การต้านทานและการกู้คืนจากการโจมตี DDoS ของบอร์ด Raspberry Pi ที่ทำงานเป็น IoT ผ่านไวไฟ

อินทัชพงค์ รัตนดิลก ณ ภูเก็ต

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
ปีการศึกษา 2564

RESISTANCE AND RESILIENCE FROM DDoS ATTACK OF RASPBERRY PI BOARD AS WIFI IoT SYSTEM

INTOUCHPONG RATTANADILOK NA PHUKET

A Thematic Paper Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Engineering

Department of Computer Engineering

College of Innovative Technology and Engineering,

Dhurakij Pundit University

Academic Year 2021

ใบรับรองสารนิพนธ์

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

หัวข้อสารนิพนธ์	การต้านทานและการกู้คืนจากการโจมตี DDoS ของบอร์ด Raspberry Pi ที่ทำงานเป็น IoT ผ่านไวไฟ
เสนอโดย	นายอินทัชพงศ์ รัตนดิลก ณ ภูเก็ต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาสารนิพนธ์	อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์

ได้พิจารณาเห็นชอบโดยคณะกรรมการสอบสารนิพนธ์แล้ว

.....ประธานกรรมการ

(รองศาสตราจารย์ ดร.ลัญฉกร วุฒิสิทธิกุลกิจ)

..... กรรมการและอาจารย์ที่ปรึกษาสารนิพนธ์

(อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.มัชฌิกา อ่องแตง)

วิทยาลัย นวัตกรรมด้านเทคโนโลยีและวิศวกรรมคอมพิวเตอร์รับรองแล้ว

.....คณบดีวิทยาลัย นวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

(อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์)

วันที่ 30 เดือน ก.ค. พ.ศ. 2565

หัวข้อสารนิพนธ์	การต้านทานและการกู้คืนจากการโจมตี DDoS ของบอร์ด Raspberry Pi ที่ทำงานเป็น IoT ผ่านไวไฟ
ชื่อผู้เขียน	อินทัชพงค์ รัตนดิลก ณ ภูเก็ต
อาจารย์ที่ปรึกษา	ดร.ชัยพร เขมะภาตะพันธ์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ปีการศึกษา	2564

บทคัดย่อ

ปัจจุบันเทคโนโลยี IoT มีการนำมาประยุกต์ใช้ในหลายอุตสาหกรรม หรือภายในชีวิตประจำวันเช่น บ้านอัจฉริยะ (Smart Home) เพื่อเพิ่มความสะดวกสบายในการใช้ชีวิต ทำให้เทคโนโลยีเป็นที่นิยมและมีการพัฒนาอย่างรวดเร็ว หากแต่เทคโนโลยีในด้านความมั่นคงปลอดภัยของเทคโนโลยี IoT ไม่ได้รับการพัฒนาให้มีประสิทธิภาพและไม่พร้อมรับมือกับการโจมตีทางไซเบอร์ในปัจจุบัน

งานวิจัยนี้ได้นำเสนอการโจมตีทางไซเบอร์ในรูปแบบ DDoS ไปยังระบบ IoT แบบบ้านอัจฉริยะ (Smart Home) ที่เป็นที่นิยมประยุกต์ใช้กันอย่างแพร่หลายให้ปัจจุบัน และยังสามารถพัฒนาขึ้นมาได้ด้วยตนเอง เพื่อแสดงให้เห็นถึงผลกระทบ ผลที่เกิดจากการโจมตี ขีดจำกัดของกำลังประมวลผลของอุปกรณ์ และความสามารถในการกู้คืนตนเองของอุปกรณ์

โดยงานวิจัยนี้ได้แสดงถึงผลลัพธ์ของการโจมตีที่ส่งผลกระทบกับระบบ IoT แบบบ้านอัจฉริยะ ทำให้ผู้ใช้งานไม่สามารถควบคุมหรือสั่งการอุปกรณ์ได้ในขณะที่เกิดการโจมตี ในรูปแบบ TCP Syn Flood Attack ส่วนการโจมตีแบบ HTTP Get Flood Attack แสดงให้เห็นถึงขีดความสามารถในการประมวลผลของอุปกรณ์ภายในระบบ การโจมตีในรูปแบบ ICMP Flood Attack และ UDP Flood Attack แสดงให้เห็นถึงประมาณการใช้งานเครือข่ายที่สูงขึ้น ทำให้การเรียกใช้บริการต่าง ๆ ของระบบผ่านเครือข่ายใช้เวลาสูงขึ้น

Thematic Paper Title	RESISTANCE AND RESILIENCE FROM DDoS ATTACK OF RASPBERRY PI BOARD AS WIFI IoT SYSTEM
Author	Intouchpong Rattanadilok Na Phuket
Thematic Advisor	Dr.Chaiyaporn Khemapatapan
Department	Computer Engineering
Academic Year	2021

ABSTRACT

Today, IoT technologies are applied in many industries. or within everyday life such as Smart Home to increase convenience of living, which make this technology popular and developing rapidly. But the security technology of IoT is not robust and not ready to deal with cyber attacking.

This Research proposes a DDoS Attack to the most widely applied Smart Home IoT System. And can also be developed by yourself. To demonstrate the effect, the consequences of attack, devices processing limitation and resilience of the device.

This Research shows the impact of attack on Smart Home IoT system. thus, users cannot control or operate the device while the attack occurs with TCP Syn Flood Attack. Http Get Flood Attack shows the processing capabilities of the device in IoT system. ICMP Flood Attack and UDP Flood Attack show higher estimate of network bandwidth usage, which make more time to access a service over the network.

กิตติกรรมประกาศ

สารนิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้อย่างสมบูรณ์ โดยได้รับความอนุเคราะห์เป็นอย่างยิ่งจากอาจารย์ ดร.ชัยพร เขมะภาตะพันธ์ อาจารย์ที่ปรึกษาที่ได้ให้ข้อคิดเห็น คำแนะนำ คำปรึกษาที่เป็นประโยชน์ต่องานวิจัย และดูแลเอาใจใส่นักศึกษาเสมอมา

ขอขอบพระคุณอาจารย์ทุกท่าน รุ่นพี่ เพื่อนร่วมรุ่น และเพื่อนร่วมงานที่ร่วมให้กำลังใจและข้อเสนอแนะในการดำเนินการจัดทำสารนิพนธ์ฉบับนี้จนสำเร็จลุล่วงไปด้วยดี

ขอขอบคุณมหาวิทยาลัยธุรกิจบัณฑิตย์

ท้ายที่สุดนี้ขอขอบคุณ พ่อ แม่ และบุคคลในครอบครัว ที่คอยให้กำลังใจและการสนับสนุนผู้วิจัยในทุกๆ ด้าน ตลอดระยะเวลาการศึกษาจวบจนสำเร็จการศึกษา

อินทัชพงค์ รัตนดิลก ณ ภูเก็ต

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเทคโนโลยี IoT ได้เข้ามามีส่วนช่วยพัฒนาการดำรงชีวิตของมนุษย์ในปัจจุบันให้สะดวกสบายขึ้น ช่วยลดขั้นตอนต่างๆ ในกระบวนการทำงาน ช่วยเพิ่มประสิทธิภาพของผลลัพธ์ หรือแม้กระทั่งช่วยเพิ่มความมั่นคงปลอดภัย การประยุกต์ใช้เทคโนโลยี IoT เกิดขึ้นทั้งในภาคอุตสาหกรรม หรือแม้กระทั่งภายในครัวเรือน ที่มีการนำเทคโนโลยีมาประยุกต์ใช้ให้เกิดเป็นบ้านอัจฉริยะ (Smart Home) เพื่อช่วยเพิ่มประสิทธิภาพภายในบ้าน และเสริมสร้างความสะดวกสบาย หรือความปลอดภัยให้แก่ผู้อยู่อาศัยและตัวเทคโนโลยี IoT เป็นองค์ความรู้ที่แพร่หลาย สามารถศึกษาหรือพัฒนาได้ด้วยตนเอง ทำให้การนำเทคโนโลยี IoT มาประยุกต์ใช้เพิ่มขึ้นแบบก้าวกระโดด

หากแต่ความสามารถด้านความมั่นคงปลอดภัยของตัวเทคโนโลยี IoT เองยังไม่ถูกสนใจหรือถูกพัฒนาให้เหมาะสมและครอบคลุมกับการใช้งานต่าง ๆ เท่าที่ควรทำให้เทคโนโลยี IoT อาจเป็นสาเหตุให้เกิดความเสียหาย หรือ อันตรายแก่ชีวิตและทรัพย์สิน ผ่านรูปแบบการโจมตีทางไซเบอร์ในรูปแบบต่างๆ เช่นการโจมตีแบบ DDoS และ DRDoS ที่เป็นที่นิยมและพบมากที่สุด โดยจุดอ่อนในเทคโนโลยี IoT ในปัจจุบันเกิดขึ้นในหลายส่วนเช่น Protocol, Sensors หรือแม้กระทั่งกำลังการประมวลผลของตัวอุปกรณ์ IoT ก็สามารถที่จะเป็นช่องโหว่ของการโจมตีได้

1.2 วัตถุประสงค์ของงานวิจัย

วัตถุประสงค์ของงานวิจัยฉบับนี้เป็นการนำเสนอการโจมตีในรูปแบบ DDoS ไปยังระบบ IoT แบบ Smart Home ที่จำลองขึ้นโดยการใช้เทคนิคการโจมตีแบบ

1. HTTP GET Flood Attack
2. TCP SYN Flood Attack
3. ICMP Flood Attack
4. UDP Flood Attack

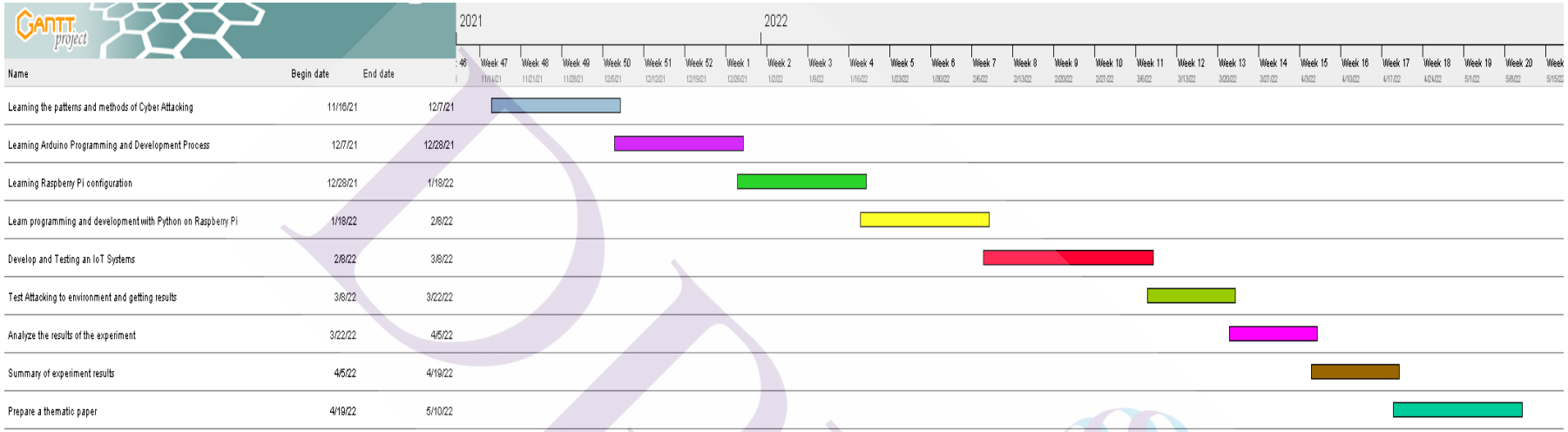
เพื่อศึกษาผลที่เกิดจากการโจมตีและเกิดระหว่างการโจมตี

1. ศึกษาปริมาณ Packet ที่ระบบสามารถตอบสนองได้
2. ศึกษาผลการทำงานของระบบ IoT ที่เกิดระหว่างการโจมตี
3. ศึกษาผลการทำงานของระบบ IoT ที่เกิดหลังการโจมตีสิ้นสุดลง
4. ศึกษาความสามารถในการกู้คืนตัวเอง (Resilience) ของระบบ IoT จำลอง

1.3 เครื่องมือที่ใช้ในงานวิจัย

1. Raspberry Pi Mobel B 8 GB
2. NodeMCU ESP 8266-12E V3 CH340 Lua WIFI
3. DHT22 Digital Temperature and Humidity Sensor
4. LED Sensor
5. Router TP-Link Archer-c64 AC-1200 Wireless
6. Lan Cable CAT.7 Network Ethernet
7. Macbook Pro 2015
8. PC Desktop
9. Virtual Machine : Kali Linux
10 . Software : Arduino, VSCodium, Wireshark, iTerm, Firefox(App Telemetry)
11. Operating System : Mac OS, Raspbian OS, Windows OS
12. Programming Language : Python, C++

1.4 แผนการดำเนินงาน



Name	Begin date	End date
Learning the patterns and methods of Cyber Attacking	11/16/21	12/7/21
Learning Arduino Programming and Development Process	12/7/21	12/28/21
Learning Raspberry Pi configuration	12/28/21	1/18/22
Learn programming and development with Python on Raspberry Pi	1/18/22	2/8/22
Develop and Testing an IoT Systems	2/8/22	3/8/22
Test Attacking to environment and getting results	3/8/22	3/22/22
Analyze the results of the experiment	3/22/22	4/5/22
Summary of experiment results	4/5/22	4/19/22
Prepare a thematic paper	4/19/22	5/10/22

1.5 ขอบเขตของงานวิจัย

1. ระบบ IoT ที่ใช้ทดสอบเป็นการจำลองระบบ IoT ในรูปแบบ Smart Home ที่สามารถสร้างและพัฒนาขึ้นได้ด้วยตนเอง

2. ระบบ IoT ที่จำลองขึ้นมาเพื่อใช้ในการทดสอบเป็นระบบปิดไม่มีการเชื่อมต่อกับ Internet ภายนอก

3. การเชื่อมต่อของเครือข่ายทั้งหมดในระบบ IoT ถูกนิยามความเร็วของการรับส่งข้อมูลเป็นแบบความเร็วสูงสุด

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำผลที่ได้จากการโจมตีไปพัฒนาต่อในด้านความมั่นคงปลอดภัยให้กับระบบ IoT ที่ถูกพัฒนาขึ้นในแบบพัฒนาด้วยตนเอง (In-House)

2. สามารถนำผลที่ได้จากการโจมตีไปพัฒนาต่อในการรับส่งข้อมูลผ่าน MQTT Protocol ให้มีความมั่นคงปลอดภัย

3. สามารถนำค่าสถิติต่างๆ จากการทดลองเพื่อนำไปเป็นคุณสมบัติมาตรฐาน (Base Line) ในการเลือกใช้ Microprocessor หรือ Microcontroller ในอนาคต

บทที่ 2
แนวคิดทฤษฎีและงานวิจัยที่เกี่ยวข้อง

งานวิจัยนี้จะกล่าวทฤษฎีที่เกี่ยวข้องกับ Cyber Attacking, Internet of Things (IoT) และบ้านอัจฉริยะ (Smart Home) ซึ่งประกอบด้วยส่วนประกอบภาคต่าง ๆ คือ ส่วนที่ใช้ไมโครโปรเซสเซอร์, ส่วนที่ใช้ไมโครคอนโทรเลอร์, ส่วนที่จำลองการโจมตี และส่วนที่ใช้สำหรับการสื่อสาร (MQTT Protocol)

โดยงานวิจัยที่มีการนำเสนอถึงความมั่นคงปลอดภัย (Security) และการโจมตีทางไซเบอร์ (Cyber Attacking) ที่เกี่ยวข้องกับ IoT นั้น และเป็นงานวิจัยที่ศึกษาระบบ IoT ในรูปแบบบ้านอัจฉริยะ (Smart Home) ซึ่งกำลังได้รับความนิยมอย่างแพร่หลายในปัจจุบัน ได้มีการนำเสนอรูปแบบการโจมตีและผลกระทบจากการโจมตีที่แตกต่างกันออกไป

2.1 งานวิจัยเรื่อง A Survey: DDoS Attack on Internet of Things

เป็นงานวิจัยของ Krushang Sonar and Asst Prof. Hardik Upadhyay, Gujarat Technology University, India, International Journal of Engineering Research and Development, 2014 เป็นงานวิจัยที่ศึกษารูปแบบการโจมตีแบบ DDoS ทั้งหมดที่เป็นไปได้ ที่สามารถโจมตีระบบหรืออุปกรณ์ IoT ให้ไม่สามารถให้บริการ หรือตอบสนองต่อผู้ใช้งานได้ โดยผลจากการศึกษาของงานวิจัยชิ้นนี้ ได้แบ่งรูปแบบหรือเทคนิคการโจมตีตามลักษณะของ IoT Architechture ดังภาพที่ 2.1

Application Layer Reprogramming Attack, Path Based DoS
Communication Between Network Layer and Application Layer WIFI [ICMP Flooding Attack] ZigBee [Hello Flooding Attack, Homing Attack, Black Hole Attack]
Network Layer Flooding Attack, Reflection-Based Flooding Attack, Protocol Exploitation Flooding Attack, Amplification-Based Flooding Attack
Perception Layer RFID [Jamming, Kill Command Attack, De-Synchronizing Attack] 802.15.4 [Wide-Band Denial and Pulse Denial, Node-Specific and Message Specific Denial, Bootstrapping Attack]

