

# Internet of Things (IoT) Device Security

MSc Research Project  
MSc. Cybersecurity

Daniel Ruane  
Student ID: 22195980

School of Computing  
National College of Ireland

Supervisor: Ross Spelman

**National College of Ireland**  
**MSc Project Submission Sheet**  
**School of Computing**

**Student Name:** Daniel Ruane  
.....  
22195980  
**Student ID:** .....

**Programme:** .....MSc of Cybersecurity..... **Year:** .....2024.....

**Module:** .....Research Project.....

**Supervisor:** .....Ross Spelman.....

**Submission Due Date:** .....12<sup>th</sup> August 2024.....

**Project Title:** ... Internet Of Things Device Security.....

**Word Count:** .....8476..... **Page Count:**.....20.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** .....Daniel Ruane.....

**Date:** .....6<sup>th</sup> August 2024.....

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission,</b> to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project,</b> both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# AI Acknowledgement Supplement

## 1 [Internet Of Things Device Security]

Your Name/Student Number	Course	Date
Daniel Ruane/22195980	MSCCYBE_JANO23_O	6/08/2024

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click [here](#).

## 2 AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
ChatGPT4	ChatGPT4 is a multimodal large language model developed by the team at OpenAI and is the fourth iteration of the GPT foundation models.	<a href="https://chatgpt.com/">https://chatgpt.com/</a>

## 3 Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used.**

ChatGPT4	
Development of Code to simulate a Mirai botnet attack	
Prompt: I would like you to develop a simple IoT login page with a username and password form field. It should be composed of a css file, a php file, a html file and a db file to save and access username and passwords. The db file should be able to be imported into myphpadmin . It should be based on the LAMP stack and use the colours blue, light blue , white and red.	Response from ChatGPT: Certainly! Below is a simple implementation of an IoT login page using the LAMP stack. The solution includes the necessary HTML, CSS, PHP, and SQL files.

## 4 Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

## 5 Additional Evidence: A

Prompt:

I would like to add an input timer on both fields on the login screen. If the fields are completed quicker than 500ms, do not submit the form. I would like to also add a button on the form page that simulates a bot completing the form in under 500ms and showing that the entry was not submitted in red text

Response:

To achieve this, we need to modify the HTML and JavaScript of the login form. We'll add a timer to track the input times and prevent form submission if the fields are completed too quickly. Additionally, we'll add a button that simulates a bot filling out the form in under 500ms, and display a message in red text if the entry is not submitted.

## 6 Additional Evidence: B

Prompt

I would like to also add honeypot fields to the IoT device login form. Please add two honeypot fields that are hidden to the human eye requesting a name and telephone number. If these fields are completed we know it was by a bot so reject the form submission

Response

To add honeypot fields to your IoT device login form and modify the form submission logic to reject submissions if these fields are filled, you can follow these steps:

1. **Modify the HTML to include hidden honeypot fields.**
2. **Update the PHP script to check for honeypot field values and reject submissions accordingly.**

# Internet of Things (IoT) Device Security

Daniel Ruane  
S22195980

## Abstract

One of the largest digital device markets in the world will reach a total of 41 billion devices this year. The Internet of Things (IoT) market is one of the largest digital device markets in the world. There is, however, a downside to all of these IoT devices that sit unsecured, in our homes, our businesses and our hospitals. These devices, that are developed and sold relatively cheaply, can be like a security timebomb, waiting to go off. The Secure Software Development Lifecycle phases (S-SDLC) that many new digital devices go through, do not seem to apply to IoT devices. This puts many IoT devices in the firing line when it comes to botnet malware. This malware is used to compromise IoT devices to steal our data and use them to perform attacks on other devices. A well-known botnet malware called Mirai, specifically infects IoT devices and uses them to become a botnet army of slave devices to perform Distributed Denial of Service Attacks (DDoS). In this research project, it is my aim to provide security guidance to IoT device manufacturers and home users alike on how to secure their IoT devices. The focus of this research project will aim to lower the number of devices that can be compromised by botnet malware blocking the theft of users' data. I also aim to reduce the number of IoT devices joining an IoT botnet, therefore reducing botnet propagation.

**Keywords:** Secure IoT Devices, Mirai Botnet, Internet of Things Security, IoT Security, Botnets, IoT Device Security, IoT Device Hardening, Bot Security, Honeypots, hCaptcha security.

## 7 Introduction

A lot of Internet of Things (IoT) devices developed today are built inexpensively by manufacturers. There is very little consideration by manufacturers in relation to the security of the devices software or the hardware that enables the device to function. Millions of IoT devices are sold each month, with everything from IoT smart sockets to smart lightbulbs to IP cameras to smart doorbells entering our home and business network environments. It's estimated that (S. Al-Sarawi, 2020) 41 billion IoT connected devices are to be up and running by the end of this year. Considering the poor security these devices inherently come packed with, many of these IoT devices can become easily hacked, allowing a threat actor to gain a foothold on to your home network, albeit invisibly.

Many of these IoT devices can also become infected with botnet malware and join up as slaves to a botnet army which can then be used to take down massive global digital media companies in a heartbeat.

Many of these new IoT devices that enter our homes and business's come preconfigured with default passwords and usernames (Kelly, 2020) out of the box, and its these settings which can make them susceptible to botnet malware. This can allow hackers to enter the device remotely and monitor information on your network. It can also allow botnet malware to compromise your device and join it to an army of slave IoT devices, called a botnet. These botnets, can then be rented out for hire to criminals on the Dark Web and used in an attack called a Distributed Denial of Service attack or DDoSaaS attack on their victims. When a website or an online application is attacked by a large enough distributed denial of service (DDOS) attack, it can pull the website down with too many server requests, therefore removing the "availability" of the service to its users and impacting the brand reputation of the supplier.

This research question is certainly worth researching as it reviews a large change in our history in relation to connectivity and the side effects that come with this extra connectivity. In this case, hackers can intrude on to our home and business networks, due to the lack of security on these IoT devices. Therefore, the question of "right to privacy" must also be examined and who is really at fault here? Many of the devices that are developed today and sold to consumers are lacking in security measures and criminals are taking huge advantage of this to aquire data and devices for them to use in botnets. It must then be stressed, that a good "security by design" is built into these devices to ensure that customers privacy is protected. Many of these IoT manufacturers need to ensure that good secure software development lifecycle (S-SDLC) methodologies are built in to any new IoT device design project from inception. Security must become an important aspect of the complete design life cycle and manufacturers need to shift left when applying security to their design lifecycle. It's also very important for governments to come together to develop and oversee security policies and laws that govern the security of these devices and require manufacturers by law to be security compliant, to sell their products in our market. An example of this is the CE mark for products sold in Europe. This could also have a digital product security section which requires the product vendors to put the device through rigorous security testing and have a well-designed security program for each device in place. Without this certification, the devices cannot be sold in our market.

