1109/ICETSIS55481.2022.9888930.
14. Khanam, R. Review of Threats in IoT Systems: Challenges and Solutions. International Journal For Science Technology And Engineering. **2023**, 11, 2, 325-341.doi: 10.22214/ijraset.2023.49016
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. doi:10.36227/techrxiv.23512389.v1
16. Gupta, S.; Lingareddy, N. Security Threats and Their Mitigations in IoT Devices. In: Pawar, P.M., Balasubramaniam, R., Ronge, B.P., Salunkhe, S.B., Vibhute, A.S., Melinamath, B. (eds) Techno-Societal 2020. Springer, Cham, 2021. doi:10.1007/978-3-030-69921-5_42
17. Bakshi, G. IoT Architecture Vulnerabilities and Security Measures. In: Bhardwaj, A., Sapra, V. (eds) Security Incidents & Response Against Cyber Attacks. EAI/Springer Innovations in Communication and Computing. Springer, Cham, 2021. doi:10.1007/978-3-030-69174-5_10
18. Hromada, D.; Costa, R. L.; Santos, L.; Rabadão, C. Security Aspects of the Internet of Things. In C. González García & V. García-Díaz (Eds.), IoT Protocols and Applications for Improving Industry, Environment, and Society (pp. 207-233). IGI Global, 2021. doi:10.4018/978-1-7998-6463-9.ch010
19. Mallik, P.; Jena, O.P. Analysis of Security Vulnerabilities of Internet of Things and It's Solutions. In: Udgata, S.K., Sethi, S., Srirama, S.N. (eds) Intelligent Systems. Lecture Notes in Networks and Systems, vol 185. Springer, Singapore, 2021. doi:10.1007/978-981-33-6081-5_35
20. Amit, Dhingra, A.; Sangwan, A.; Sindhu, V. DDOS Attack on IOT devices. Int J Circuit Comput Networking **2022**; 3(1), 33-42. doi:10.33545/27075923.2022.v3.i1a.41
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. doi:10.3390/fi15050187
22. Mohd Bakry, B. B.; Bt Adenan, A. R.; Mohd Yussoff, 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, 2022, pp. 40-45. doi: 10.1109/ICONDA56696.2022.10000370.

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. doi:10.3390/s23135941
24. Kampel, L.; Kitsos, P.; Simos, D. E. Locating Hardware Trojans Using Combinatorial Testing for Cryptographic Circuits. IEEE Access **2022**, 10, 18787-18806. doi: 10.1109/ACCESS.2022.3151378.
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, Korea, 2021, pp. 1-5, doi: 10.1109/ISCAS51556.2021.9401143.
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, 2011, pp. 1007-1009, doi: 10.1109/ICQR2MSE.2011.5976774.
27. Shakya, B.; He, T.; Salmani, H. et al. Benchmarking of Hardware Trojans and Maliciously Affected Circuits. J Hardw Syst Secur **2017**, 1, 85–102. doi:10.1007/s41635-017-0001-6
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. doi:10.1007/s10836-023-06054-x
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, 2023, pp. 1-5, doi: 10.1109/PCEMS58491.2023.10136083.
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. doi:10.3390/electronics11244138
31. Hassan R.; Meng, X.; Basu, K.; Dinakarrao, S. M. P. Circuit Topology-Aware Vaccination-Based Hardware Trojan Detection. Trans. Comp.-Aided Des. Integ. Cir. Sys. 42, 9 (Sept. 2023), 2852–2862. doi:10.1109/TCAD.2023.3234440
32. Yasaei, R.; Chen, L.; Yu, S.-Y.; Faruque, M.A. A. Hardware trojan detection using graph neural networks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **2022**, 1-14. doi: 10.1109/TCAD.2022.3178355.
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, 2024, pp. 47-52. doi: 10.1109/DDECS60919.2024.10508924.
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, 2016, pp. 6-10. doi: 10.1109/ICSSA.2016.8.
35. Piggin, 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, 2016, pp. 1-6, doi: 10.1049/cp.2016.0860.
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. doi:10.2991/ijndc.2013.1.2.5
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). Association for Computing Machinery, New York, NY, USA, 2023, pp.49–59. doi:10.1145/3558482.3590195
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, 2022, pp. 765-771. doi: 10.1109/CICN56167.2022.10008323.
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, 2021, 1-7. doi:10.1007/978-3-030-88428-4
40. Srinivasa, S.; Pedersen; J.M.; Vasilomanolakis, E. 2022. 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). Association for Computing Machinery, New York, NY, USA, 742–755. doi:10.1145/3564625.3564645
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, 2022, pp. 30-34. doi:10.1109/6GIoTT57212.2022.00014.
42. Moein, S.; Gulliver, T. A.; Gebali, F.; Alkandari, A. A New Characterization of Hardware Trojans. IEEE Access **2016**, 4, 2721-2731. doi: 10.1109/ACCESS.2016.2575039.

 doi:10.20944/preprints202407.1498.v1