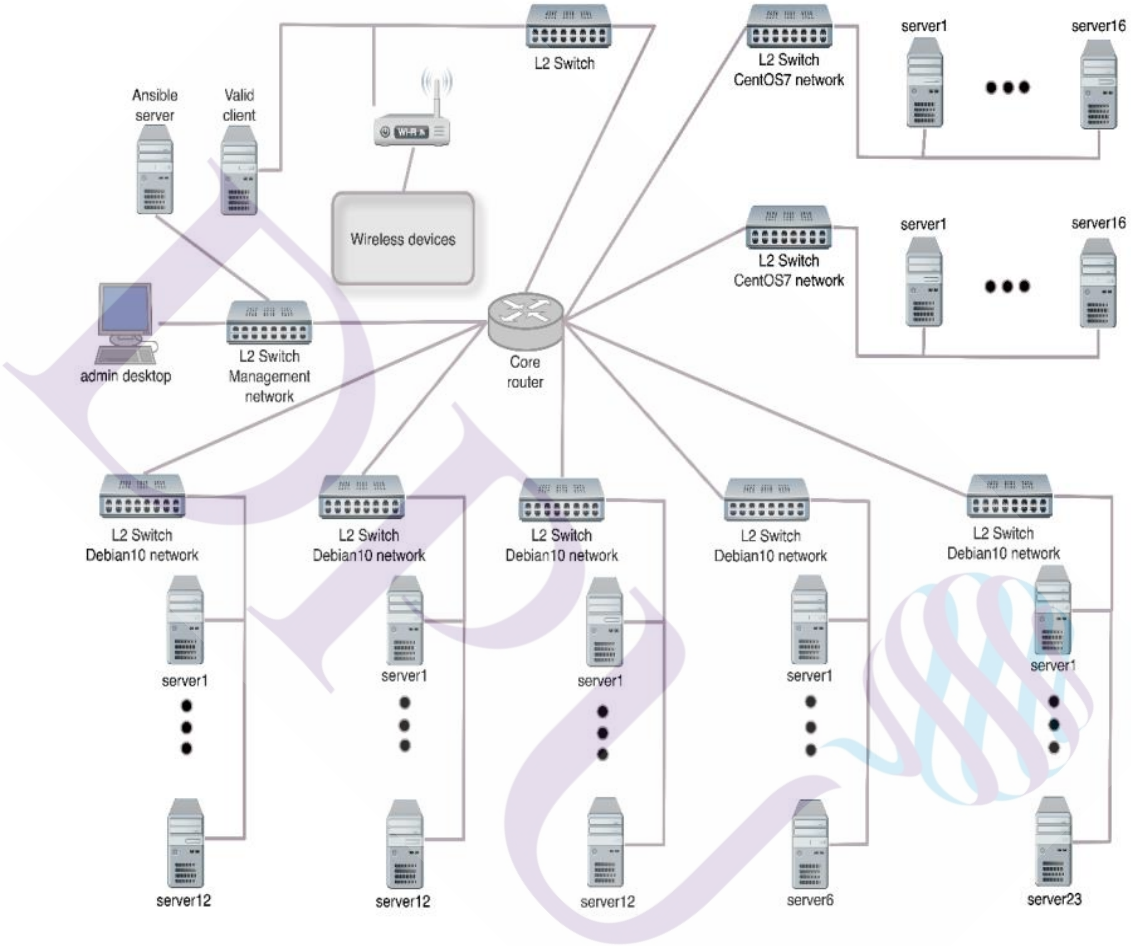
ภาพที่ 2.1 รูปแบบการโจมตีแบ่งตาม IoT Architecture

จากการศึกษางานวิจัยนี้ ทำให้เห็นถึงรูปแบบการโจมตีใบแบบต่างๆ ที่สามารถเกิดขึ้นได้กับแต่ละส่วนในสถาปัตยกรรมของระบบ IoT สามารถนำความรู้ของรูปแบบการโจมตีแต่ละชนิด มาประยุกต์สำหรับใช้สร้างชุดการโจมตีสำหรับการทดลองได้อย่างมีประสิทธิภาพ

2.2 งานวิจัยเรื่อง Resistance of IoT Sensors against DDoS Attack in Smart Home Environment

เป็นงานวิจัยของ Ladislav Huraj, Marek Simon, Department of Applied Informatics, University of SS. Cyril and Methodius และ Tibor Horak, Institute of Applied Informatics, Automation and Mechatronics, Faculty of Materials Science and Technology, Trnava, Slovak University of Technology, Slovakia, Sensors MDPI, 2020 เป็นงานวิจัยที่ศึกษาการโจมตีแบบ DDoS ไปยังอุปกรณ์ Home Assistance ที่มีวางขายทั่วไปตามห้องตลาด ที่สามารถนำมาเชื่อมต่อกัน

เป็น Center Hub สำหรับควบคุมอุปกรณ์ IoT ต่าง ๆ ภายในบ้านได้ โดยงานวิจัยชิ้นนี้เลือกรูปแบบการโจมตี 2 แบบได้แก่ SYN Flood Attack และ HTTP GET Flood Attack ภายในงานวิจัยชิ้นนี้ออกแบบ Environment สำหรับสร้างชุดการโจมตีโดยใช้ Ansible real-time attack environment เพื่อสร้างชุดโจมตีแบบ DDoS ดังภาพที่ 2.2



ภาพที่ 2.2 โครงสร้างของ Ansible real-time attack environment สำหรับสร้างชุดการโจมตีแบบ DDoS

โดยมีการเลือกอุปกรณ์ IoT ที่มีวางขายทั่วไปตามท้องตลาดตามรายการดังนี้สำหรับการทดสอบการโจมตี

Control Center

1. Fibaro System Management (Home Center)
2. Philips Hue Bridge 2.0
3. Amazon Echo Dot (3rd Gen)
4. Smart-home controller Athom Homey Pro 2.0
5. Google Home

IoT Devices

1. Fiabro Wall Plug
2. Smart LED bulbs Philips Hue

และเลือกใช้ Mobile Application สำหรับควบคุมและออกคำสั่งอุปกรณ์ IoT ตามรายการนี้

1. Philips Hue Application
2. Fibaro Home Center Application
3. Homey Application
4. Amazon Alexa Application
5. Google Home Application

แบ่งสถาณการณ์สำหรับการทดสอบออกแบบ 3 สถาณการณ์ดังนี้

1. ผลิตภัณฑ์ Control Center ทำงานร่วมกับ Mobile Application ของผลิตภัณฑ์
2. ผลิตภัณฑ์ Control Center ทำงานร่วมกับ Mobile Application ข้ามผลิตภัณฑ์
3. ผลิตภัณฑ์ Control Center ที่มีการประมวลแบบ Cloud ทำงานร่วมกับ Mobile Application ของผลิตภัณฑ์

จากการวิเคราะห์งานวิจัยนี้ แสดงให้เห็นว่าการโจมตีแบบ SYN Flood Attack ไม่ส่งผลต่อการทำงานของตัวอุปกรณ์ Control Center และอุปกรณ์ IoT แต่การโจมตีแบบ HTTP GET Flood Attack ส่งผลต่อการควบคุมหรือสั่งการอุปกรณ์ IoT ดังภาพที่ 2.3

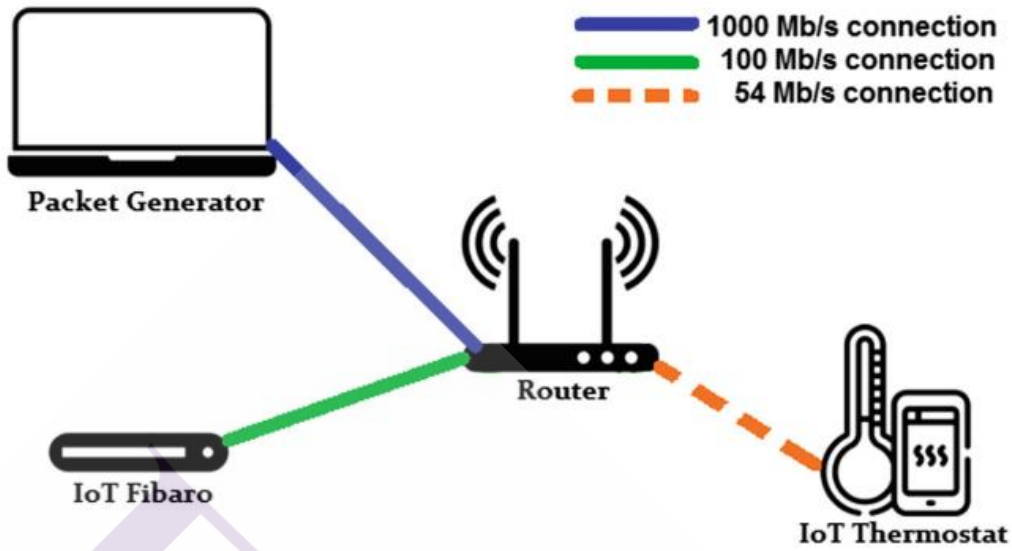
Scenario	Attacked Device	Controlling Mobile App	Testing Sensor	Communication
Scenario 1	Philips Hue Bridge	Philips Hue app	smart-bulb	With restrictions
Scenario 1	Fibaro Home Center 3	Fibaro Home Center app	wall plug	With restrictions
Scenario 2	Philips Hue Bridge	Homey app	smart-bulb	Inoperable
Scenario 2	Philips Hue Bridge	Amazon Alexa app	smart-bulb	Fully
Scenario 2	Philips Hue Bridge	Google Home app	smart-bulb	Fully
Scenario 2	Fibaro Home Center 3	Homey app	wall plug	Fully
Scenario 2	Fibaro Home Center 3	Amazon Alexa app	wall plug	Inoperable
Scenario 2	Fibaro Home Center 3	Google Home app	wall plug	Fully
Scenario 3	Athom Homey	Homey app	smart-bulb	Inoperable
Scenario 3	Amazon Echo Dot	Amazon Alexa app	smart-bulb	Fully
Scenario 3	Google Home	Google Home app	smart-bulb	Fully
Scenario 3	Athom Homey	Homey app	wall plug	Inoperable
Scenario 3	Amazon Echo Dot	Amazon Alexa app	wall plug	Fully
Scenario 3	Google Home	Google Home app	wall plug	Fully

ภาพที่ 2.3 ภาพรวมแสดงผลการโจมตีในรูปแบบ HTTP Get Flood Attack

โดยผลการโจมตีแบบ Restrictions คือไม่สามารถควบคุมหรือสั่งการอุปกรณ์ IoT ได้เลย ผลการโจมตีแบบ Inoperable คือสามารถสั่งการ หรือออกคำสั่งผ่าน Mobile Application ได้ แต่อุปกรณ์ IoT ไม่ตอบสนองต่อชุดคำสั่งนั้น ผลการโจมตีแบบ Fully คือสามารถออกคำสั่งหรือควบคุมการทำงานของ Mobile Application และอุปกรณ์ IoT ได้ปกติ ซึ่งได้ข้อสังเกตุว่าอุปกรณ์ Control Center ที่มีการประมวลคำสั่งแบบ Cloud สามารถทนต่อการโจมตีได้อย่างสมบูรณ์

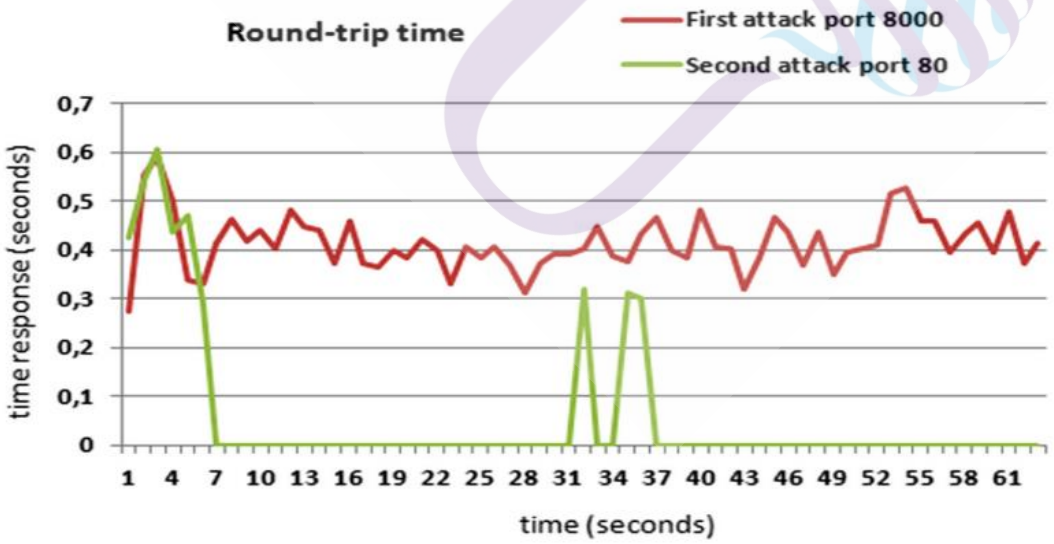
2.3 งานวิจัยเรื่อง Vulnerability of Smart IoT-Based Automation and Control Devices to Cyber Attack

เป็นงานวิจัยของ Tibor Horak, Marek Simon, Ladislav Huraj, and Roman Budjac, Institute of Applied Informatics, Automation and Mechatronics, Faculty of Materials Science and Technology, Trnava, Slovak University of Technology, Slovakia, Sensors MDPI, 2020 เป็นงานวิจัยที่ศึกษาและทดสอบการโจมตีในรูปแบบของ DDoS และ DRDoS ไปยังอุปกรณ์ Home Assistance ของผลิตภัณฑ์ Fibaro ที่เชื่อมต่อกับอุปกรณ์ IoT อย่าง Thermostat ในรูปแบบเครือข่ายของ Star Topology ดังภาพที่ 2.4



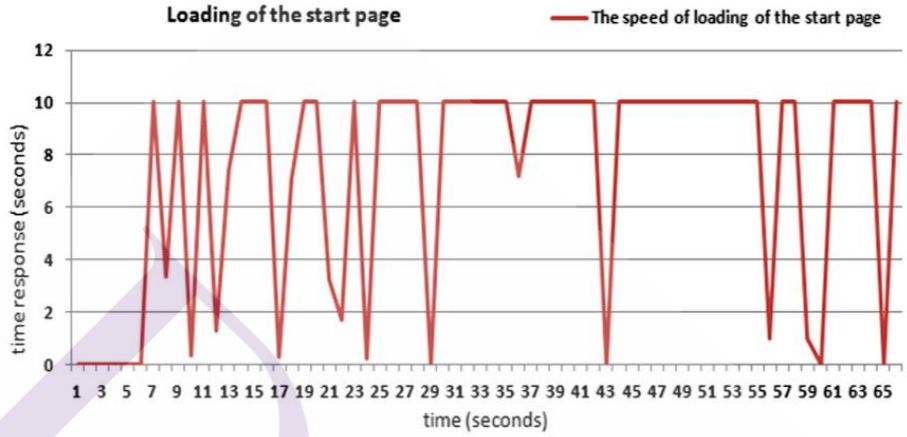
ภาพที่ 2.4 Topology สำหรับการทดลอง

การทดสอบการโจมตีแบบ DDoS ไปยังอุปกรณ์ IoT Fibaro ใช้วิธีการโจมตีในรูปแบบ SYN Flood Attack ไปยัง port 80 และ 8000 ภายใต้ระยะเวลาการโจมตีที่ 60 วินาที แสดงให้เห็นถึงการไม่ตอบสนองของอุปกรร์ตลอดระยะเวลาการโจมตีดังภาพที่ 2.5



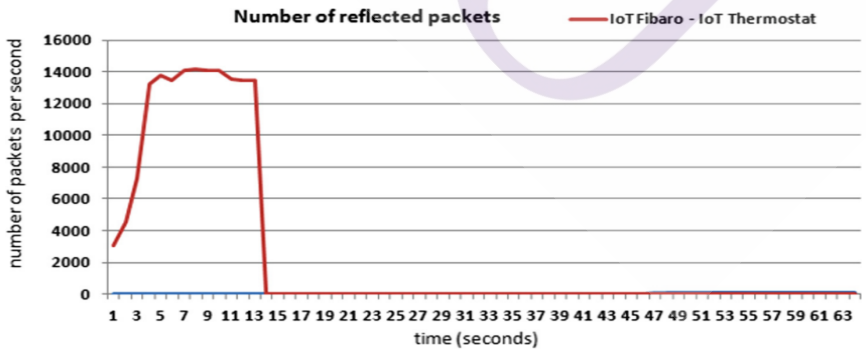
ภาพที่ 2.5 กราฟแสดงผลจากการโจมตีในรูปแบบ DDoS

และสำหรับการโจมตีในรูปแบบ HTTP GET Flood Attack ภายใต้ระยะเวลาการโจมตี 60 วินาทีโดยการใช้ Source การโจมตีที่มากกว่า 9000 Sources แสดงให้เห็นระยะเวลาของการตอบสนองที่เพิ่มสูงขึ้นอย่างต่อเนื่องตลอดระยะเวลาการโจมตีดังภาพที่ 2.6