A few years ago (Shah, 2019) in Las Vegas, a casino was hacked by a threat actor looking to steal data on some of the casinos most wealthy clients. The hackers were trying to steal the entire "High Roller" database belonging to the casino. The most relevant part of this story to the research project is that the hack was carried out via an IoT temperature monitor for a massive fish tank located on the main casino floor. The temperature sensor from the fish tank had connected to the main SQL Server and had started to download the SQL DB with all the casinos data and was in the process of uploading it to a command and control (C&C) server. The hackers were caught in the reconnaissance phase of their attack and the anomaly was recognised by the casinos AI threat detection platform, DarkTrace, which automatically blocked the upload.

**Research Question.** How to stop botnet propagation and increase IoT device security?

## 8 Related Work

### **Internet of Things Device Security – A Review.**

For 2024, the IoT market will grow to be worth (Research, 2024) \$500 Billion globally and this figure is expected to grow to over \$3,000 Billion by 2033. In their Landscape of IoT Security paper, Eryk Schiller, Andy Aidoo, Jara Fuhrer, Jonathan Stahl, Michael Zi.rjen, Burkhard Stiller from the University of Zurich remind us of the small amount of system resources that Internet of Things devices intrinsically operate with. There are very little device resources available to use in relation to CPU, RAM and networking. When we have such a small amount of system resources available, it prevents us from using complicated encryption algorithms which require a large compute overhead to solve complex mathematical cyphers. The question must then be asked, if this then infers that IoT devices are not able to be secured correctly because of their physical hardware limitations? The answer is part Yes and part No. The first thing that needs to be done is to build new models and protocols that can work with such tiny resources. By using low compute and low memory intensive cyphers like Simon and Speck, it is very possible to secure the communication channels of IoT devices. When it comes to the manufacturers concern and their questions like “will this new security library interfere with the devices original design purpose and slow it down?”, arise, the IoT device will continue to operate as designed, with the added benefit of secure communications by using these lightweight encryption libraries. While these lightweight encryption libraries secure the devices data in transit and at rest, it does not make the device secure against malware attack from botnet malware like Mirai. It’s therefore very important that these devices are also secured against malware compromise. If the IoT device is left without security measures that use a “defence in depth” methodology, it can be compromised quickly and form part of a botnet, causing major disruption to other services, devices and systems.

### **IoT Device Data Encryption.**

(Sunil Kumar, 2024) One of the main reasons why business’s start an IoT project is to save money. 54% of all (Laborde, 2023) business’s that commenced a companywide IoT project said that “cost saving measures” were the biggest driver for the project. Saving money is also very important when selecting the IoT device that will be used to deliver these money saving projects. Manufactures of IoT devices understand this and keep device manufacture costs very low to remain competitive. Devices are therefore very limited when it comes to resources and have just enough CPU, Memory, battery size and networking capability to complete their designed task, and no more. In a Review of Lightweight Security and Privacy for Resource Constrained IoT Devices, there are several lightweight block cyphers identified to encrypt IoT data at rest and in transit. The most recent technology is a lightweight cypher from the NSA that was developed in 2013. The major drawback with block cyphers is that over time, with the increasing availability of more computing power, the higher the probability that the cypher will become broken and insecure. Meaning that cyphers from back in 2013 are no longer as secure as they were when they were released, back 11 years ago. The problem then arises that there are very few lightweight secure cyphers to encrypt IoT devices data at rest and data in transit.

### **Security Analysis of Internet of Things Devices utilising Mobile Computing.**

(Bin Liao, 2020) Internet of Things devices are available in many verticals including smart cities, healthcare, transport and smart homes to name but a few. The devices are processing very important data which could be patient health information of an extremely private nature. The data must therefore be protected with robust security measures, ensuring that the devices confidentiality, integrity and availability are not compromised. In this paper, the security surrounding the protection of the device data and the device itself, is reviewed. The main idea is to use mobile devices like iPhones or Android devices to secure the IoT device. This (Liao, 2020) synergy between an IoT device and a mobile device would compensate for the low compute overhead that comes with an IoT device and hands off this element of security to the mobile device in relation to encryption, authentication and authorisation related to users data. The iPhone or Android device would be able to connect to the IoT device with strong authentication methods including biometric via an application over Bluetooth Low Energy (BLE) connection. When configuring the IoT device, a secure mobile application running on the iPhone or Android device can be used to access the IoT devices administration and configuration area. This would be the only way of communicating with the device's configuration area. This would be an excellent method to secure the IoT device, it blocks many open ports and communication protocols that are intrinsically open on IoT devices, therefore reducing the overall attack surface. The only problem with this security application method is the iPhone or Android device that is securing the communication channel may also become compromised. Its therefore very important to ensure that mobile device is kept patched and updated with the latest software releases and set to automatically update to receive the latest security updates, including zero-day attack security updates.

### **Vulnerability studies and security postures of IoT devices: A smart home case study.**

(Mason et al, 2020) Smart Home, Internet of Thing devices are (Davis, 2020) widely available to consumers in the form of smart bulbs, smart sockets, smart wearables, smart fridges and washing machines, smart cameras to smart locks, smart thermostats, smart speakers, smart TV's etc. All these devices can be compromised, however, this paper aims to prove that better, well-known brands do a better job of securing their IoT devices than lesser well-known brands. The lesser well-known brands business model works on the idea of cheaper is better with a lower consumer price point. The lesser-known brands then end up outbidding the better-known brands for the customer's hard-earned cash. Because of this lower price point, there is an influx of these unsecured devices on our home and business networks. The lesser-known brands main aim is to get the device to do just what is required and what it says on the box, and to do this as cheaply as possible. This cheap product that we use on our home networks are generally full of security vulnerabilities. Many of these vulnerabilities are found by security researchers and posted to CVE databases. Many of these companies are contacted about these security flaws, but most do not respond and continue to make a poorly secured, cheap devices.



## **Understanding The Mirai Botnet**

The papers that have been previously discussed above, revolve around the encryption of data from IoT devices. Of course, this is a significant security measure when processing and storing critical data. However, the largest global threat from IoT devices in 2024, is botnets. If we step back to 2016, over eight years ago, the world's largest botnet infected over 600,000 devices. In the first minute of existence the (Antonakakis, 2017) Mirai Botnet infected 834 IoT devices, in the first 10 minutes, this surged to 11,000 devices, in the first 20 hours, 64,500 devices became compromised. The Mirai malware doubling time was 75 minutes and this led to one of the largest DDOS attacks on record at 600 Gigabits Per Second (Gbps) In 2016, there were 6.4 billion Internet of Things devices, globally. Today we have at least seven times that number of IoT devices, however, the security posture with these cheap IoT devices, is still the same, extremely poor. Because the world relies so heavily on many online services and tools to do our daily work, it's extremely important to secure these devices against malware including botnet worms. Without security, these devices will be used to bring about utter anarchy. Used by criminal gangs that are transitioning from physical to digital theft, in a new era of crime.

## **9 Research Methodology**

The answer to my research question will be in the form of experimental testing to ensure the confidentiality, integrity and availability (CIA Triad) of an IoT device can be secured easily by manufacturers against a botnet malware attack, simulated on Mirai malware heuristics. I intend to use pre-existing, current, research data to show that the security methodologies applied to the device will improve the IoT devices security posture and the overall data security of the device.