ภาพที่ 2.6 กราฟแสดงระยะเวลาการตอบสนองต่อการโจมตีแบบ HTTP Get Flood Attack

การโจมตีในรูปแบบ DRDoS ไปยังอุปกรณ์ Fibaro โดยวิธีการโจมตีแบบ ICMP Echo Flood Attack เพื่อให้เกิดการสะท้อน Packet ของการโจมตีไปยังอุปกรณ์ IoT Thermostat แสดงให้เห็นว่าอุปกรณ์ Thermostat ไม่สามรถวัดค่าอุณหภูมิและความชื้นได้ตลอดระยะเวลาการโจมตีดังภาพที่ 2.7

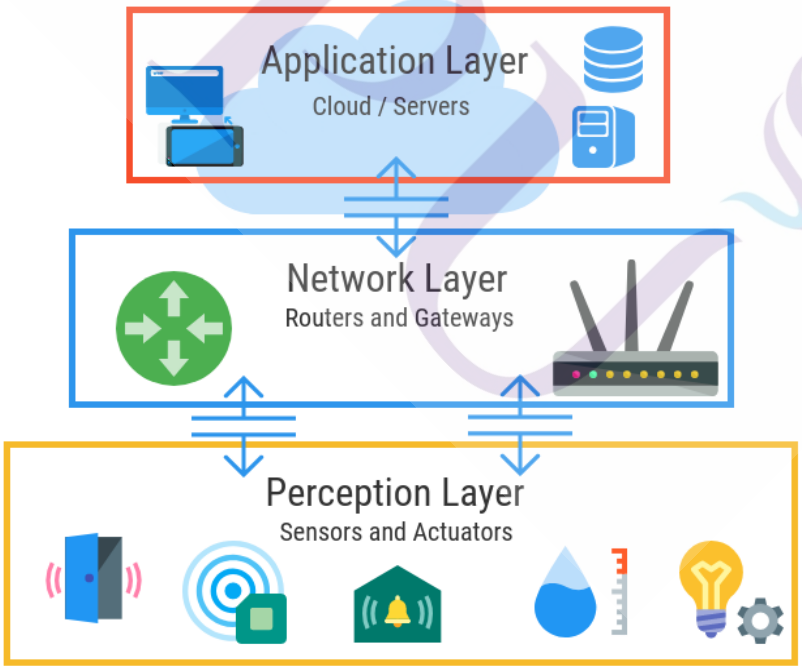


ภาพที่ 2.7 กราฟแสดงการไม่ตอบสนองของอุปกรณ์ Thermostat ต่อจำวน Packet ที่เกิดสะท้อนมายังตัวอุปกรณ์

จากการวิเคราะห์งานวิจัยชิ้นนี้ ทำให้ทราบว่าการโจมตีในรูปแบบ DDoS ส่งผลต่อการทำงานของที่ถูกโจมตีและการโจมตีในรูปแบบ DRDoS นอกจากอุปกรณ์ที่ถูกโจมตีจะส่งผลให้เกิดผลกระทบต่อการทำงานแล้ว ยังเกิดการสะท้อน Packet ไปยังอุปกรณ์อื่นๆ ที่เชื่อมต่ออยู่ทำให้ผลกระทบต่ออุปกรณ์อื่นๆ ด้วย

2.4 ทฤษฎี IoT (Internet of Thing)

Internet of Things (IoT) หรือ อินเตอร์เน็ตสรรพสิ่ง เป็นแนวคิดที่เกี่ยวกับเครือข่ายวัตถุ อุปกรณ์ พาหนะ สิ่งปลูกสร้าง และสิ่งอื่นๆ ที่มีวงจรอิเล็กทรอนิกส์ ซอฟต์แวร์ เซนเซอร์ และการเชื่อมต่อกับเครือข่าย ซึ่งทำให้วัตถุเหล่านี้สามารถเก็บบันทึกและแลกเปลี่ยนข้อมูลได้ ซึ่ง IoT ทำให้วัตถุสามารถรับรู้สภาพแวดล้อมและสามารถบังคับ/ความคุมได้จากระยะไกลผ่านเครือข่ายการสื่อสารที่มีอยู่แล้ว ผลที่ได้ตามมาจาก IoT คือประสิทธิภาพ ความแม่นยำ และประโยชน์ทางอุตสาหกรรมทางเศรษฐกิจ โดยโครงสร้างทางสถาปัตยกรรมของ IoT ประกอบด้วย 3 ระดับชั้น ดังภาพที่ 2.7



ภาพที่ 2.8 สถาปัตยกรรมของทฤษฎี IoT

ชั้น Perception Layer เป็นชั้นที่เกี่ยวข้องกับอุปกรณ์ Sensors, อุปกรณ์ Actuators ที่มีความสามารถรับรู้หรือตอบสนองต่อสภาพแวดล้อมรอบๆตัว โดยลักษณะพื้นฐานของชั้น Perception Layer มี 4 ประการ ดังนี้

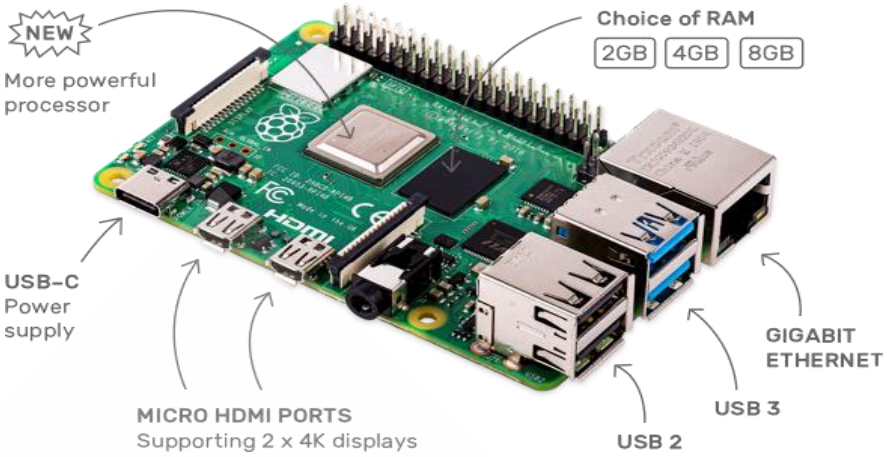
1. ระบุวัตถุ (Identify Objects)
2. รับรู้วัตถุ (Perceive Objects)
3. เก็บรวบรวมข้อมูล (Correct Information)
4. ควบคุมแบบอัตโนมัติ (Automatically Control)

ชั้น Network Layer เป็นชั้นที่เกี่ยวกับการรับส่งข้อมูลหรือการติดต่อสื่อสารผ่านเครือข่าย ซึ่งทำงานร่วมกับชั้น Application Layer โดยหน้าที่สำคัญที่เกิดขึ้นภายในชั้นนี้คือ การส่งผ่านข้อมูล, คัดเลือกเส้นทาง และควบคุมการรับส่งข้อมูล ภายในชั้น Network Layer นี้ประกอบด้วย 2 ส่วนหลักคือ เทคโนโลยีที่เกี่ยวกับการสื่อสารและ Protocol ที่มีหลากหลายในปัจจุบัน ดังนี้ MQTT, DDS, AMQP, XMPP, JMS, REST เป็นต้น

ชั้น Application Layer เป็นชั้นที่ผู้ใช้งานสามารถมีปฏิสัมพันธ์กับตัวอุปกรณ์ ซึ่งชั้นนี้จะมีการให้บริการในด้านเฉพาะตามแต่ละหน้าที่ของแต่ละ Application ให้กับผู้ใช้งาน ดังที่เห็นที่มีแพร่หลายมากในปัจจุบันเช่น บ้านอัจฉริยะ (Smart Home) ที่เมื่อผู้ใช้ควบคุม/ออกคำสั่งผ่านอุปกรณ์โทรศัพท์เคลื่อนที่ไปยังอุปกรณ์ Thermostat

2.5 เทคโนโลยี Raspberry Pi

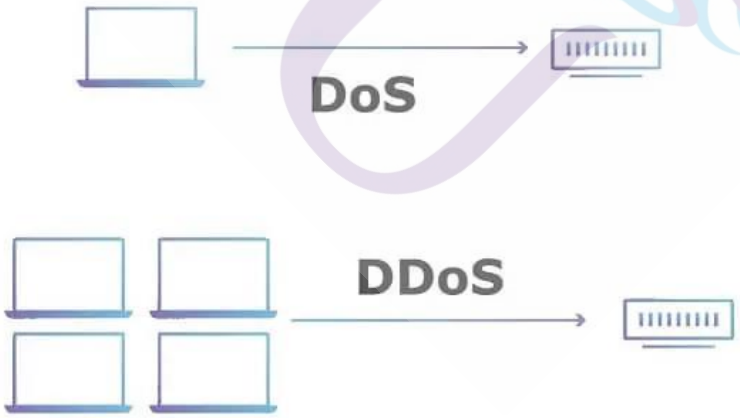
Raspberry Pi เป็นอุปกรณ์ในตระกูล Microcontroller หรือ Single board ขนาดเล็ก (SBCs) ที่ถูกพัฒนาที่ประเทศอังกฤษ ภายใต้องค์กรที่ชื่อว่า Raspberry Pi Foundation ร่วมกับ Boardcom ซึ่งมีเป้าหมายสำหรับสอนเกี่ยวกับวิทยาการคอมพิวเตอร์ภายในโรงเรียนและสำหรับกลุ่มประเทศด้านอุตสาหกรรม ซึ่งกลายเป็นที่นิยมมากในปัจจุบัน โดยเฉพาะในงานที่เกี่ยวกับด้านหุ่นยนต์ดังภาพที่ 2.8



ภาพที่ 2.9 ส่วนประกอบต่างๆของ Raspberry Pi

2.6 การโจมตีแบบ DDoS

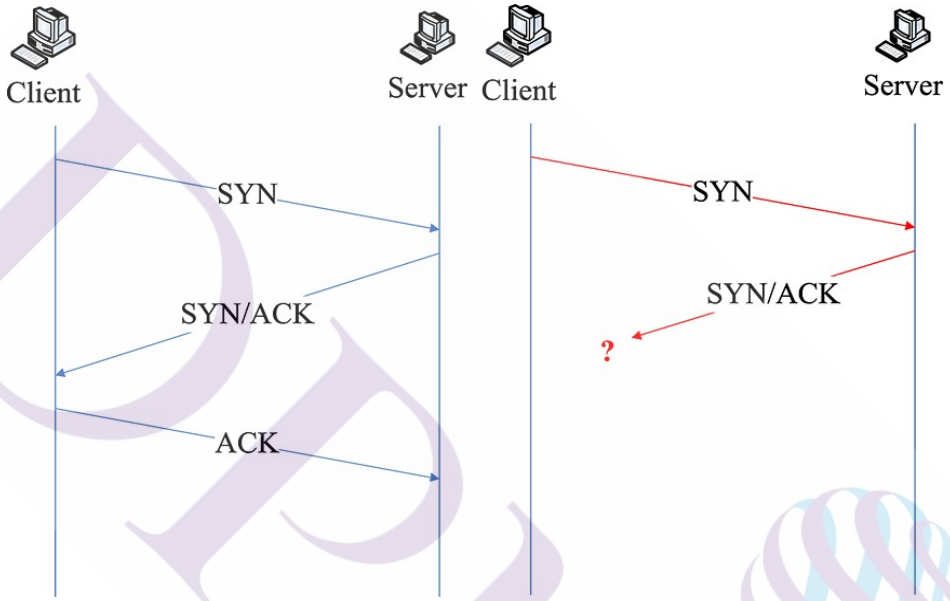
DDoS หรือ Distributed Denial of Service เป็นการโจมตีทางไซเบอร์รูปแบบหนึ่งที่ผู้โจมตีจะทำการส่ง Traffic หรือคำขอเข้าถึงข้อมูลจากต้นทางในหลายๆที่ไปยังเป้าหมายที่ต้องการโจมตีในระยะเวลาพร้อมๆกัน ซึ่งทำให้ปริมาณ Traffic สูงเกินกว่าที่เป้าหมายจะสามารถรองรับได้ ส่งผลทำให้เป้าหมายไม่สามารถทำงานหรือให้บริการต่อได้ โดยข้อแตกต่างระหว่าง DDoS กับ DoS ต่างกันตรงที่ Dos มีการโจมตีจากแหล่งต้นทางจากแหล่งเดียวดังภาพที่ 2.9



ภาพที่ 2.10 ความแตกต่างระหว่าง DDoS VS DoS

2.7 การโจมตีแบบ TCP SYN Flood Attack

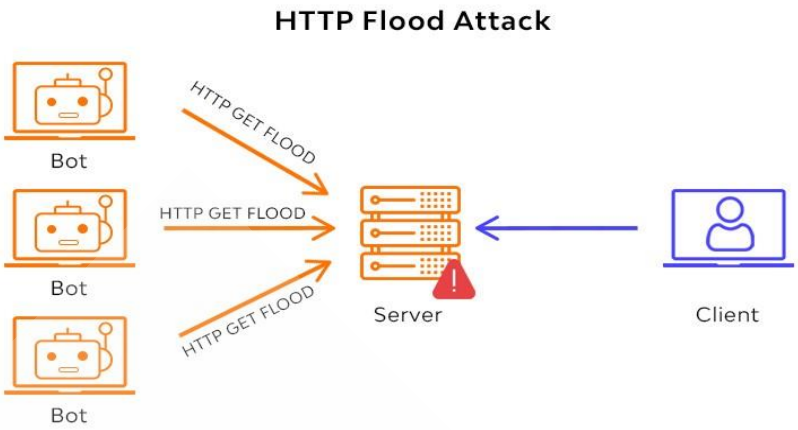
SYN Flooding เป็นการโจมตีโดยการส่ง TCP Packet ไปยังเป้าหมายเหมือนขั้นตอนการเริ่มต้นร้องขอการเชื่อมต่อ (Negotiation) ตามรูปแบบการเชื่อมต่อแบบ TCP ปกติแต่เมื่อเครื่องเป้าหมายทำการตอบกลับด้วย SYN-ACK เครื่องที่ทำการโจมตีจะไม่ทำการส่งข้อมูลกลับไป ทำให้เครื่องเป้าหมายค้างอยู่ในสถานะ Half-Open ซึ่งหากมีการส่ง Packet ไปยังเครื่องเป้าหมายเป็นจำนวนมากจะทำให้เครื่องเป้าหมายให้บริการช้าลงหรือไม่สามารถให้บริการได้ [1] ดังภาพที่ 2.10



ภาพที่ 2.11 ภาพแสดงขั้นตอนการเชื่อมต่อแบบ TCP ปกติและแบบถูกโจมตี

2.8 การโจมตีแบบ HTTP GET Flood Attack

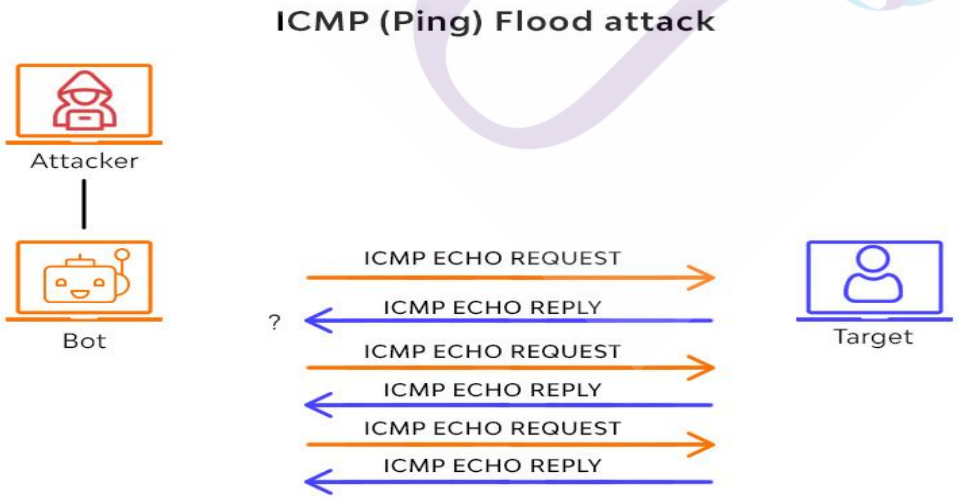
HTTP GET Flooding เป็นรูปแบบการโจมตีที่เกิดขึ้นในชั้น Application Layer โดยเครื่องที่โจมตีจะทำการส่ง GET Request ไปยังเครื่องเป้าหมาย เมื่อได้รับ Request เครื่องเป้าหมายจะเข้าสู่ขั้นตอน Resource Allocation เพื่อจองทรัพยากรสำหรับการประมวลผล แต่หากเครื่องเป้าหมายได้รับจำนวน Request จำนวนมากๆจะทำให้ทรัพยากรของเครื่องเป้าหมายในขั้นตอน Resource Allocation เต็มและทำให้เครื่องเป้าหมายไม่สามารถให้บริการได้ [1] ดังภาพที่ 2.11



ภาพที่ 2.12 การโจมตีแบบ HTTP Get Flooding ไปยังเครื่องเป้าหมาย

2.9 การโจมตีแบบ ICMP Flood Attack

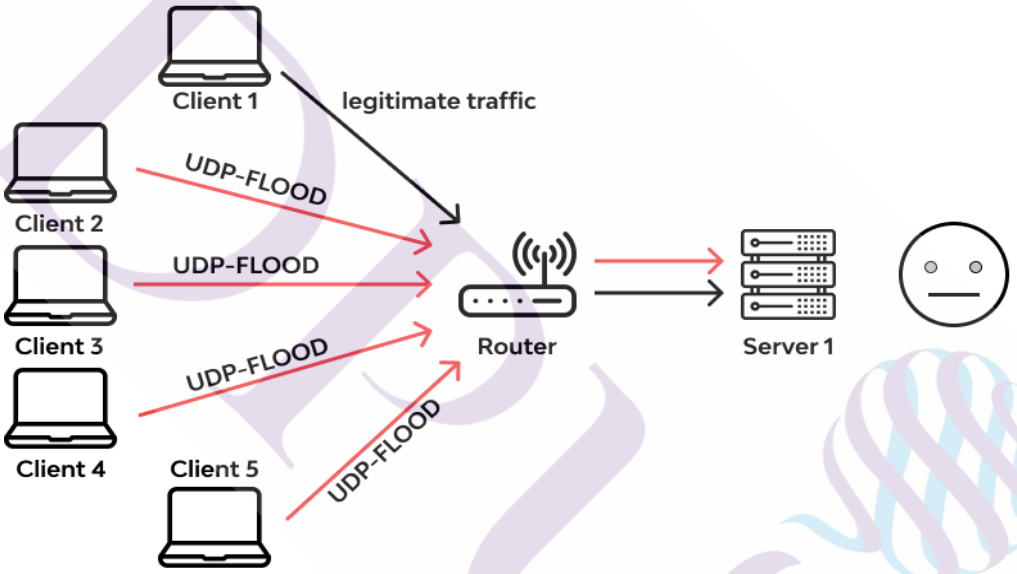
ICMP Flooding เป็นรูปแบบการโจมตีที่ผู้โจมตีทำการส่ง ICMP Echo Request (Ping) จำนวนมากไปยังเครื่องเป้าหมาย ซึ่งในขั้นตอนการส่ง ICMP Echo Request แบบปกติไปยังเครื่องปลายทาง 1 ครั้งเครื่องต้นทางจะได้รับการตอบ ICMP Response กลับมา 1 ครั้งซึ่งหากการโจมตีไปยังเครื่องเป้าหมายในปริมาณมาก การโจมตีในรูปแบบนี้จะส่งผลให้ Traffic ของเครือข่ายเต็มและทำให้เครื่องอื่นๆภายในเครือข่ายไม่สามารถเข้าถึงเครื่องที่ถูกโจมตีได้ [1] ดังภาพที่ 2.12



ภาพที่ 2.13 การโจมตีแบบ ICMP Flooding ไปยังเครื่องเป้าหมาย

2.10 การโจมตีแบบ UDP Flood Attack

UDP Flooding เป็นรูปแบบการโจมตีที่ผู้โจมตีทำการส่ง IP Packet ที่มีส่วนประกอบของ User Datagram Protocol (UDP) ไปยังเครื่องเป้าหมายในลักษณะ Random Ports เมื่อเครื่องเป้าหมายได้รับ Packet เครื่องเป้าหมายจะทำการประมวลที่เกี่ยวกับ UDP ของผู้ใช้งานนั้น ซึ่งเมื่อไม่พบหรือไม่สามรถประมวลผลได้ เครื่องเป้าหมายจะตอบ Destination Unreachable Packet กลับไปยังเครื่องต้นทาง ซึ่งหากเครื่องเป้าหมายได้รับ Packet ที่ปริมาณที่มากจะส่งผลให้ Traffic ของเครือข่ายเต็มทำให้เครื่องอื่นๆภายในเครือข่ายไม่สามารถเข้าถึงเครื่องที่ถูกโจมตีได้ [1] ดังภาพที่ 2.13



ภาพที่ 2.14 การโจมตีแบบ UDP Flooding ไปยังเครื่องเป้าหมาย

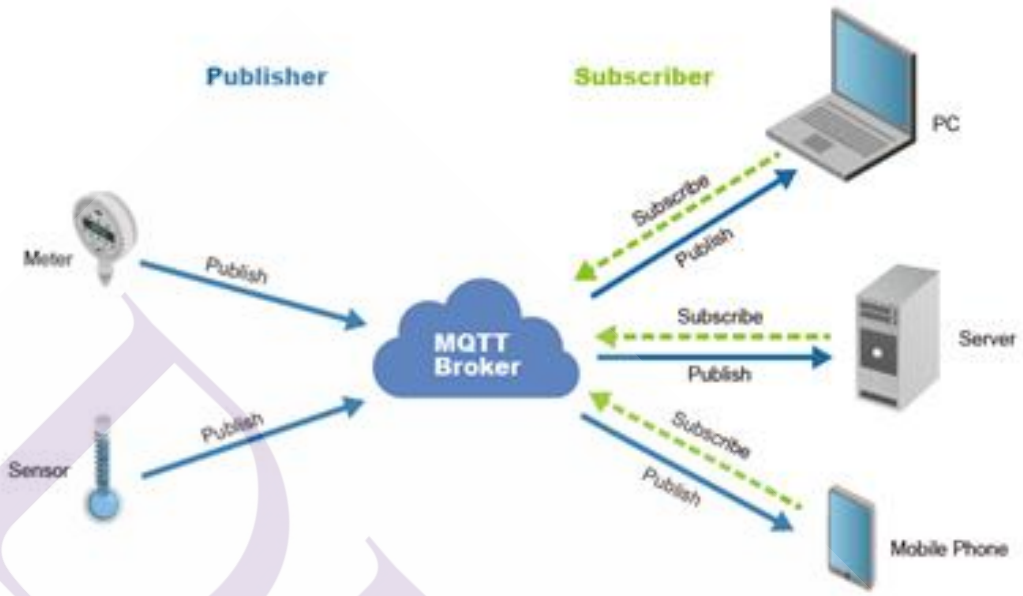
2.11 ทฤษฎี MQTT Protocol

MQTT (Message Queue Telemetry Transport) เป็น Protocol ที่ออกแบบมา เพื่อการเชื่อมต่อแบบ M2M (Machine-to-Machine) หรืออุปกรณ์ติดต่อสื่อสารกับอุปกรณ์ซึ่งเป็นส่วนหนึ่งของเทคโนโลยี IoT มีลักษณะการทำงานแบบ Clients and Broker Network ถูกออกแบบมาให้ทำงานแบบ Real-Time สำหรับปริมาณข้อมูลที่มีขนาดเล็ก ทำให้ใช้พลังงานต่ำถูกพัฒนาต่อมาจาก TCP/IP Protocol ส่วนประกอบของ MQTT ประกอบด้วย 4 ส่วนหลักได้แก่

1. Broker (Server) คือตัวกลางในการรับส่งข้อมูลระหว่าง Publisher และ Subscriber
2. Client (Publisher) คือตัวส่งข้อมูลผ่าน Broker ตาม Topic ที่ตกลงไว้

3. Client (Subscriber) คือตัวรับข้อมูลผ่าน Broker ตาม Topic ที่ตกลงไว้

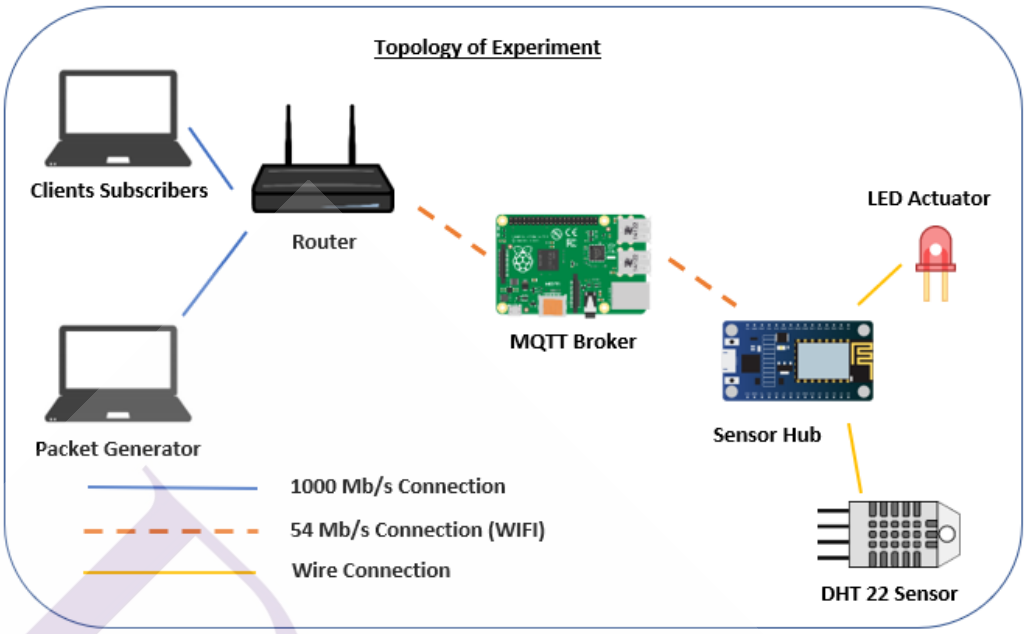
4. Topic คือหัวเรื่องที่รับส่งข้อมูลระหว่าง Publisher และ Subscriber ดังภาพที่ 2.14



ภาพที่ 2.15 ส่วนประกอบทั้งหมดของ MQTT Protocol

บทที่ 3
การสร้างระบบ IoT และ ชุดคำสั่งการโจมตี

งานวิจัยนี้ได้พัฒนาสร้างระบบจำลองการทดสอบออกมาในลักษณะ IoT แบบ Smart Home ที่ปัจจุบันมีการประยุกต์ใช้และสามารถสร้างขึ้นเอง ซึ่งมีการใช้งานที่แผร่หลายในปัจจุบัน เป็นรูปแบบหนึ่งของระบบ IoT ที่ไม่ขึ้นอยู่กับยี่ห้อของผลิตภัณฑ์ในท้องตลาด ภายในระบบจำลองใช้ Raspberry Pi Model B 8 GB ที่ติดตั้ง Raspbian OS เป็นระบบปฏิบัติการ พร้อมทั้งติดตั้ง Mosquitto Broker สำหรับรับส่งข้อมูลระหว่าง Publisher และ Subscriber ผ่าน MQTT Protocol พร้อมทั้งติดตั้ง FLASK ที่เป็น Web Framework ของ Python ที่รองรับ WSGI นอกจากนั้นยังติดตั้ง Paho-Mqtt Library ช่วยในการรับส่ง Message ระหว่าง Web และ MQTT Broker และติดตั้ง SocketIO Library ที่ช่วยทำให้ Web สามารถทำงานแบบ Asynchronous ส่วนในฝั่งของ Sensor Hub ใช้บอร์ด ESP 8266 ที่ติดตั้ง PubSubClient Library สำหรับรับส่ง Message กับ MQTT Broker ผ่าน MQTT Protocol และติดตั้ง DHT 22 Library เพื่อให้สามารถอ่านค่าอุณหภูมิและความชื้นจาก DHT 22 Sensor โดยมี LED Actuator และ DHT 22 Sensor เชื่อมต่อกับบอร์ด ESP 8266 ผ่าน GPIO Wire Connection และบอร์ด ESP 8266 เชื่อมต่อกับ MQTT Broker ผ่าน WIFI Connection การเชื่อมต่อระหว่าง MQTT Broker และอุปกรณ์ High Speed Router เชื่อมต่อกันผ่าน WIFI Connection โดยการเชื่อมต่อแบบ WIFI Connection ทั้งหมดในระบบจำลองใช้ความเร็วที่ 54 Mb/s ส่วนของ Packet Generator หรืออุปกรณ์สร้างชุดการโจมตีใช้เครื่องคอมพิวเตอร์ที่ติดตั้ง Kali Linux เป็นระบบปฏิบัติการ และติดตั้ง HPing3 เป็นเครื่องสำหรับสร้างชุดการโจมตีในรูปแบบ TCP SYN Flood Attack, ICMP Flood Attack และ UDP Flood Attack พร้อมทั้งติดตั้ง WRK เป็นเครื่องมือสำหรับสร้างการโจมตีในรูปแบบ HTTP GET Flood Attack ดังภาพที่ 3.1



ภาพที่ 3.1 ภาพรวมของระบบ IoT และชุดการสร้างการโจมตี

3.1 ระเบียบวิธีวิจัย

3.1.1 ศึกษาทฤษฎีต่าง ๆ ของระบบ Internet of Things (IoT) แล้วนำเอาทฤษฎีที่เกี่ยวข้องมาประยุกต์ใช้กับงานวิจัยที่นำเสนอ

3.1.2 ศึกษาการทำงานของอุปกรณ์ที่ใช้ในงานวิจัย

ทำการศึกษาทฎษฎีต่าง ๆ ของตัวอุปกรณ์ที่ใช้ในระบบ เช่น ไมโครโพรเซสเซอร์ Raspberry Pi, ไมโครคอนโทรลเลอร์ ESP 8266, เซ็นเซอร์วัดอุณหภูมิและความชื้น DHT22 เพื่อให้ทราบถึงวิธีการใช้งานและเพื่อนำมาประยุกต์ใช้ให้มีประสิทธิภาพมากที่สุด

3.1.3 ศึกษาทฤษฎี MQTT Protocol

ทำการศึกษาการทำงานส่วนในประกอบต่าง ๆ ภายในโครงสร้าง MQTT เพื่อนำมากำหนดรูปแบบการรับส่งข้อมูลภายในระบบ IoT ให้มีประสิทธิภาพสูงสุด

3.1.4 ศึกษาการพัฒนา Web Application บน Raspberry Pi

ทำการศึกษาหลักการ แนวคิด และวิธีการพัฒนา Web Application ด้วย Python และนำมาติดตั้งบน Raspberry Pi เพื่อใช้สำหรับควบคุมการสั่งการไปยัง Sensor ได้อย่างมีประสิทธิภาพ

3.1.5 ศึกษาทฤษฎีรูปแบบการโจมตีแบบ DDoS

ทำการศึกษารูปแบบการโจมตี ความแตกต่างของแต่ละรูปแบบการโจมตี วิธีการโจมตีและชุดคำสั่งสำหรับสร้างการโจมตี เพื่อให้การโจมตีมีประสิทธิภาพสูงสุด

3.1.6 ออกแบบระบบและโครงสร้างการเชื่อมต่อ

ทำการออกแบบระบบ IoT ทั้งในส่วนของ ซอฟต์แวร์และฮาดแวร์ที่มีอยู่ภายในระบบทั้งหมด ซึ่งจะทำให้เห็นภาพรวมทั้งหมดของระบบภายในงานวิจัยนี้

3.1.7 พัฒนาโปรแกรม

ทำการพัฒนาโปรแกรมโดยแบ่งเป็น 4 ส่วนคือส่วนที่เกี่ยวของกับ Web Application ของระบบที่ติดตั้งบน Raspberry Pi ส่วนของอุปกรณ์ IoT ที่เชื่อมต่อการทำงานระหว่างเซนเซอร์ที่ติดตั้งบนบอร์ด ESP 8266 ส่วนที่เกี่ยวกับการรับส่งข้อมูลผ่าน MQTT Protocol และส่วนที่เกี่ยวกับการสร้างชุดคำสั่งสำหรับการโจมตีในรูปแบบต่างๆ ไปยังระบบ IoT

3.1.8 ทดสอบอุปกรณ์และอัลกอริทึมที่ใช้ในระบบ

ทำการทดสอบอุปกรณ์ อัลกอริทึม การรับส่งข้อมูล และการโจมตีตามที่ได้ศึกษาข้อมูลมา เพื่อยืนยันผลลัพท์และวิธีการที่สามารถใช้ได้จริง

3.1.9 ทำการทดสอบการโจมตี

การโจมตีไปยังระบบ IoT ที่พัฒนาขึ้นตามวิธีการและขั้นตอนที่ได้ศึกษา และออกแบบการทดสอบไว้ เพื่อให้ได้ผลลัพท์ของระบบที่เกิดจากการถูกโจมตี

3.1.10 วิเคราะห์และสรุปผล

หลังจากที่ทำการโจมตี และได้ผลลัพท์ที่เกิดจากการโจมตี นำข้อมูลที่ได้มาวิเคราะห์ว่าสอดคล้องกับสมมุติฐานตามทฤษฎีหรือไม่ เมื่อวิเคราะห์เสร็จแล้วก็ทำการสรุปผลลัพท์ที่ได้จากการทำการวิจัย

3.1.11 รวบรวมข้อมูลที่ได้จัดทำสารนิพนธ์

ทำการรวบรวมข้อมูลทั้งหมดที่เกิดจากการวิจัย จากแนวคิดและหลักการทางทฤษฎีเพื่อจัดทำสารนิพนธ์

3.2 แนวทางการวิจัยและพัฒนาระบบ

เทคโนโลยี IoT (Internet of Thing) เป็นเทคโนโลยีที่ทำให้อุปกรณ์ฮาดแวร์ ซอฟต์แวร์ผสานการทำงานร่วมกับสภาพแวดล้อมภายนอกรอบข้าง และเชื่อมโยงกันเข้าผ่านอินเตอร์เน็ต ซึ่งทำให้เกิดการสร้างชุดข้อมูลปริมาณมหาศาล ช่วนให้การทำงานต่าง ๆ หรือกระบวนการของมนุษย์ในปัจจุบันสะดวกสบายขึ้น ซึ่งในงานวิจัยนี้ได้นำเอาเทคโนโลยี IoT ในรูปแบบของบ้านอัจฉริยะ (Smart Home)มาใช้จำลองระบบสำหรับรองรับการถูกโจมตี เพื่อนำผลที่ได้มาวิเคราะห์ถึงผลกระทบ และลักษณะการทำงานที่ผิดไปจากเดิมในขณะที่เกิดการโจมตี