In my research, I have come across several software and web-based tools that will help me to research my question. Publish or Perish is a software application that I currently run on my Mac. It allows me to find papers via search terms and search keywords which also comes with many configuration settings. I also use a web-based, AI application called Research Rabbit. This tool is excellent for finding similar papers with similar characteristics to my research question. I also use the NCI library to see past theses papers and its helpful to understand previous student approaches to research questions and it also helps as all the papers there are peer reviewed. The last tool I like to use is Connected Papers, this is like Research Rabbit and allows me search for papers connected to my question.

## **10 Design Specification with \*AI Assistance\***

The design specification and solution development would need to circle around a weakness or security flaw found in IoT devices. The design flaw that allows many IoT devices to be compromised by botnet malware is directly related to passwords. The problem with many IoT devices is that they are shipped with a default username and password.

These default usernames and passwords come in many versions, but usually take the form of admin for the username and 12345 or 00000 or password123 for the default password. (Colon, 2016) Many IoT devices are left running in production with the default password and username still in place and this is the security flaw that will be addressed with an ICT security solution.

Many IoT devices have a log in page that allows the user to get to the admin dashboard, where they can change configuration settings. It's at this page where botnets can brute force access to the device and change other internal settings to allow further compromise of the device and finally, joining the botnet as a botnet slave device.

The Mirai botnet used a combination of 61 default passwords to gain access to over 650,000 IoT devices via brute force attack on similar pages like this. Therefore, to replicate this brute force attack and put in place measures to significantly counter this attack, it will require the development of a simulation IoT login page running on a LAMP stack, Python code development to simulate a bot attack and form security counter measures to prove we can reduce the risk of IoT device compromise. I have used AI in the past to develop some small code snippets and intend to use it in this research project to assist in developing the code for the IoT login page and the Mirai Bot, brute force simulation code, written in Python. The AI model used to write and much of the code is ChatGPT4.

ChatGPT4 or Generative Pre-Trained Transformer 4 is a multimodal large language model developed by the team at OpenAI and is the fourth iteration of the GPT foundation models. The model was trained using a combination of first supervised learning on a large dataset and then reinforcement training using human and AI feedback methods. The finer details of the training such as model size, architecture or hardware used was not reported by the team at OpenAI. Most likely to keep the technology private as its currently such a competitive industry. Knowing exactly what was required to setup the IoT device login page and being very familiar with the LAMP stack, I used ChatGPT4 to create the basic files and iterate upon these until the simulation was a success. There was a large amount of input from my side, and it required some significant debugging, but overall the use of ChatGPT4 to develop much of the code was very helpful and aided in development of the simulation. I used Brackets version 2.2 to modify the code produced by ChatGPT4.

I prompted ChatGPT4 to develop an IoT device login page that was composed of a CSS file, which would style the IoT device login page and the elements such as buttons and borders, two php files, one to handle the login logic and the other to show a success pages after successful login, a html file which was the index/home page that displayed all of the page elements and an SQL file to import into MAMPS PhPMyAdmin to generate the database and insert the password and username that the botnet simulation would try to brute force. The next step was to build the Lamp stack or in my case as I'm a Mac user with a Macbook pro running a 1.4 GHz Quad Core Intel i5 with 8GB ram and Ventura 13.3.1(a), I used MacOS, Apache, MySQL and PHP otherwise known as MAMP. I used MAMP version 6.0 as the technology stack to serve the IoT login page. The database server was MySQL 5.7.39. The web server was Apache 2.4.54 and the PHP version 7.4.33. Finally, phpMyAdmin version was 5.2.0.

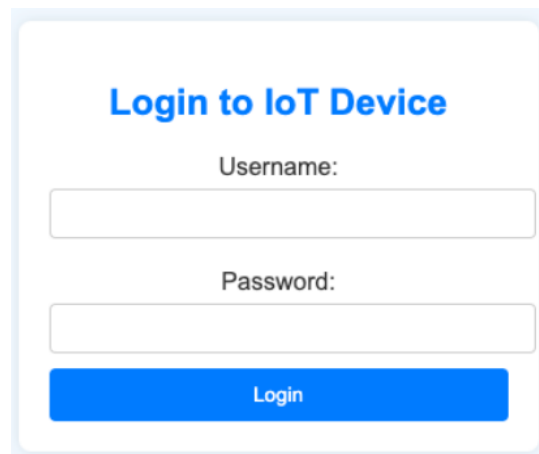
The next part of the architecture design would need to use a tool to simulate multiple bots attacking simultaneously. For this, the tool would need to use multithreading and be able to automate the completion of user input fields with the password and username to be used in the brute force attack simulation. Selenium WebDriver (See Fig 1.0) and Chrome Web driver was chosen as the architecture to simulate the Mira Bot attack.

**FIG 1.0 Selenium Webserver running on MacOS**

```
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050
Dans-MacbookPro:htdocs druanes$ selenium-server standalone --port 4444
16:01:24.607 INFO [LoggingOptions.configureLogEncoding] - Using the system default encoding
16:01:24.614 INFO [OpenTelemetryTracer.createTracer] - Using OpenTelemetry for tracing
16:01:25.485 INFO [NodeOptions.getSessionFactories] - Detected 8 available processors
16:01:25.487 INFO [NodeOptions.discoverDrivers] - Looking for existing drivers on the PATH.
16:01:25.487 INFO [NodeOptions.discoverDrivers] - Add '--selenium-manager true' to the startup command to setup drivers automatically.
16:01:26.270 WARN [SeleniumManager.lambda$runCommand$1] - Unable to discover proper msedgedriver version in offline mode
16:01:26.413 WARN [SeleniumManager.lambda$runCommand$1] - Unable to download safaridriver in offline mode
16:01:26.432 INFO [NodeOptions.report] - Adding Safari for {"browserName": "safari", "platformName": "mac"} 1 times
16:01:26.433 INFO [NodeOptions.report] - Adding Safari Technology Preview for {"browserName": "Safari Technology Preview", "platformName": "mac"} 1 times
16:01:26.435 INFO [NodeOptions.report] - Adding Chrome for {"browserName": "chrome", "platformName": "mac"} 8 times
16:01:26.435 INFO [NodeOptions.report] - Adding Edge for {"browserName": "MicrosoftEdge", "platformName": "mac"} 8 times
16:01:26.436 INFO [NodeOptions.report] - Adding Firefox for {"browserName": "firefox", "platformName": "mac"} 8 times
16:01:26.495 INFO [Node.<init>] - Binding additional locator mechanisms: relative
16:01:26.515 INFO [GridModel.setAvailability] - Switching Node 884489c9-3056-4b70-ac84-2f6880e8970c (uri: http://192.168.178.225:4444) from DOWN to UP
16:01:26.515 INFO [LocalDistributor.add] - Added node 884489c9-3056-4b70-ac84-2f6880e8970c at http://192.168.178.225:4444. Health check every 120s
16:01:26.667 INFO [Standalone.execute] - Started Selenium Standalone 4.23.0 (revision 4df0a231af): http://192.168.178.225:4444
```