โดยแนวทางการวิจัยนี้ได้แนวคิดมาจากปัญหาในปัจจุบันว่าเทคโนโลยีต่างๆ ในด้านสารสนเทศมีการพัฒนาก้าวหน้าไปอย่างมากและภัยคุกคามทางไซเบอร์ที่เกิดในปัจจุบันก็เกิดเพิ่มมากขึ้น และมีความรุนแรงที่สูงตามไปด้วย ในด้านของเทคโนโลยี IoT เองก็มีพัฒนาขึ้นอย่างกว้างขวางและรวดเร็ว ดังที่เห็นได้ในปัจจุบัน แต่ด้านความมั่นคงปลอดภัยของตัวเทคโนโลยี IoT เองกลับพัฒนาได้ช้ากว่าตัวเทคโนโลยีอย่างมาก ทำให้ตัวเทคโนโลยี IoT อาจเป็นต้นเหตุหรือภัยใกล้ตัวที่ทำให้เกิดความเสียหายแก่เราได้ จึงเป็นที่มาของงานวิจัยนี้เพื่อให้ทราบถึงผลลัพท์หรือผลกระทบที่เกิดจากการถูกโจมตี เพื่อทราบถึงระดับความรุนแรง และขอบเขตความสามารถในการกู้คืนตัวเองของระบบ IoT ที่สามารถพัฒนาได้ด้วยตนเอง

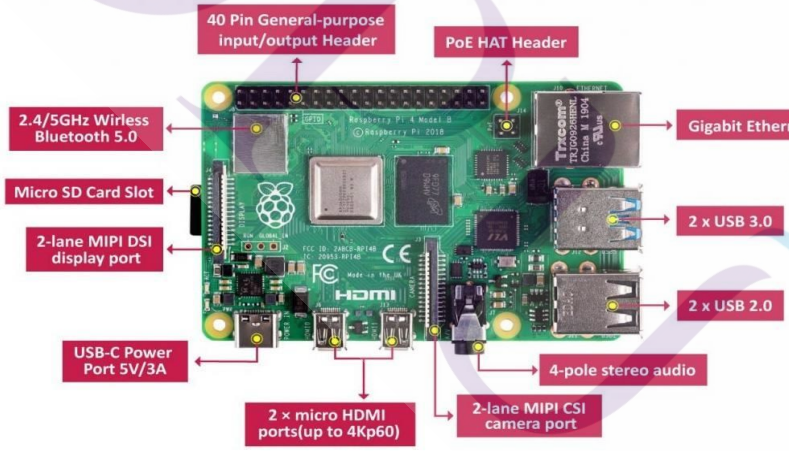
3.3 ไมโครโพรเซสเซอร์ Raspberry Pi

ในระบบ IoT ของงานวิจัยนี้ใช้อุปกรณ์ไมโครโพรเซสเซอร์ Raspberry Pi 4 Model B RAM 8 GB ซึ่งให้ประสิทธิภาพการทำงานระดับ Desktop เทียบเท่า x86 PC Systems ที่รองรับการพัมนาแบบ Open Source ทำให้ผู้ใช้งานสามารถพัฒนาต่อยอดได้ทั้งในด้านของ Hardware และ Software

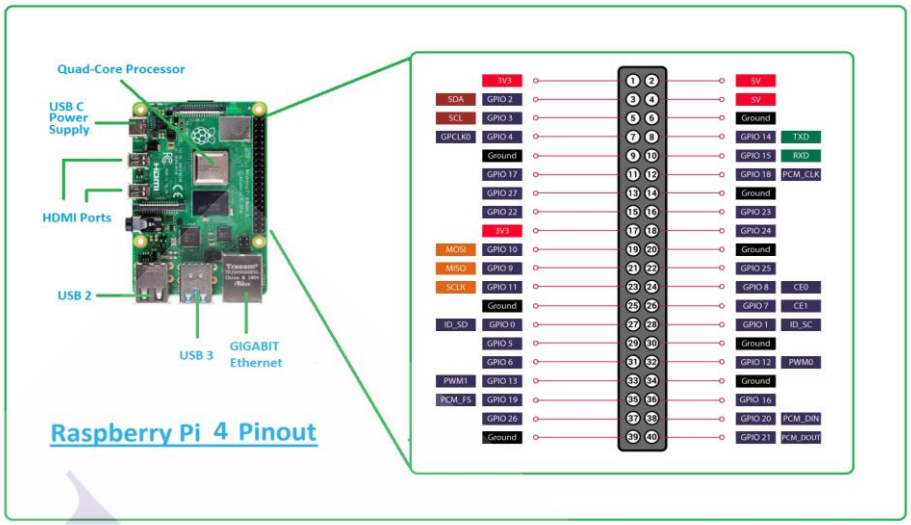
ภาพที่ 3.2 ไมโครโพรเซสเซอร์ Raspberry Pi 4 Model B

Layout และ Pin out ของไมโครโพรเซสเซอร์ Raspberri Pi 4 Model B

1. GPIO 40 Pins
2. PoE HAT Header
3. 2.4 / 5 GHz Wireless & Bluetooth 5.0
4. Gigabit Ethernet
5. Micro SD Card Slot
6. Display Port
7. USB 3.0
8. USB 2.0
9. USB-C Power
10. Micro HDMI Port
11. Camera Port
12. Stereo Audio



ภาพที่ 3.3 Layout ของ Raspberry Pi 4 Model B



ภาพที่ 3.4 Pinout ของ Raspberry Pi 4 Model B

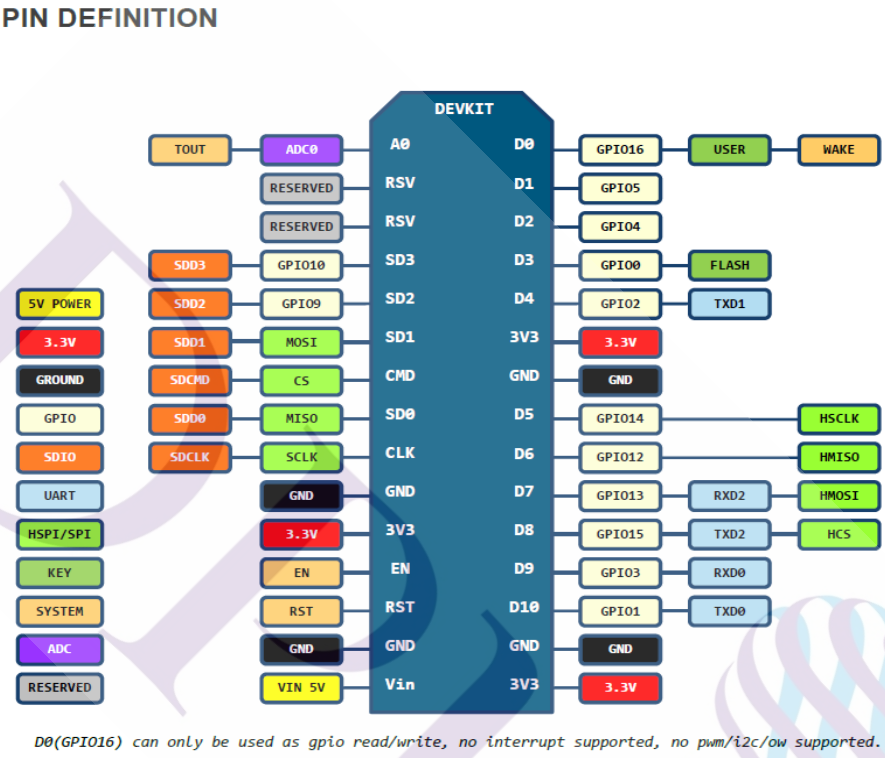
3.4 ไมโครคอนโทรลเลอร์ NodeMCU ESP 8266

ในระบบ IoT ของงานวิจัยนี้ใช้อุปกรณ์ไมโครคอนโทรลเลอร์ NodeMCU ESP 8266 ซึ่งเป็นบอร์ดคอนโทรลเลอร์ที่มีลักษณะการทำงานด้วยคำสั่งภาษา C คล้ายกับบอร์ด Arduino แต่มีลักษณะที่พิเศษกว่าตรง สามารถเชื่อมต่อกับ WIFI ได้ รองรับคำสั่ง Deep Sleep ในการประหยัดพลังงาน และสามารถ Wake Up กลับมาพร้อมใช้งานด้วยเวลาน้อยกว่า 2 มิลลิวินาที พร้อมพอร์ต micro USB เพื่อรองรับการเชื่อมต่อเข้ากับเครื่องคอมพิวเตอร์เพื่อพัฒนาโปรแกรม

ภาพที่ 3.5 ไมโครคอลโทรลเลอร์ Node MCU ESP 8266

Layout และ Pin out ของ Node MCU ESP 8266

1. Micro USB
2. Flash Button
3. Reset Button



ภาพที่ 3.6 Pin out ของ Node MCU ESP 8266

3.5 เซนเซอร์วัดอุณหภูมิและความชื้น (DHT22)

ในระบบ IoT ของงานวิจัยนี้ใช้เซนเซอร์สำหรับวัดอุณภูมิ และความชื้นในอากาศชนิด DHT22 ซึ่งมาราคาถูกใช้งานง่าย ให้ผล Output ออกมาเป็นค่า Digital ใช้พลังงานขนาด 3.3 - 6 โวลต์

ภาพที่ 3.7 เซนเซอร์วัดอุณหภูมิและความชื้น DHT22

คุณสมบัติของเซนเซอร์วัดอุณหภูมิและความชื้น DHT22

1. ย่านวัดความชื้น 0 - 100%RH (Relative Humidity)
2. ค่าความแม่นยำ +/- 2%RH
3. ความไวและความละเอียดต่อความชื้น 0.1%RH
4. Output เป็นค่า Digital
5. อ่านค่าสัญญาณทุก 2 วินาที

3.6 แอคชูเอเตอร์ LED

ในระบบ IoT ของงานวิจัยนี้ใช้แอคชูเอเตอร์ LED แบบ 2 Pin ขนาด 5 มิลลิเมตร ซึ่งใช้พลังงานต่ำ มีอายุการใช้งานยาว และให้อุณหภูมิต่ำขณะเปล่งแสงทำให้ส่งผลกระทบต่อเซนเซอร์วัดอุณหภูมิและความชื้นน้อย

คุณสมบัติของแอคชูเอเตอร์ LED

1. ใช้แรงดันไฟฟ้า 1.9 - 2.4 โวลต์
2. ความเข้มของการส่องสว่าง 500 - 2000 mcd
3. ทำงานที่กระแสไฟ 20 mA

ภาพที่ 3.8 แอคชูเอเตอร์ LED 5 mm

3.7 เราเตอร์ TP Link Archer C64 AC1200 Wireless

ในระบบ IoT ของงานวิจัยนี้ใช้เราเตอร์ Archer C64 ใช้การเชื่อมต่อระหว่าง Node IoT กับ MQTT Broker และ เชื่อมต่อ Clients Subscriber, Packet Generator กับ MQTT Broker

คุณสมบัติของ Archer C64

1. ความเร็ว Wireless สูงสุดที่ย่าน 5 GHz อยู่ที่ 867 Mbps และย่าน 2.4 GHz ความเร็วสูงสุดอยู่ที่ 400 Mbps

2. หน่วยประมวลผล 1.2 GHz

3. รองรับการเชื่อมแบบสายผ่าน Gigabit LAN Port ที่ความเร็วสูงสุด 100 Mbps

ภาพที่ 3.9 เราเตอร์ TP Link Archer C64

3.8 Hping3

งานวิจัยนี้ใช้เครื่องสำหรับสร้างชุดการโจมตีที่เรียกว่า Hping3 ซึ่งเป็นเครื่องในด้านเครือข่ายที่มีความสามารถสร้างชุด Packet ในรูปแบบ ICMP, UDP และ TCP ที่สามารถปรับเองได้ พร้อมแสดงการตอบกลับของเป้าหมาย เป็นเครื่องมือที่นิยมใช้ในการทดสอบ Firewall Rules, ใช้ในกระบวนการสแกน Port, ใช้ทดสอบประสิทธิภาพของเครือข่ายด้วย Protocol แบบต่าง ๆ สามารถใช้งานได้สะดวกทำงานบนระบบปฏิบัติการ Kali Linux

รูปแบบการทำงานหลักของ Hping3

1. RAW IP Mode
2. ICMP Mode
3. UDP Mode
4. SCAN Mode
5. Listen Mode

```
root@kali:~# hping3 -h
usage: hping3 host [options]
  -h  --help      show this help
  -v  --version   show version
  -c  --count     packet count
  -i  --interval  wait (uX for X microseconds, for example -i u1000)
      --fast      alias for -i u10000 (10 packets for second)
      --faster    alias for -i u1000 (100 packets for second)
      --flood      sent packets as fast as possible. Don't show replies.
  -n  --numeric   numeric output
  -q  --quiet     quiet
  -I  --interface interface name (otherwise default routing interface)
  -V  --verbose   verbose mode
  -D  --debug     debugging info
  -z  --bind      bind ctrl+z to ttl           (default to dst port)
  -Z  --unbind    unbind ctrl+z
      --beep      beep for every matching packet received
Mode
  default mode     TCP
  -0  --rawip      RAW IP mode
  -1  --icmp       ICMP mode
  -2  --udp        UDP mode
  -8  --scan       SCAN mode.
                   Example: hping --scan 1-30,70-90 -S www.target.host
  -9  --listen     listen mode
IP
  -a  --spoof      spoof source address
  --rand-dest      random destionation address mode. see the man.
  --rand-source    random source address mode. see the man.
  -t  --ttl        ttl (default 64)
  -N  --id         id (default random)
  -W  --winid      use win* id byte ordering
  -r  --rel        relativize id field          (to estimate host traffic)
  -f  --frag       split packets in more frag.  (may pass weak acl)
  -x  --morefrag   set more fragments flag
  -y  --dontfrag   set don't fragment flag
```

ภาพที่ 3.10 รายการแสดงค่าการปรับแต่งของ Hping3

โดยชุดคำสั่งสำหรับการสร้างการโจมตีในงานวิจัยนี้ มีดังนี้

TCP SYN Flood Attack

“ sudo hping3 -d 120 -S -V -p 8181 192.168.1.103 --flood --rand-source ”

-d = Data Size 120

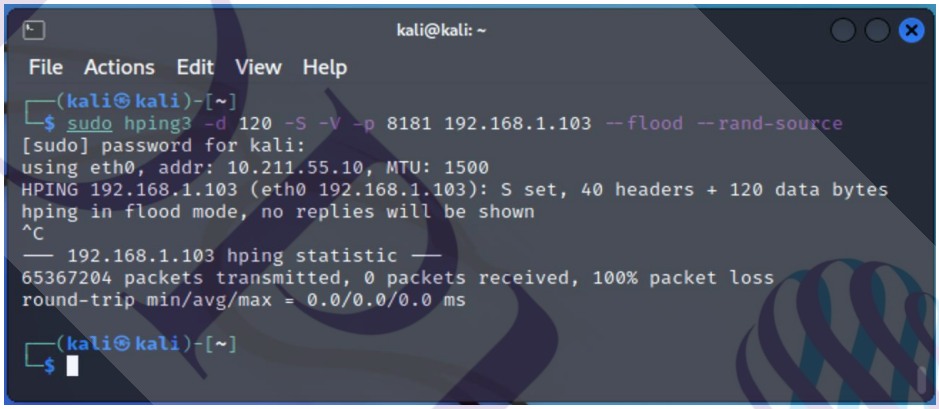
-S = SYN Packet (TCP is Default Mode)

-V = Verbose Mode

-p = Port เครื่องปลายทาง

--flood = ส่ง Packet ไปยังเครื่องปลายทางเร็วที่สุดเท่าที่จะทำได้ โดยไม่สนใจการตอบกลับ

--rand-source = ปลอมหมายเลข IP Address เครื่องต้นทาง



ภาพที่ 3.11 ชุดคำสั่งสำหรับสร้างการโจมแบบ TCP SYN Flood Attack

ICMP Flood Attack

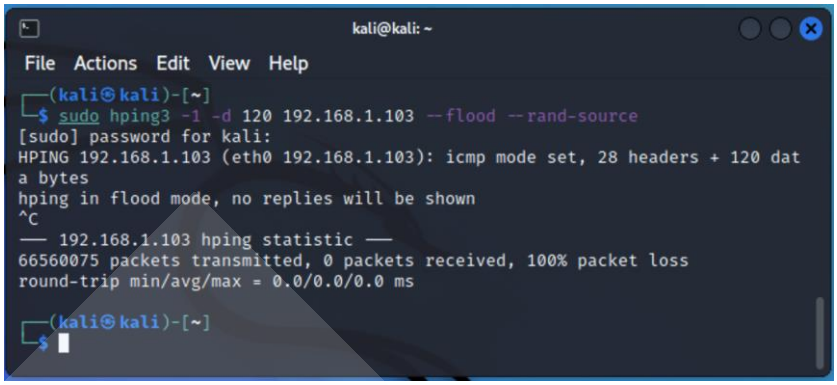
“ sudo hping3 -1 -d 120 192.168.1.103 --flood --rand-source ”