Selenium is a set of (Deshpande, 2023) open-source tools developed to automate testing on web applications across multiple browsers. It was developed by Jason Higgins from ThoughtWorks in Chicago. He found the task of manual web application testing very laboursome and developed a JavaScript program to automate web application testing and called it JavaScriptTestRunner. This was the beginning of Selenium back in 2004. Today it has a large userbase and well documented code libraries that can be developed in Python. This tool was selected to automate our Mirai Botnet attack on the IoT Device login page on a Chrome browser. (See FIG 2.0)

**FIG 2.0 IoT Device Login Page Running on MAMP**



I prompted ChatGPT4 to develop a Python script to use the Selenium libraries in the development of the Mirai Botnet simulation script. The version of Python in use is 3.12.4. Originally, I had wanted to use Python Requests, but this is now deprecated for installation via Homebrew as it doesn't follow their CORE models anymore. Homebrew or "brew" is a package installation tool used on MacOS via CLI and it can simplify extremely complex installation routines, the version in use is Homebrew 4.3.12. I used homebrew to install Selenium Webserver in standalone mode.

In my final Mirai Botnet simulation script, I prompted ChatGPT4 to include a progress bar on the MacOS CLI via TQDM. TQDM (Shaditya, 2022) libraries are used to create a python-based progress bar on the CLI to see the progress of an underway task, up to that point it was difficult to see the progress of the simulation botnet attack and I wanted to ensure that the user could see what was happening and be aware that the simulation was running correctly. I also prompted ChatGPT4 to develop a method using Pillow to save a screenshot of the login success page and assign it a name of the bot that was successful in the brute forcing of the login page. This page is shown after a successful login by brute force attack on the IoT Device login page. Pillow is a set of Python libraries that can be used to automate image processing. I installed both libraries on my MAC with "pip". Pip is a package management tool used for installing and managing Python libraries. The packages managed by pip are stored in the Python Package Index repo or the PyPI. It is another handy tool like homebrew but just for Python packages.

I also prompted ChatGPT4 to use a subset of the RockYou text file as a password generator for the bots to use when brute forcing the IoT login page. The first 1000 lines of the rockyou.txt file were used as the file is over 125mb in size. It was very important to try and replicate the botnet brute force attack as closely as possible and size constraints are of huge importance when dealing with IoT devices that have small resources such as memory and compute power. The original Mirai botnet used 61 default passwords, but since then, things have moved on and botnet malware has become even more effective, hence the use of the rockyou.txt subset of passwords to brute force the device.

The next technology used was JavaScript. This was used to create an EventListener on the input fields, username and password. This EventListener monitored the speed at which the input fields were populated, and the form was submitted. I setup a time threshold of 500ms to identify the Mirai bots trying to complete the form. A human would take perhaps 30 seconds to complete the form whereas a bot would complete it in under 500ms. Therefore, all submissions under 500ms are declined.

Associated with these security measures were form field honeypots. These are an invisible user input field that look like a regular user input field to bots as they review the code, but to humans, the field does not display on the user interface. Therefore, if the input box has data in it when the form is submitted, we know the input has been generated by a bot and we decline the form submission. The final piece of technology used was hCaptcha. This form of (Team, 2024) Captcha uses machine learning to identify bots and their action on a page is recorded and reviewed against other bot threat signatures. It also uses AI to generate the “are you a human” “Turin test” type questions and their associated graphics. It is by far, one of the best Captcha platforms available and is in use by many large entities including the company Shopify. A free account was used for this SaaS platform with the latest API deployed in the simulation.

## 11 Implementation

The implementation of the simulation which aimed to prove that the applied security measures could impact the propagation of a botnet was done in three stages. The first stage measured the security impact of having no security features applied to the IoT Device Login page. The second stage reviewed the impact of a single security measure in the form of a JavaScript input timer that stopped the form submission if the form was completed in under 500ms and the third and final stage used three security measures to block the attempted brute force of the IoT Device login page. The security measures used in the third stage were a JavaScript input submission timer, a set of honeypot fields invisible to the human eye and hCaptcha, a Completely Automated Public Turing Test to tell Computers and Humans Apart or Captcha platform that uses AI and ML to identify bot targets.

### Stage One

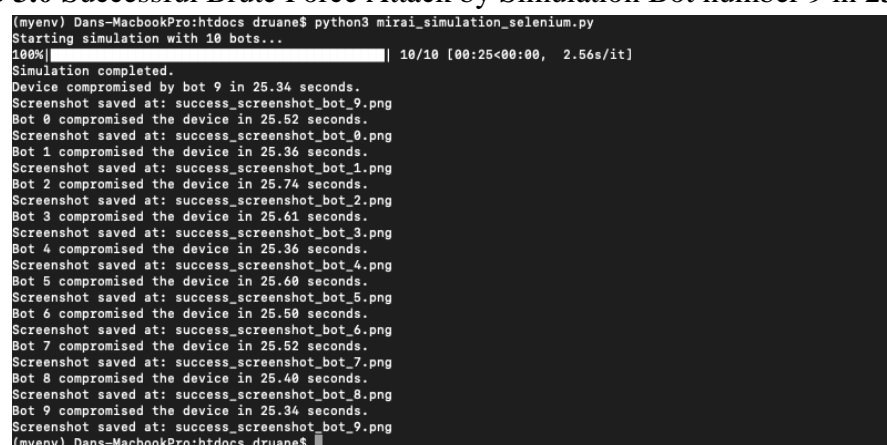
The first stage of the simulation began with the development of the web files that would simulate an IoT device login page. This page normally brings the user to the admin dashboard of the IoT device where they can make changes to the device’s configuration settings. The simulated login page for the first stage just needed a username and password login box along with a success page that followed a successful login. There were no security elements present in this login page and it was used to show a baseline of zero security measures applied. To develop these classes, ChatGPT4 was prompted to create the following files. An index.html page to display the IoT Login page on the Apache Webserver. A CSS

page to handle the styling of the index.html page elements with the name style.css. A login.php file to handle the login logic and connection to the MySQL DB to retrieve the password and username details. A success.php file to display a success page after a successful login attempt. An SQL file with the settings to setup a new MySQL DB on import named `iot_db.sql`.

The next file developed was a Python script using selenium libraries to simulate a Mirai Botnet attack with 10 bots. This Python script used several Selenium libraries which included the Chrome Webdriver to manage the Chrome browser natively and ThreadPoolExecutor to manage the multithreading used to attack the IoT login page by 10 bots, simultaneously. It also used the Python tqdm library to create a progress bar on the MacOS CLI and the PIL (Pillow) library to take a screenshot of the success page upon successful login. The final file was the RockYou.txt file downloaded from the RockYou GitHub site. This was made into a smaller file called `rockyou_subset.txt` with over 1000 lines of passwords.

Once the MAMP stack was up and running and all files were placed in the htdocs folder in MAMP and the IoT login page was available at localhost on `http://127.0.0.1:8888/index.html`, the process to enable the Mirai attack simulation could begin. The MacOS terminal was opened and the command to start the selenium webserver was used, followed by the commands to setup and initiate tqdm and pillow. The next command ran the Selenium Python script, and the simulation began. The simulation worked by opening 10 Chrome browsers at the same time, each browser was directed to the IoT device login page at `http://127.0.0.1:8888/index.html`. The simulation then inputted the usernames contained in the Python script, these were “admin”, “user”, “iot” and “test” and used the password list of 1000 passwords in a random way using the `rockyou_subset.txt` file. The simulation continued and eventually stopped when the IoT device login form had been successfully brute forced by the Mirai Simulation script (See Fig 3.0)

**FIG 3.0** Successful Brute Force Attack by Simulation Bot number 9 in 25.34s

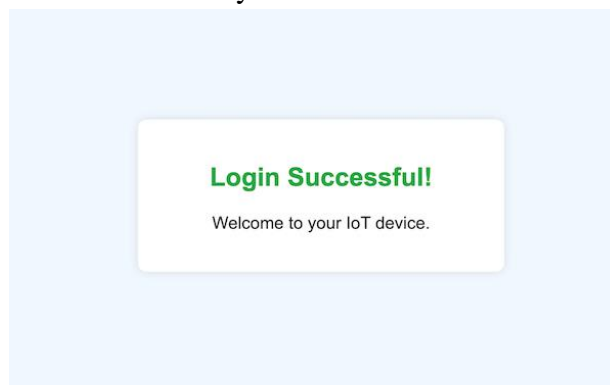


```
(myenv) Dans-MacbookPro:htdocs druanes$ python3 mirai_simulation_selenium.py
Starting simulation with 10 bots...
100%|████████████████████████████████████████████████████████████████████████████████| 10/10 [00:25<00:00, 2.56s/it]
Simulation completed.
Device compromised by bot 9 in 25.34 seconds.
Screenshot saved at: success_screenshot_bot_9.png
Bot 0 compromised the device in 25.52 seconds.
Screenshot saved at: success_screenshot_bot_0.png
Bot 1 compromised the device in 25.36 seconds.
Screenshot saved at: success_screenshot_bot_1.png
Bot 2 compromised the device in 25.74 seconds.
Screenshot saved at: success_screenshot_bot_2.png
Bot 3 compromised the device in 25.61 seconds.
Screenshot saved at: success_screenshot_bot_3.png
Bot 4 compromised the device in 25.36 seconds.
Screenshot saved at: success_screenshot_bot_4.png
Bot 5 compromised the device in 25.60 seconds.
Screenshot saved at: success_screenshot_bot_5.png
Bot 6 compromised the device in 25.50 seconds.
Screenshot saved at: success_screenshot_bot_6.png
Bot 7 compromised the device in 25.52 seconds.
Screenshot saved at: success_screenshot_bot_7.png
Bot 8 compromised the device in 25.40 seconds.
Screenshot saved at: success_screenshot_bot_8.png
Bot 9 compromised the device in 25.34 seconds.
Screenshot saved at: success_screenshot_bot_9.png
(myenv) Dans-MacbookPro:htdocs druanes$
```

The 10 bots all managed to brute force the login in under 30 seconds and the first bot to brute force the login was bot number 9. Each bot saved evidence that it successfully brute forced the login page by taking a screenshot of the login success page. (See Fig 4.0) These results would now act as the baseline and show how susceptible the IoT login page is without any

major security measures. The next stage is to implement the first security measure. The JavaScript timed input submission.

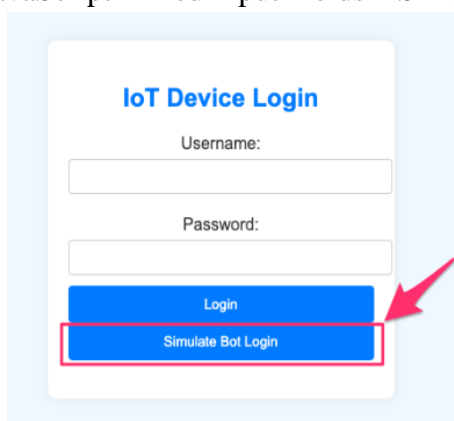
**FIG 4.0** Screenshot taken by bot after successful brute force attack



## Stage Two

The next stage in the testing was stage two. This is the stage where we would implement a security measure and test how successful that security measure was. We now had a baseline of 25.34s for a bot to brute force the IoT login page without any security measures. It was now time to test our theory. For this we needed to change the index.html file to add several JavaScript EventListeners. These monitored the completion of the DomContentLoaded and the user input fields onFocus and onBlur for “username” and “password”. If the fields were completed in under 500ms and the form was submitted, an error was shown “form submitted too quickly” and the form submission failed. The next part of the code change consisted of adding a button on the form to test the bot blocking feature and ensure it worked. This updated JavaScript added another EventListener and created a button that automatically populated the fields username and password with the correct username and password and submitted the form in 100ms. This would also be used later in stage two (See FIG 5.0).

**FIG 5.0** JavaScript Timed Input Fields – Simulate button



Most humans will complete the form in 30 seconds or so, a bot from the Mirai botnet would complete the user inputs fields in less than 500ms. This should block many attempts and the new code was tested by clicking on the “Simulate Bot Login”, it returned the form was submitted too quickly and proved that the timed input fields were working as designed. The

next step was to attack it with the Mirai Botnet simulation Python script. The steps in stage 1 were repeated and the `mirai_simulation_selenium.py` file was started from the terminal window and stage two simulation began on the IoT device login page. The output after 125.99s of testing was “simulation completed – device not compromised after 125.99s with 10 bots attacking” (See Fig 6.0)

**FIG 6.0** Simulation two completed – Device not compromised

```
Starting simulation with 10 bots...
100%|██████████████████████████████████████████████████████████████████████████████| 10/10 [02:05<00:00, 12.58s/it]

Simulation completed.
Device not compromised after 125.99 seconds with 10 bots attacking.
(myenv) Dans-MacbookPro:htdocs druanes$
```

This is a positive result and shows that the security measure implemented can stand up to the simulated Mirai Botnet attack. However, experience has taught me, that this one measure alone will not be enough to stop the botnet from acquiring the simulation device. A layered security approach will be required to significantly stop the propagation of an IoT botnet. Just as this simulation has used AI to test and build a simulation botnet, bad actors are using the power of AI to develop botnet malware. This is where the Good AI vs Bad AI battle will take place and those on the winning side, will win big. In the past, if a bot was not performing and getting the job done, it would need to be updated or patched to deal with the new set of security measures and how to circumvent them, this ever-changing set of new measures and updates to circumvent them is the battle that I speak of and with the power of AI, bad actors can do this update to malware binaries in a few seconds. Therefore, the 500ms timer will not work for very long time and the Mirai malware developers will use a way to overcome or circumvent this security measure and instead of having their Mirai Botnet try to brute force the username and password form in a couple of milliseconds, they will have it submit the form in a second or so, thus defeating the single security measure. When they see success from this tweak to their brute force code with a percentage of their bots, they will know that its successful. It's important then to add several technologies that work together in a defence in depth strategy to block the attack. The security features added in stage three address this and use a layered security posture on the IoT Device login page.