-1 = ICMP Packet (ICMP Mode)

-d = Data Size 120

--flood = ส่ง Packet ไปยังเครื่องปลายทางเร็วที่สุดเท่าที่จะทำได้ โดยไม่สนใจการตอบกลับ

--rand-source = ปลอมหมายเลข IP Address เครื่องต้นทาง



ภาพที่ 3.12 ชุดคำสั่งสำหรับสร้างการโจมแบบ ICMP Flood Attack

UDP Flood Attack

“ sudo hping3 --udp -d 120 -p 8181 192.168.1.103 --flood --rand-source ”

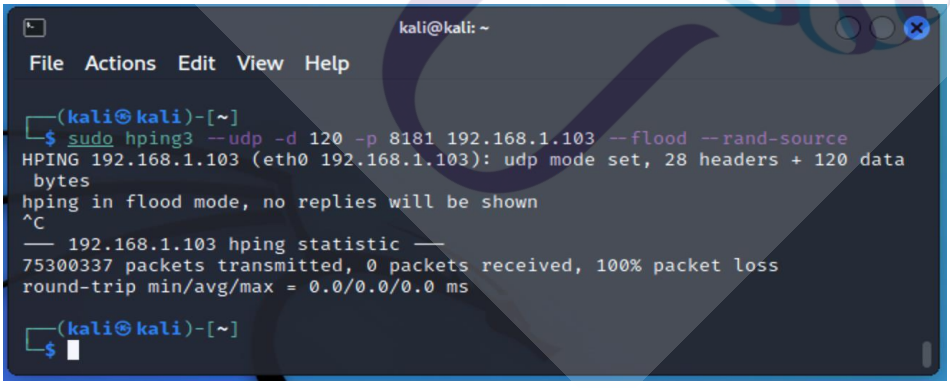
--udp = UDP Packet (UDP Mode)

-d = Data Size 120

-p = Port เครื่องปลายทาง

--flood = ส่ง Packet ไปยังเครื่องปลายทางเร็วที่สุดเท่าที่จะทำได้ โดยไม่สนใจการตอบกลับ

--rand-source = ปลอมหมายเลข IP Address เครื่องต้นทาง



ภาพที่ 3.13 ชุดคำสั่งสำหรับสร้างการโจมแบบ UDP Flood Attack

3.9 WRK

งานวิจัยนี้ใช้เครื่องมือ WRK เป็นเครื่องมือที่ใช้วัดประสิทธิภาพของ HTTP ซึ่งสร้างชุด Request Packet ปริมาณมากสำหรับส่งไปยังเป้าหมาย และเป็นเครื่องมือที่รองรับการสร้างชุดคำสั่งที่สนับสนุนการทำงานแบบ Multi-Core และ Multithread สามารถติดตั้งได้ง่าย และสามารถสร้างรายงานการวิเคราะห์จาก Response ได้ โดยชุดคำสั่งสำหรับการสร้างการโจมตีในงานวิจัยนี้ มีดังนี้

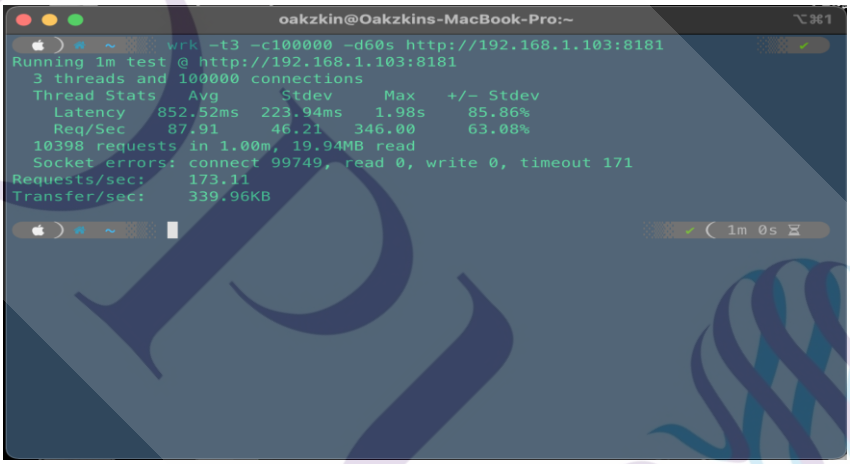
HTTP GET Flood Attack

“ wrk -t3 -c100000 -d60s http://192.168.1.103:8181 ”

-t3 = ทำงานแบบ 3 Thread

- c = จำนวน Request ทั้งหมดที่ต้องการ

-d = ระยะเวลาทั้งหมดของการส่ง Request



ภาพที่ 3.14 ชุดคำสั่งสำหรับสร้างการโจมแบบ HTTP GET Flood Attack

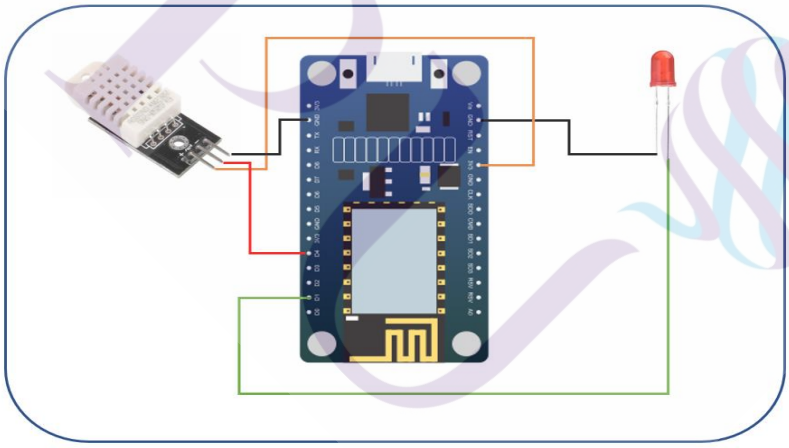
3.10 การเชื่อมต่ออุปกรณ์

เนื่องจากการสร้างระบบ IoT จะต้องมีการเชื่อมต่อเซนเซอร์เข้ากับไมโครคอนโทรลเลอร์ เพื่อจะอ่านค่าจากอุปกรณ์เซนเซอร์ DHT22 และรับคำสั่งสำหรับสั่งงานแอคชูเอเตอร์ LED เพื่อให้พร้อมสำหรับการรับส่งข้อมูลกับ MQTT Broker ผ่าน MQTT Protocol

ตารางการเชื่อมต่ออุปกรณ์เซนเซอร์แอคชูเอเตอร์เข้ากับบอร์ด ESP 8266

เซนเซอร์วัดอุณหภูมิและความชื้น	**บอร์ด ESP 8266**
+	D4 (GPIO02)
Out	3V3
-	GND

แอคชูเอเตอร์ LED	**บอร์ด ESP 8266**
+	D1 (GPIO5)
-	GND



ภาพที่ 3.15 การต่อเซนเซอร์ DHT22 และ แอคชูเอเตอร์ LED เข้ากับบอร์ด ESP 8266

3.11 ชุดคำสั่งการทำงานบนบอร์ด ESP 8266

ในการออกแบบชุดคำสั่งการควบคุมและสั่งการทำงานของบอร์ด ESP 8266 นั้นได้เลือกพัฒนาโปรแกรมด้วยภาษา C เนื่องจากอุปกรณ์ฮาร์ดแวร์ ไมโครคอนโทรลเลอร์ที่เลือกใช้ในงานวิจัยนี้รองรับการพัฒนาชุดคำสั่งที่พัฒนาด้วยภาษา C

ชุดคำสั่งสำหรับไมโครคอนโทรลเลอร์ ESP 8266

```
//โหลด Library WIFI , PubSybClient และ DHT22
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"

//กำหนดค่า DHT22 ให้กับตัวแปร DHTTYPE
#define DHTTYPE DHT22

//กำหนดค่าสำหรับการเชื่อต่อม WIFI ให้กับตัวแปร
const char* ssid = "Takeshi_Factory_2.4G";
const char* password = "xxxxxxxxx";

//กำหนดค่าหมายเลข IP Address ของ MQTT Server
const char* mqtt_server = "192.168.1.103";


WiFiClient espClient;
PubSubClient client(espClient);

//กำหนดค่า PIN สำหรับ LED และ DHT22
const int ledGPIO5 = 5;

const int DHTPin = 2;

//กำหนดค่าชนิดของ DHT22
DHT dht(2, DHTTYPE);

//กำหนดค่าเวลาและหน่วยเวลา
long now = millis();
long lastMeasure = 0;

//Function สำหรับการเชื่อมต่อบอร์ด ESP 8266 กับ WIFI Router
void setup_wifi() {
 delay(10);

 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
 }
 Serial.println("");
 Serial.print("WiFi connected - ESP IP address: ");
 Serial.println(WiFi.localIP());
}
```

```
//Function รับ Message จาก Subscriber ใน Topic ที่กำหนดมายัง ESP 8266
void callback(String topic, byte* message, unsigned int length) {
 Serial.print("Message arrived on topic: ");
 Serial.print(topic);
 Serial.print(". Message: ");
 String messageTemp;

 for (int i = 0; i < length; i++) {
  Serial.print((char)message[i]);
  messageTemp += (char)message[i];
 }
 Serial.println();
//ตรวจสอบค่าคำสั่งที่สั่งการมาเพื่อเปิด-ปิด LED
 if(topic=="esp8266/5"){
    Serial.print("Changing GPIO 5 to ");
    if(messageTemp == "1"){
     digitalWrite(ledGPIO5, HIGH);
     Serial.print("On");
    }
    else if(messageTemp == "0"){
     digitalWrite(ledGPIO5, LOW);
     Serial.print("Off");
    }
 }
 Serial.println();
}

//Function สำหรับ Reconnect ไปยัง MQTT Broker เมื่อการเชื่อมต่อเดิมถูกตัด
void reconnect() {
 /
 while (!client.connected()) {
  Serial.print("Attempting MQTT connection...");

  if (client.connect("ESP8266Client")) {
   Serial.println("Connected");
   client.subscribe("esp8266/5");
  } else {
   Serial.print("failed, rc=");
   Serial.print(client.state());
   Serial.println(" try again in 5 seconds");
```

```
    delay(5000);
   }
  }
}
void setup() {
 dht.begin();

 pinMode(ledGPIO5, OUTPUT);
 //กำหนดค่า Baud Rate สำหรับการดูผ่าน Serial Monitor
 Serial.begin(115200);
 setup_wifi();
 client.setServer(mqtt_server, 1883);
 client.setCallback(callback);
}

//เริ่มต้นการทำงาน
void loop() {
 if (!client.connected()) {
  reconnect();
 }
 if(!client.loop())

  client.connect("ESP8266Client");
 now = millis();

 if (now - lastMeasure > 10000) {
  lastMeasure = now;
  //อ่านค่าอุณหภูมิ ค่าชื้น จาก DHT22
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  //ตรวจสอบหากไม่สามารถอ่านค่าจาก DHT22 ได้
  if (isnan(h) || isnan(t) || isnan(f)) {
   Serial.println("Failed to read from DHT sensor!");
   return;
```

```
  }
//คำนวณค่าอุณหภูมิให้อยู่ในหน่วย Celsius
  float hic = dht.computeHeatIndex(t, h, false);
  static char temperatureTemp[7];
  dtostrf(hic, 6, 2, temperatureTemp);


  static char humidityTemp[7];
  dtostrf(h, 6, 2, humidityTemp);

  //Publish ค่าอุณหภูมิและความเชื่อมไปยัง Subscriber ผ่าน MQTT Broker
  client.publish("/esp8266/temperature", temperatureTemp);
  client.publish("/esp8266/humidity", humidityTemp);

  //โชว์ค่าอุณหภูมิและความชื้นบนหน้าจอ Serial Monitor
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.print(" *F\t Heat index: ");
  Serial.print(hic);
  Serial.println(" *C ");
 }
}
```

3.12 การติดตั้ง การกำหนดค่าของไมโครโพรเซสเซอร์ Raspberry Pi

การติดตั้ง Mosquitto Broker

“ sudo apt install -y mosquitto mosquitto-clients ”

“ sudo systemctl enable mosquitto.service ”

“ sudo vi /etc/mosquitto/mosquitto.conf ”

เพิ่มคำสั่ง

listen 1883

allow_anonymous true

การติดตั้ง Flask

“ sudo apt-get install python-pip python-flask git-core ”

“ sudo pip install flask ”

“ sudo pip install paho-mqtt ”

การติดตั้ง Web SocketIO

“ sudo pip install flask-socketio ”

3.13 พัฒนา Web Application ด้วย Python โดยใช้ Flask Web Framework

สร้างไฟล์ main.html

```
<!DOCTYPE html>
<head>
  <title>Raspberry Pi Web Server</title>
  <link rel="stylesheet" href="{{url_for('static', filename = 'css/bootstrap.min.css')}}"
crossorigin="anonymous">
  <link rel="stylesheet" href="{{url_for('static', filename = 'css/bootstrap-theme.min.css')}}"
crossorigin="anonymous">
  <script src="{{url_for('static', filename = 'js/bootstrap.min.js')}}"
crossorigin="anonymous"></script>
  <script src="{{url_for('static', filename = 'js/jquery-3.1.1.min.js')}}"
crossorigin="anonymous"></script>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script type="text/javascript" src="{{url_for('static', filename = 'js/socket.io.min.js')}}"></script>
  <script type="text/javascript" charset="utf-8">
    $(document).ready(function() {
      var socket = io.connect('http://' + document.domain + ':' + location.port);
      socket.on('connect', function() {
       socket.emit('my event', {data: connected!'});
      });
      socket.on('dht_temperature', function(msg) {
    var nDate = new Date();
       $('#readingsUpdated').text(nDate.getHours() + ':' + nDate.getMinutes() +
         'm:' + nDate.getSeconds() + 's').html();
       $('#temperature').text(msg.data).html();
      });
```

```
      socket.on('dht_humidity', function(msg) {
       $('#humidity').text(msg.data).html();
      });
    });
  </script>
</head>

<body>
  <h1>Raspberry Pi Web Server With MQTT</h1>
  {% for pin in pins %}
  <h2>{{ pins[pin].name }}
  {% if pins[pin].state == 'True' %}
    is currently <strong>on</strong></h2><div class="row"><div class="col-md-2">
    <a href="/esp8266/{{pin}}/0" class="btn btn-block btn-lg btn-default" role="button">Turn
off</a></div></div>
  {% else %}
    is currently <strong>off</strong></h2><div class="row"><div class="col-md-2">
    <a href="/esp8266/{{pin}}/1" class="btn btn-block btn-lg btn-primary" role="button">Turn
on</a></div></div>
  {% endif %}
  {% endfor %}
  <h3>DHT Readings (updated <span id="readingsUpdated"></span>)</h3>
  <h3>Temperature: <span id="temperature"></span>ºC</h3>
  <h3>Humidity: <span id="humidity"></span>%</h3>
</body>
</html>
```

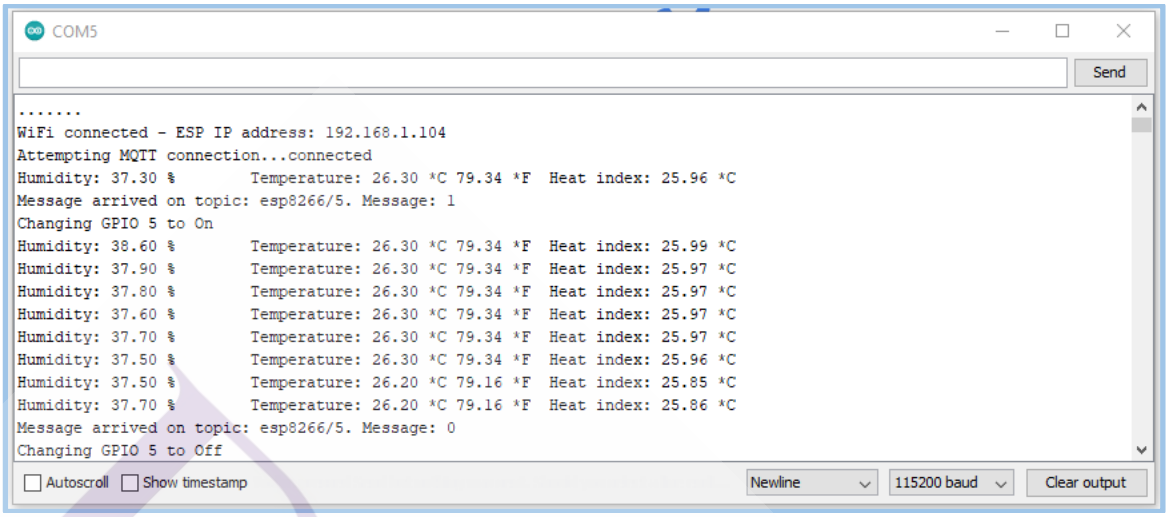
สร้างไฟล์ app.py

```
import paho.mqtt.client as mqtt
from flask import Flask, render_template, request, url_for
from flask_socketio import SocketIO, emit

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'
socketio = SocketIO(app)
```

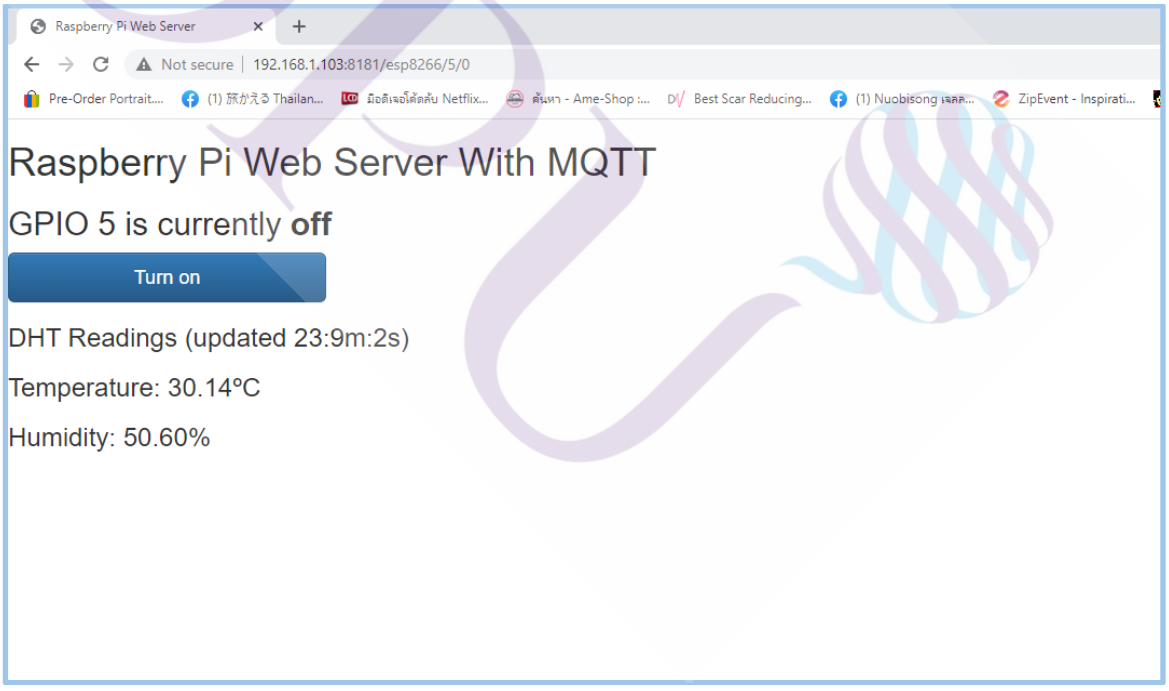
```
def trim_t(data):
  trim1_t = str(data).strip('\'')
  trim2_t = trim1_t.strip('b\'')
  trim3_t = trim2_t.strip()
  return trim3_t

def trim_h(data):
  trim1_h = str(data).strip('\'')
  trim2_h = trim1_h.strip('b\'')
  trim3_h = trim2_h.strip()
  return trim3_h


def on_connect(client, userdata, flags, rc):
   print("Connected with result code "+str(rc))

   client.subscribe("/esp8266/temperature")
   client.subscribe("/esp8266/humidity")

def on_message(client, userdata, message):

  print("Received message '" + str(message.payload) + "' on topic '"+ message.topic + "' with QoS
" + str(message.qos))

  if message.topic == "/esp8266/temperature":
     print("temperature update")
     socketio.emit('dht_temperature', {'data': trim_t(message.payload)})
  if message.topic == "/esp8266/humidity":
     print("humidity update")
     socketio.emit('dht_humidity', {'data': trim_h(message.payload)})

mqttc=mqtt.Client()
mqttc.on_connect = on_connect
mqttc.on_message = on_message
mqttc.connect("localhost",1883,60)
mqttc.loop_start()
```

```
pins = {
  5 : {'name' : 'GPIO 5', 'board' : 'esp8266', 'topic' : 'esp8266/5', 'state' : 'False'}
  }


templateData = {
  'pins' : pins
  }
@app.route("/")
def main():
  return render_template('main.html', async_mode=socketio.async_mode, **templateData)


@app.route("/<board>/<changePin>/<action>")
def action(board, changePin, action):
  changePin = int(changePin)
  devicePin = pins[changePin]['name']

  if action == "1" and board == 'esp8266':
    mqttc.publish(pins[changePin]['topic'],"1")
    pins[changePin]['state'] = 'True'
  if action == "0" and board == 'esp8266':
    mqttc.publish(pins[changePin]['topic'],"0")
    pins[changePin]['state'] = 'False'

  templateData = {
    'pins' : pins
  }
  return render_template('main.html', **templateData)

@socketio.on('my event')
def handle_my_custom_event(json):
   print('received json data : ' + str(json))

if __name__ == "__main__":
  socketio.run(app, host='0.0.0.0', port=8181, debug=True)
```

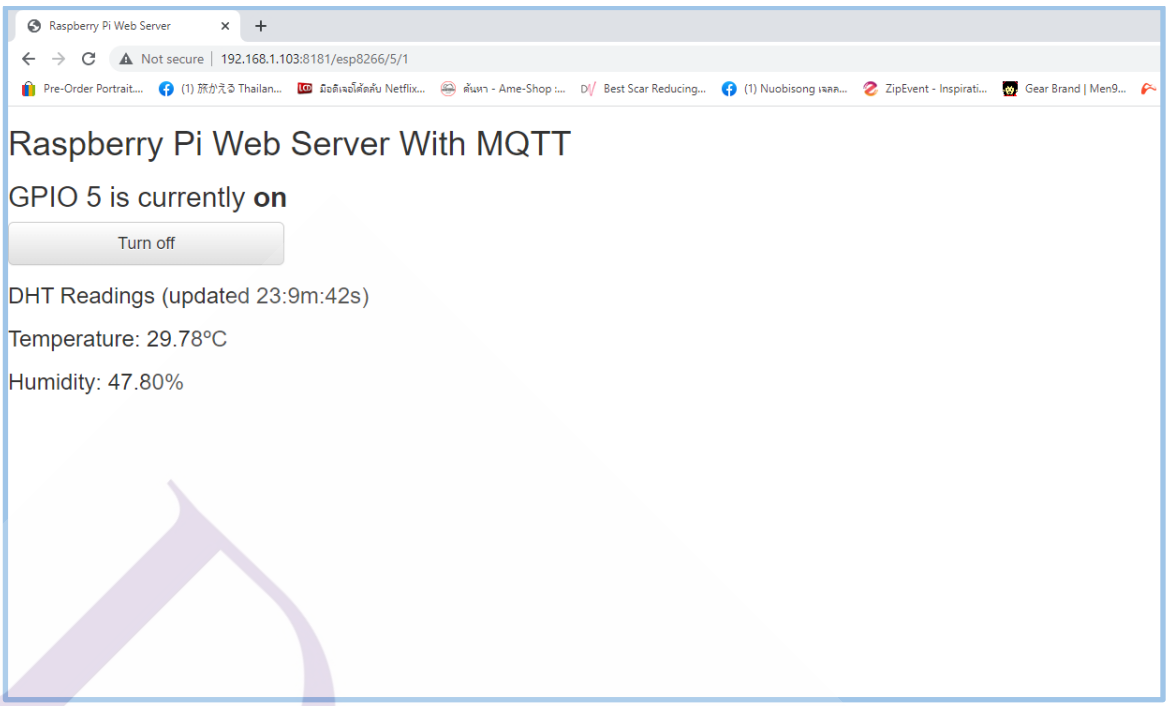
3.14 ทดสอบการทำงาน ระหว่าง Publisher และ Subscriber ผ่าน MQTT Broker



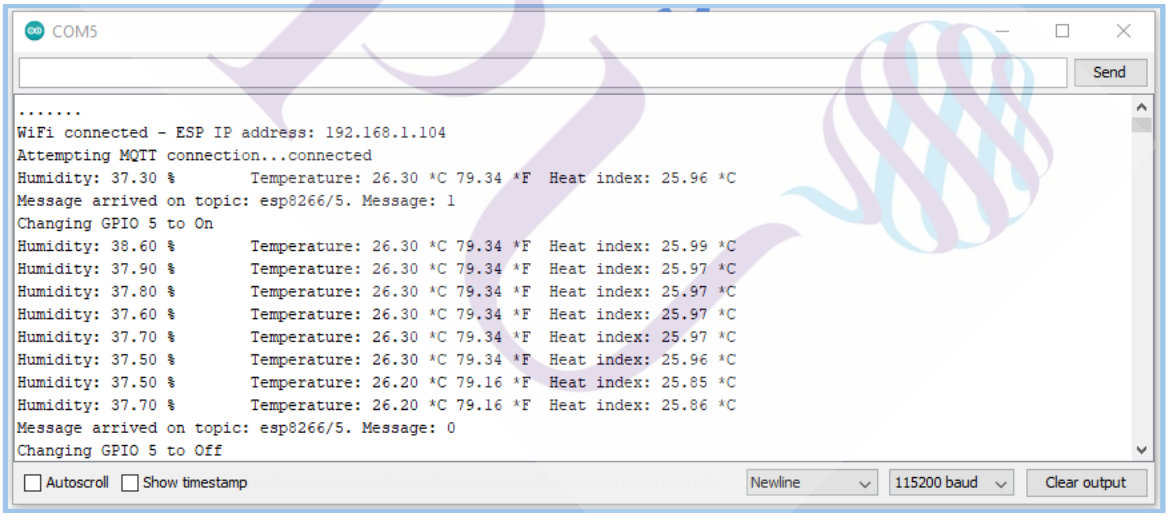
ภาพที่ 3.16 แสดงการทำงานของบอร์ด ESP 8266



ภาพที่ 3.17 แสดงการทำงานของ Web Application



ภาพที่ 3.18 แสดงการสั่งเปิด LED ผ่าน Web Application



ภาพที่ 3.19 ESP 8266 รับ Message จาก Web Application ผ่าน MQTT Broker เพื่อเปิด LED

3.15 แผนการโจมตี

ภายในงานวิจัยฉบับนี้ออกแบบแผนการโจมตีออกเป็น 2 ช่วง

1. ระหว่างการโจมตี
2. หลังการโจมตีสิ้นสุดลง

และมีลำดับของรูปแบบการโจมตีคือ

1. HTTP GET Flood Attack
2. TCP SYN Flood Attack
3. ICMP Flood Attack
4. UDP Flood Attack

โดยใช้การโจมตีเป็นระยะเวลาต่อเนื่อง 1 นาทีสำหรับรูปแบบ HTTP GET Flood Attack และ10 นาที สำหรับการโจมตีแบบ TCP SYN Flood, ICMP Flood, UDP Flood Attack โดยทุก ๆ 1 นาที จะทำการออกคำสั่งเพื่อสั่ง เปิดและปิด แอคชูเอเตอร์ LED, เรียกดูค่าอุณหภูมิและความชื้นผ่าน Web Application, ดูค่าอุณหภูมิความชื้นโดยตรงผ่าน Serial Monitor, ดูค่า Response Time ของ Web Application, ดูค่า CPU Usage ของ Raspberry Pi และดูค่า Memory ของ Raspberry Pi

หลังการโจมตีสิ้นสุดลงจะทำการออกคำสั่งทุก ๆ 1 นาทีเป็นระยะเวลา 5 นาที เพื่อทดสอบความสามารถในการกู้คืนระบบ โดยการออกคำสั่งเปิดและปิด แอคชูเอเตอร์ LED, ดูค่าอุณหภูมิและความชื้นผ่าน Web Application, ดูค่าอุณหภูมิความชื้นโดยตรงผ่าน Serial Monitor, ดูค่า Response Time ของ Web Application, ดูค่า CPU Usage ของ Raspberry Pi และดูค่า Memory ของ Raspberry Pi

บทที่ 4
ผลการวิจัย

4.1 การวิจัย

ในส่วนนี้จะกล่าวถึงการวิจัยและผลลัพท์ที่เกิดขึ้นจากการโจมตีด้วยรูปแบบการโจมตีแบบ TCP SYN Flood Attack, HTTP GET Flood Attack, ICMP Flood Attack และ UDP Flood Attack โดยจะแสดงวิธีการโจมตี, ระยะเวลาการโจมตี, จำนวน Packet ที่โจมตี, แสดงผลการทำงานของระบบ IoT ระหว่างเกิดการโจมตีและแสดงผลการกู้คืนตัวเองของระบบ IoT

4.2 ผลการวิจัย

ในงานวิจัยนี้เป็นการทดสอบการโจมตีระบบ IoT ที่สามารถสร้างด้วยตัวเองในรูปแบบของบ้านอัจฉริยะ (Smart Home) ด้วยรูปแบบการโจมตีในลักษณะ DDoS ที่เป็นที่นิยม เพื่อวัดประสิทธิภาพการทำงาน, ทดสอบขีดจำกัดของกำลังการประมวลผล และเพื่อทดสอบความมั่นคงปลอดภัยของระบบ IoT เมื่อมีการนำไปประยุกต์ใช้งาน

4.2.1 HTTP GET Flood Attack

ทำการโจมตี โดยการสร้าง GET Requests ผ่านเครื่องมือ WRK โดยแบ่งจำนวน Requests ในการทดสอบออกเป็น 500 1000 5000 10000 50000 100000 โดยกำหนดระยะเวลาการโจมตีที่ 1 นาที และได้ผลการทดสอบดังตารางนี้ [3]

Web Page: สามารถเข้าถึง Web Application

Web Load Time: ระยะเวลาในการเข้าถึง Web Application ในหน่วย มิลลิวินาที

LED: ระยะเวลาการตอบสนองของการสั่งการ LED ในหน่วย มิลลิวินาที

Humidity: การรับข้อมูลอุณหภูมิและความชื้นมาแสดงผลบน Web Application

ESP Humidity: การวัดอุณหภูมิและความชื้นของ DHT22 ที่เชื่อมต่อกับบอร์ด ESP8266

PI CPU(%) : ปริมาณการใช้กำลังการประมวลผลของ Raspberry Pi

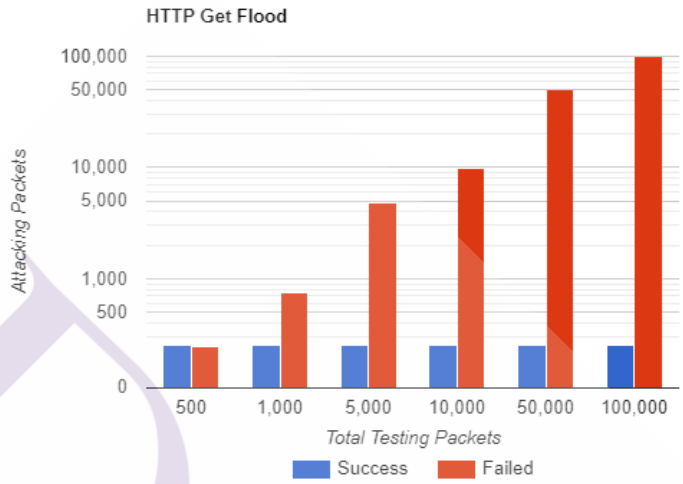
Pi Mem(%) : ปริมาณการใช้หน่วยความจำของ Raspberry Pi

Average Maximum Request/Sec: ปริมาณ Packet สูงสุดต่อวินาทีที่ Raspberry Pi สามารถประมวลผลได้

	Web Page	**Web Load Time (ms)**	**LED**	**Humidity**	**ESP Humidity**	**Pi CPU (%)**	**Pi Mem (%)**	**Average Maximum Requests/Sec**
Packet								
500	Y	2298	3030	Y	Y	94.9	0.3	172
1000	Y	3989	4032	Y	Y	98.5	0.3	174
5000	Y	3031	3601	Y	Y	93	0.3	173
10000	Y	2800	3349	Y	Y	93.1	0.3	175
50000	Y	7098	9240	Y	Y	95.9	0.3	173
100000	Y	8704	9952	Y	Y	97.4	0.3	173

การโจมตีทั้ง 6 ครั้งแสดงให้เห็นว่า Raspberry Pi สามารถรับปริมาณ GET Requests ได้สูงสุดเฉลี่ยอยู่ที่ 173 Requests ต่อวินาที และปริมาณการใช้ CPU ของ Raspberry PI เพิ่มสูงขึ้นระหว่างที่เกิดการโจมตี การเข้าถึง Web Page ระหว่างการโจมตีเกือบทั้งหมดจะไม่สามารถเข้าถึงได้ หากครั้งไหนที่สามารถเข้าถึงได้จะมี Response Time ที่สูงจากปกติ การสั่งเปิดและปิดแอคชูเอเตอร์ LED ก็มี Response Time ที่สูงขึ้น การรับค่าอุณหภูมิและความชื้นจาก MQTT Broker มาแสดงบน

Web Application ยังทำงานได้ การอ่านค่าอุณหภูมิและความชื้นของ DHT 22 ที่เชื่อมต่อกับบอร์ด ESP 8266 ยังทำงานได้ปกติ



ภาพที่ 4.1 กราฟแสดงปริมาณ Packet ที่ Raspberry Pi สามารถประมวลผลได้และประมวลผลไม่ได้ เทียบกับจำนวนปริมาณ Packet ทั้งหมดที่โจมตี

4.2.2 TCP SYN Flood Attack

การทดสอบโจมตี ไปที่ระบบ IoT จำลองการทดสอบ ด้วยการโจมตีด้วย TCP SYN Flood Attack โดยใช้ HPing 3 เป็น Packet Generator โดยการใช้ Command “ sudo hping3 -d 120 -S -V -p 8181 192.168.1.103 --flood --rand-source ”