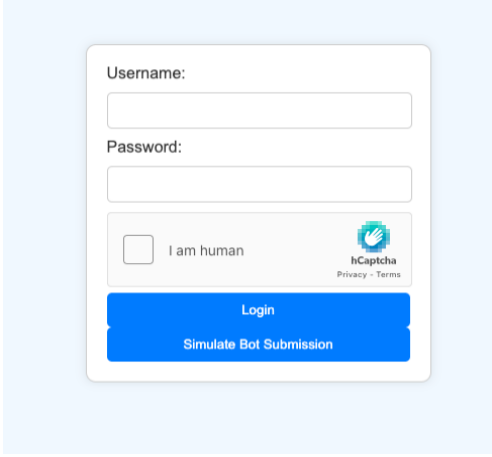
## Stage Three

In stage three, the security principle of “Defence in Depth” was applied to the IoT device login form. The requirement for this was to ensure that a robust and technologically advanced layered defence was in place to ensure the confidentiality, integrity and availability of the IoT device. This would need to stand up against a sustained botnet attack of many thousands of devices as would happen in the real world. To do this, two other security measures were added to the IoT login form. The first was the addition of (Edison, 2019) “Honeypot” fields. These are fields that are hidden to the human eye via CSS on the field `{display:none}`. The bot reads the code on the page of the input form and is led to believe that these fields are real and does not take notice of the CSS file. It therefore fills out these invisible or honeypot fields as it wants to ensure that all fields are completed, therefore the form doesn’t fail submission because required fields have not been completed. When the honeypot fields are completed and filled out by the bot, the form submission should fail and therefore block the

bot from access to the success/admin area of the IoT device. To do this ChatGPT4 was prompted to add two invisible fields to the form called name and telephone number.

A JavaScript function was then generated to simulate the bot completing this field. This required edits to index.html, the login.php file and the style.css file to complete. The third security element to be added to the login page was a (Google, 2024) Captcha from the team at hCaptcha. The addition of this captcha would require an account setup on the SaaS platform hCaptcha. With this, the site key and secret key could be generated and the configuration settings of the hCaptcha could be modified. Once the hCaptcha keys were generated ChatGPT4 was prompted to add hCaptcha to the IoT login page. This required updating the index.html to include the hCaptcha widget to display on the login page and the site key. The login.php file would also require editing to include the hCaptcha secret key and response verification to ensure the form could be submitted after a successful Captcha test. (See Fig 7.0)

**FIG 7.0** Captcha, Honeypot and Timed Input Fields

The image shows a login form on a light blue background. The form has a white border and contains the following elements: a 'Username:' label above a text input field; a 'Password:' label above a text input field; an hCaptcha widget with a checkbox labeled 'I am human' and the hCaptcha logo; a blue 'Login' button; and a blue 'Simulate Bot Submission' button. The hCaptcha widget also includes links for 'Privacy' and 'Terms'.

There were now three security measures active on the IoT login page. Timed input fields, honeypot fields and Captcha by hCaphca. With all three security measures in place and tested, it was time to run the `Mirai_Simulation_Selenium.py` Python script. This would attempt to brute force the login page with 10 bots. The output from the test was saved and no bots managed to brute force the IoT device login page. (See Fig 8.0)

**FIG 8.0** - Simulation three completed – Device not compromised.

```
Starting simulation with 10 bots...
100% | 10/10 [02:05<00:00, 12.58s/it]
Simulation completed.
Device not compromised after 125.99 seconds with 10 bots attacking.
(myenv) Dans-MacbookPro:htdocs druanes$
```

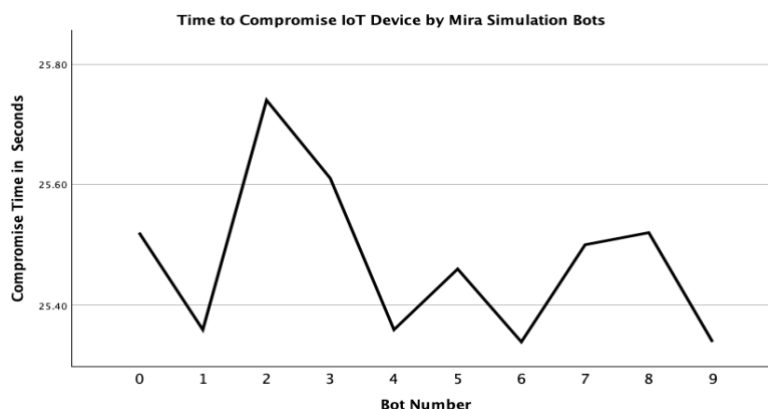
## 12 Evaluation



## 12.1 Experiment 1

The first experiment in our testing was created to establish an attack base line. This experiment used zero security measures on the simulation IoT device login page. The below graph in Fig 9.0 shows that all 10 bots were able to compromise the device login page in under 26s. Each bot supplied a screenshot of the success page after login with bot number 9 successfully brute forcing the login page first in 25.35s

**FIG 9.0** Time in seconds to compromise IoT Login Page



To put this information into a real-world scenario requires us to use more bots and have the attack run with more time allocated. If we consider the example of 100 Mirai bots trying to attack the device over a 12-hour period, we can deduce the following with zero security measures.

Time to Successful Brute Force Without Any Security Measures – 24.35s

Number of seconds in 1 hour = 3600s

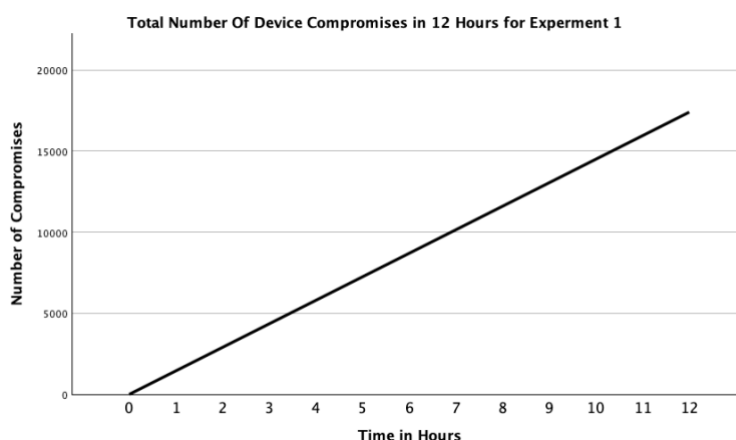
Number of compromises per hour with 10 bots =  $3600/25.34 = 142$

$142 \times 12$  to give a 12-hour period = 1704 with 10 bots but what about 100 bots?

$1704 \times 10 = 17,040$  successful brute force attempts with 100 bots attacking.

Therefore, without any security measures in place and a botnet size of well over 100 with 100 bot devices attacking the IoT Device login page, it would be compromised well over 17,040 times in a 12-hour period. (See Graph in Fig 10.0)

**Graph – FIG 10.0**



## 12.2 Experiment 2

The next experiment introduced one single security measure. Timed input fields. This experiment ran for 125 seconds, and the IoT Device login form was not compromised (See Fig 9.0) This security measure monitors a bot's heuristics and the speed at which it completes the form. If it completes the form in under 500ms, the form submission is blocked, it is approximately 50% effective at filtering out the number of bots that try to brute force the login page. This is because bots are becoming (Rose, 2024) smarter and can mimic the heuristics of a human on a web page. The older bots could not do this and since 2020, the block rate success of timed input fields has dropped from approximately 88% down to approximately 50%.

100 bots reduced by 50% = 50 Bots attacking across 12 hours and we must also consider that the brute force can happen no faster than 500ms. Considering it takes 100ms for a single bot to make a brute force attempt we must multiply the time taken by 5.

$24.35 \times 5 = 121.75$  seconds per successful brute force attempt with 10 bots

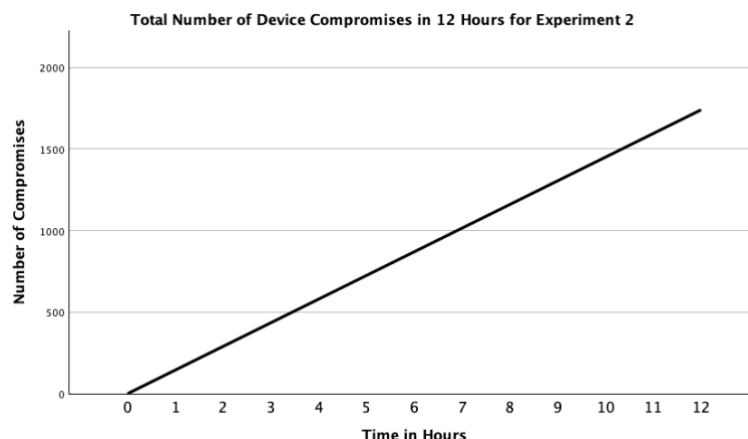
3600s in 1 hour/ 121.75 = 29 successful brute force attempts in 1 hour by 10 Bots

$29 \times 5 = 145$  successful attempts by 50 bots in 1 hour

$145 \times 12 = 1,740$  successful brute force attempts in 12 hours.

Therefore, by adding the timers on the input fields, the form has become **~10x more secure**. (See Fig 11.0)

**FIG 11.0** 10x Times More Secure than Experiment 1



## 12.3 Experiment 3

The third experiment used all three security measures and ended in 125 seconds also without any compromises occurring. Timed input fields were used on the form along with Honeypot fields which are approximately (Sheikh, 2020) 90% effective and the hCaptcha which is approximately 99.9 percent effective thanks to its bot detection engine and this Captcha is now also used by Cloudflare and Shopify to protect their platforms from bots.

With experiment 2, its estimated that 1740 successful brute force attempts occurred over 12 hours.

If we now use the effectiveness of each measure, we can now deduce the overall successful attacks over 12 hours with all security measures in place.

1740 successful bot attacks over 12 hours, honeypot fields filter out 90% of the remaining attacking bots. Therefore  $1,740 \times 0.10 = 174$  bot attacks remain over 12-hour period

Also adding hCaptcha we get approximately 99.9% success rate at blocking the remaining bots.

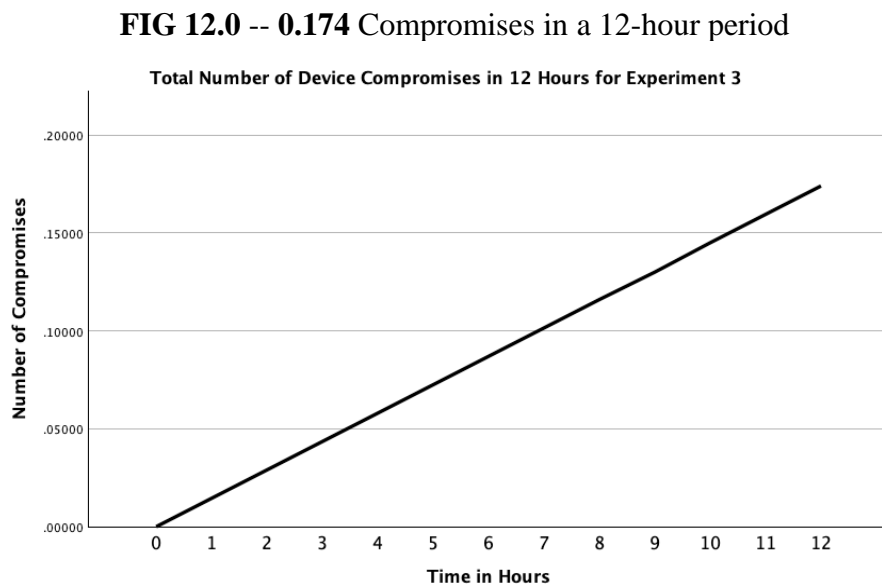
Therefore  $174 \times 0.001 = \mathbf{0.174}$  attacks over a 12-hour period with all security measures in place.

Now let's calculate how long it would take for 1 bot to successfully attack the IoT login page with all the security measure in place.

$1/0.174 = 5.747$ . Therefore, we multiply 12 hours by 5.747 to get **68.97 hours** or approximately **3 days** before our IoT Login page would become compromised.

This is of course an approximation, and many factors could shorten or lengthen the time to compromise. Now let's review the effectiveness of all the security measures by improvement factor. The original unsecured IoT login form became compromised 17,040 times in 12 hours and with all the security measures in place there were 0.174 successful attacks in 12 hours.  $17040/0.174 = 97,931$

The total effectiveness improvement factor is **97,931** with all security measures in place (See Fig 12.0)



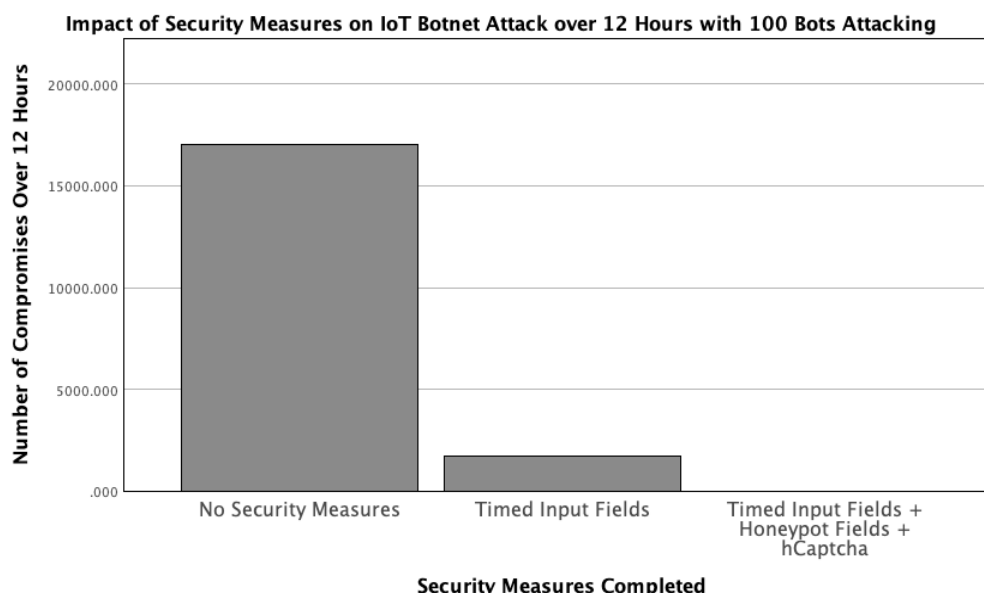
## 12.4 Discussion

The results derived from the three experiments above show that a defence in depth strategy is required to protect the confidentiality, integrity and availability of IoT devices. There are many ways an IoT device can become compromised, however the most common way is by users leaving the default configuration of username and password in place while the device is in use. The three experiments that were undertaken showed that its important to use security measures on any login areas to prevent botnet propagation. The experiments were somewhat constrained by the amount of compute available. Any more than 10 bots would cause a failure and the simulation would crash. With unlimited simulation power, it would have been possible to simulate 100 or even 10,000 bots.

This would have given a more exact figure of the time it would take for a botnet to acquire a slave device via brute force technique. The overall impact of the designed security measure

held up very well and the three measures of timed input fields + honeypot fields + hCaptcha performed well in the simulation (See Fig 13.0)

**FIG 13.0 – Overall Impact of Security Measures on the IoT Device**



There are many more layers of defence that can be added to the IoT device login in form. The next layer I would use is “input validation”. I did not use this initially as I was looking for the biggest impact from a particular security element. The rockyou\_subset.txt file comes from the latest version of the rockyou file and contains updated passwords that are at least 8 characters long that meet many of the password input validation checks and therefore it would not make a huge impact on the brute force attack. If this were an injection attack simulation, input validation would be my first security measure that I would use to block attacks. The use of rate limiting of login attempts from various IP address and the setup of an automatic five-day access block on the second or third incorrect password attempt is another method I would like to test in a future project.

## 13 Conclusion and Future Work

The question of how to restrict botnet propagation via local IoT device security measures has been an interesting project to research. The world currently revolves around the uptime of services and a brand will only be as good as their product, which is required to be up and online 24/7. The DDoSaaS platforms that are available on the dark web for hire will become more and more popular as ransomware moves into triple and quadruple extortion where attackers try to take down systems belonging to their victims to make them pay the ransom. Its then incredibly important for IoT manufacturers to up their game when it comes to the S-SDLC of IoT devices. As we have seen in the above experiments a well-protected login page is a very good first step to block the propagation of botnets and therefore reduce the overall size of a DDOS attack.

Another important measure, that has not been discussed in this research project is network alerts which happen if a device is under attack on a network, think endpoint detection and

response or EDR. As we have seen from the experiments, even with a layered approach to security mechanisms on a device, it will still become compromised in three days. Therefore, if the attack is not blocked at the network level, the device will then become compromised. This shows how important network security alerts are when strange traffic is witnessed on the network. As we have seen in the research, it's also very important to stop using default passwords and instead use the devices serial number as a password or go password less and use two step authentication and biometric login. These are some future ideas and research projects I would like to review to help secure the internet and keep services online and running 24 hours a day.

## References

### 14 Bibliography

S. Al-Sarawi, M. Anbar, R. Abdullah and A. B. Al Hawari, "Internet of Things Market Analysis Forecasts, 2020–2030," *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, London, UK, 2020, pp. 449-453

Kelly, C., Pitropakis, N., McKeown, S. and Lambrinoudakis, C., 2020, June. "Testing and hardening IoT devices against the Mirai botnet." In *2020 International conference on cyber security and protection of digital services (cyber security)* (pp. 1-8). IEEE.

Shah, T. and Venkatesan, S., 2019. A method to secure IoT devices against botnet attacks. In *Internet of Things–ICIOT 2019: 4th International Conference, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 4* (pp. 28-42). Springer International Publishing.

P. Research, "Internet of Things (IoT) Market," [Online]. Available: <https://www.precedenceresearch.com/internet-of-things-market>. [Accessed 11th April 2024].

S. Laborde, "43+ Stunning IoT Statistics for 2023 [The Rise of IoT]," [Online]. Available: <https://techreport.com/statistics/internet-of-things-statistics/>. [Accessed 11th April 2024].

Liao, B., Ali, Y., Nazir, S., He, L. and Khan, H.U., 2020. *Security analysis of IoT devices by using mobile computing: a systematic literature review*. *IEEE Access*, 8, pp.120331-120350.

Davis, B.D., Mason, J.C. and Anwar, M., 2020. *Vulnerability studies and security postures of IoT devices: A smart home case study*. *IEEE Internet of Things Journal*, 7(10), pp.10102-10110.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M. and Kumar, D., 2017. *Understanding the mirai botnet*. In *26th USENIX security symposium (USENIX Security 17)* (pp. 1093-1110)

B. M. J. a. A. M. Davis, "Vulnerability studies and security postures of IoT devices: A smart home case study," *IEEE Internet of Things Journal*, Online, 2020.

M. Colon, "Mirai Botnet Propagates By Leveraging Weak Default Passwords," 2016. [Online]. Available: <https://www.scmagazine.com/news/mirai-botnet-propogates-by-leveraging-weak-default-passwords>.

C. Deshpande, "What is Selenium: Getting Started with Automation Testing," 2023. [Online]. Available: <https://www.simplilearn.com/tutorials/selenium-tutorial/what-is-selenium#:~:text=Selenium%20is%20an%20open%2Dsource,test%20software%20and%20mobile%20applications>

R. Shaditya, "Python / How to make a terminal progress bar using tqdm," 2022. [Online]. Available: <https://www.geeksforgeeks.org/python-how-to-make-a-terminal-progress-bar-using-tqdm/>.

H. Team, "Private Learning is a Revolution in Security ML," 2024. [Online]. Available: <https://www.hcaptcha.com/#deploy>.

Edison, "How to enable Honeypot field in Comments Form for anti-spam," 10 July 2019. [Online]. Available: <https://processwire.com/talk/topic/21923-how-to-enable-honeypot-field-in-comments-form-for-anti-spam/#:~:text=Honeypot%20is%20simply%20an%20hidden,and%20will%20populate>

Google, "What is CAPTCHA?," 2024. [Online]. Available: <https://support.google.com/a/answer/1217728?hl=en#:~:text=CAPTCHA%20helps%20protect%20you%20from>

M. T. B. & S. A. Sheikh, "Improving Security Control of Text-Based CAPTCHA Challenges using Honeypot and Timestamping," 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/9076410>

K. Rose, "Bot Detection: What It Is and How to Block Bad Bots," 2024. [Online]. Available: <https://fingerprint.com/blog/bot-detection/>. [Accessed 6th August 2024].