ทำการโจมตีเป็นระยะเวลา 10 นาที โดยสร้าง Packet ส่งไปโจมตีทั้งหมด 65,367,204 Packet โดยแต่ละ Packet มีขนาด 120 Bytes และ ปลอมหมายเลข IP Address ต้นทาง (Spoofing Source IP) ค่าเฉลี่ยการโจมตีอยู่ที่ 108,000 Packet ต่อ วินาทีและได้ผลการทดสอบดังตารางนี้ [2],[3]

Web Page : สามารถเข้าถึง Web Application

Web Load Time : ระยะเวลาในการเข้าถึง Web Application ในหน่วย มิลลิวินาที

LED : ระยะเวลาการตอบสนองของการสั่งการ LED ในหน่วย มิลลิวินาที

Humidity : การรับข้อมูลอุณหภูมิและความชื้นมาแสดงผลบน Web Application

ESP Humidity : การวัดอุณหภูมิและความชื้นของ DHT22 ที่เชื่อมต่อกับบอร์ด ESP8266

PI CPU(%) : ปริมาณการใช้กำลังการประมวลผลของ Raspberry Pi

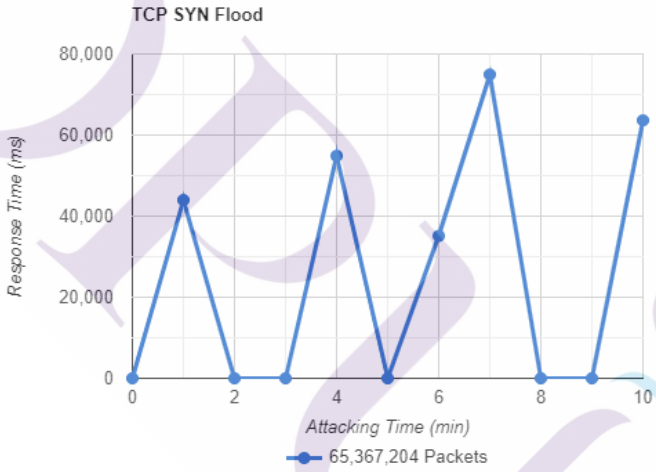
Pi Mem(%) : ปริมาณการใช้หน่วยความจำของ Raspberry Pi

	WebPage	**Web Load Time(ms)**	**LED**	**Humidity**	**ESP Humidity**	**PI CPU (%)**	**PI Mem (%)**
Time							
1min	Y	43973	25481 / ไม่สามารถสั่งปิด LED ได้	Y	Y	132	0.4
2min	N		N	N	Y	76.9	0.4
3min	N		N	N	Y	23	0.9
4min	Y	54876	N	N	Y	113	0.5
5min	N		N	N	Y	106	0.5
6min	Y	35136	N	Y	Y	86.5	0.5
7min	Y	74932	N	Y	Y	93.5	0.5
8min	N		N	N	Y	120	0.5
9min	N		N	N	Y	113	0.5
10min	Y	63635	N	N	Y	103	0.5

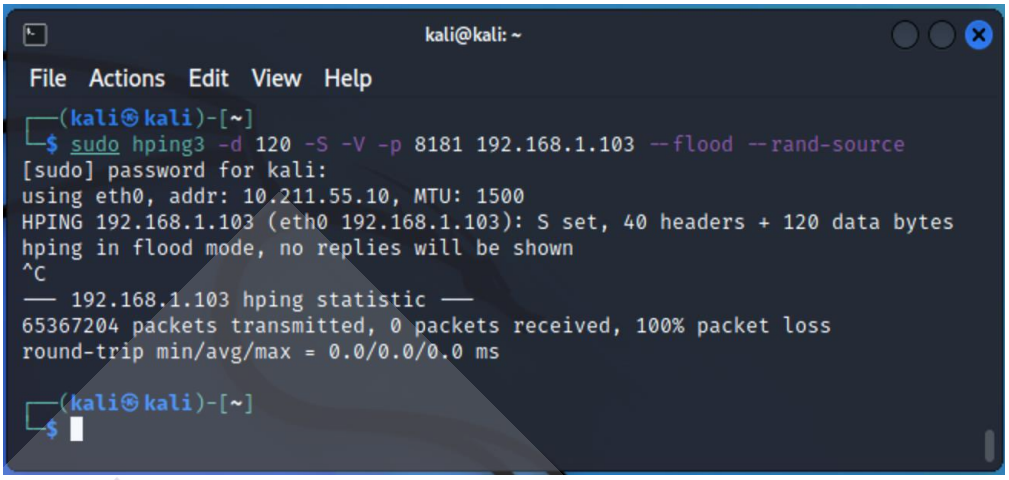
Raspberry Pi ที่ทำหน้าที่เป็น MQTT Broker มีปริมาณการใช้ CPU ที่สูงขึ้น และการเข้าถึง Web Application สำหรับความคุมแอคชูเอเตอร์ LED มีการทำงานที่ช้าลง

ระหว่างการโจมตีตลอด 10 นาที สามารถเรียก Web Application ได้เพียง 5 ครั้งจากทั้งหมด 10 ครั้ง การสั่งเปิดและปิดแอคชูเอเตอร์ LED มีเพียงครั้งเดียวที่สามารถสั่งเปิดได้โดยที่ไม่สามารถสั่งปิดได้ ครั้งอื่น ๆ ไม่สามารถสั่งเปิดและปิดแอคชูเอเตอร์ LED ได้ การแสดงค่าอุณหภูมิและความชื้นที่รับค่าจาก MQTT Broker บน Web Application สามารถทำได้เพียง 3 ครั้ง การทำงานของเซนเซอร์ DHT 22 ที่ไมโครคอนโทรลเลอร์ ESP 8266 ยังทำงานได้ปกติ

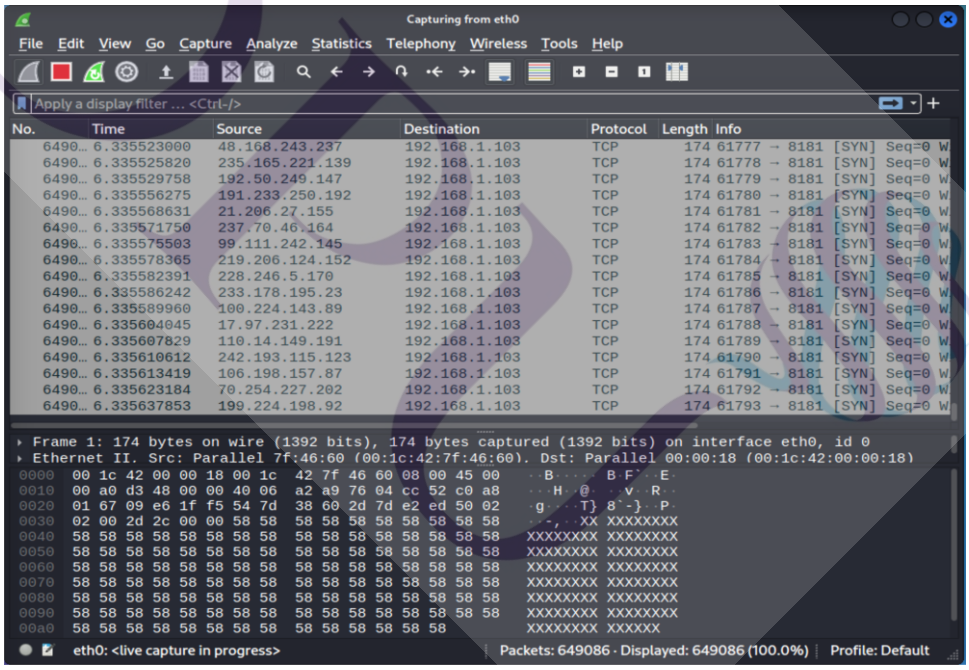
Raspberry Pi มีความสามารถในการกู้คืนตนเอง (Resilience) โดย Web Application กลับมาให้บริการได้หลังจากหยุดโจมตี 15-20วินาที แต่ไม่สามารถควบคุมสั่งการแอคชูเอเตอร์ LED ให้เปิดได้ ต้องทำการ Reboot ไมโครคอนโทรลเลอร์ ESP 8266



ภาพที่ 4.2 กราฟแสดงระยะเวลาการตอบสนองของ Web Application เทียบต่อระยะเวลาการโจมตีแบบ TCP SYN Flood Attack



ภาพที่ 4.3 คำสั่งสร้างการโจมตีแบบ TCP SYN Flood Attack และจำนวน Packet ทั้งหมดที่ส่งไปโจมตี



ภาพที่ 4.4 ตรวจสอบ TCP Packet ที่ส่งออกไปทำการโจมตีโดยใช้ Wire shark

4.2.3 ICMP Flood Attack

การทดสอบการโจมตีไปที่ระบบ IoT จำลองการทดสอบ โดยใช้ HPing3 สร้าง ICMP Packet โดยการใช้ Command

“ sudo hping3 -1 -d 120 192.168.1.103 --flood --rand-source ”

ทำการโจมตีเป็นระยะเวลา 10 นาทีโดยสร้าง Packet ไปโจมตีทั้งหมด 66,560,075 Packet โดยแต่ละ Packet มีขนาด 120 Bytes และปลอมหมายเลข IP Address ต้นทาง (Spoofing Source IP) ค่าเฉลี่ยการโจมตีอยู่ที่ 110,000 Packet ต่อ วินาที ได้ผลการทดสอบดังตารางนี้ [2],[3]

Web Page: สามารถเข้าถึง Web Application

Web Load Time: ระยะเวลาในการเข้าถึง Web Application ในหน่วย มิลลิวินาที

LED: ระยะเวลาการตอบสนองของการสั่งการ LED ในหน่วย มิลลิวินาที

Humidity: การรับข้อมูลอุณหภูมิและความชื้นมาแสดงผลบน Web Application

ESP Humidity: การวัดอุณหภูมิและความชื้นของ DHT22 ที่เชื่อมต่อกับบอร์ด ESP8266

PI CPU(%) : ปริมาณการใช้กำลังการประมวลผลของ Raspberry Pi

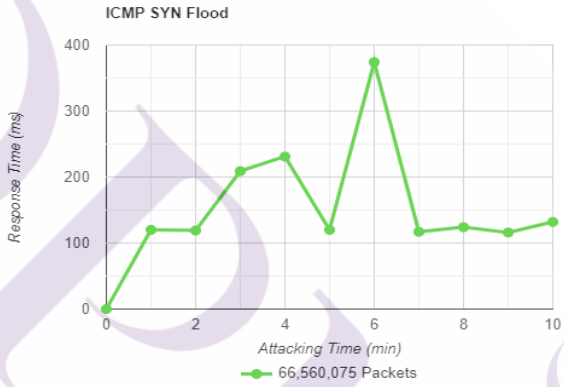
Pi Mem(%) : ปริมาณการใช้หน่วยความจำของ Raspberry Pi

	WebPage	**Web Load Time(ms)**	**LED**	**Humidity**	**ESP Humidity**	**PI CPU (%)**	**PI Mem (%)**
Time							
1min	Y	120	Y	Y	Y	21.9	0.9
2min	Y	119	Y	Y	Y	14.9	0.8
3min	Y	209	Y	Y	Y	2.7	0.5
4min	Y	231	Y	Y	Y	26.9	0.8
5min	Y	120	Y	Y	Y	17.9	0.8
6min	Y	374	Y	Y	Y	23.5	0.8
7min	Y	117	Y	Y	Y	19.9	0.8
8min	Y	124	Y	Y	Y	14.2	0.8
9min	Y	116	Y	Y	Y	16.2	0.8
10min	Y	132	Y	Y	Y	21.8	0.8

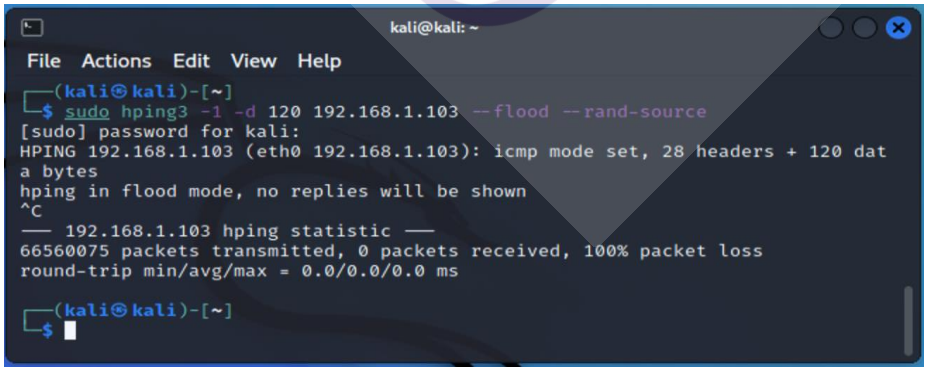
Raspberry Pi ที่ทำหน้าที่เป็น MQTT Broker มีปริมาณการใช้ CPU ที่สูงขึ้นเล็กน้อย และการเรียก Web Application สำหรับความคุมแอคชูเอเตอร์ LED มีการทำงานที่ใกล้เคียงกับช่วงเวลาปกติ

ระหว่างการโจมตีตลอดระยะเวลา 10 นาที การเข้าถึง Web Application สามารถทำได้ปกติ การสั่งการคำสั่งเปิดและปิดแอคชูเอเตอร์ LED สามารถทำได้ปกติ การรับค่าอุณหภูมิและความชื้นจาก MQTT Broker มาแสดงบน Web Application สามารถแสดงได้ปกติ เซนเซอร์ DHT22 ที่เชื่อมต่อกับไมโครคอนโทรลเลอร์ ESP 8266 สามารถทำงานได้ปกติ

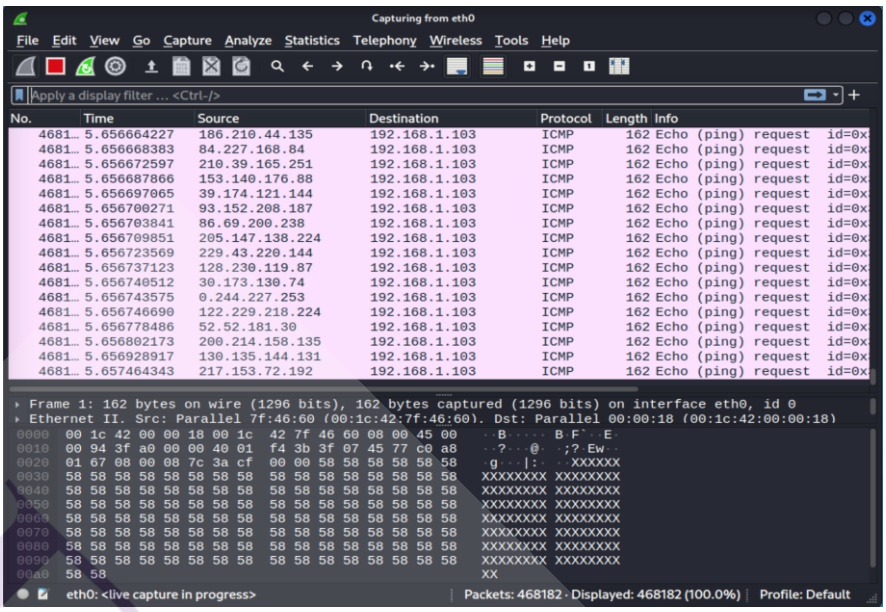
Raspberry Pi มีความสามารถในการกู้คืนตนเอง (Resilience) ที่ค่า Resource ต่าง ๆ เช่น CPU และ Memory กลับมาทำงานปกติทันทีหลังการโจมตีสิ้นสุดลง



ภาพที่ 4.5 กราฟแสดงระยะเวลาการตอบสนองของ Web Application เทียบต่อระยะเวลาการโจมตีแบบ ICMP Flood Attack



ภาพที่ 4.6 คำสั่งสร้างการโจมตีแบบ ICMP Flood Attack และจำนวน Packet ทั้งหมดที่ส่งไปโจมตี



ภาพที่ 4.7 ตรวจสอบ ICMP Packet ที่ส่งออกไปทำการโจมตีโดยใช้ Wire shark

4.2.4 UDP Flood Attack

การทดสอบการโจมตีไปที่ระบบ IoT จำลองการทดสอบ โดยใช้ HPing3 สร้าง UDP Packet โดยการใช้ Command

“ sudo hping3 --udp -d 120 -p 8181 192.168.1.103 --flood --rand-source ”

ทำการโจมตีเป็นระยะเวลา 10 นาทีโดยสร้าง Packet ไปโจมตีทั้งหมด 75,300,337 Packet โดยแต่ละ Packet มีขนาด 120 Bytes และ ปลอมหมายเลข IP Address ต้นทาง (Spoofing Source IP) ค่าเฉลี่ยการโจมตีอยู่ที่ 125,000 Packet ต่อ วินาที ได้ผลการทดสอบดังตารางนี้ [3]

Web Page: สามารถเข้าถึง Web Application

Web Load Time: ระยะเวลาในการเข้าถึง Web Application ในหน่วย มิลลิวินาที

LED: ระยะเวลาการตอบสนองของการสั่งการ LED ในหน่วย มิลลิวินาที

Humidity: การรับข้อมูลอุณหภูมิและความชื้นมาแสดงผลบน Web Application

ESP Humidity: การวัดอุณหภูมิและความชื้นของ DHT22 ที่เชื่อมต่อกับบอร์ด ESP8266

PI CPU(%) : ปริมาณการใช้กำลังการประมวลผลของ Raspberry Pi

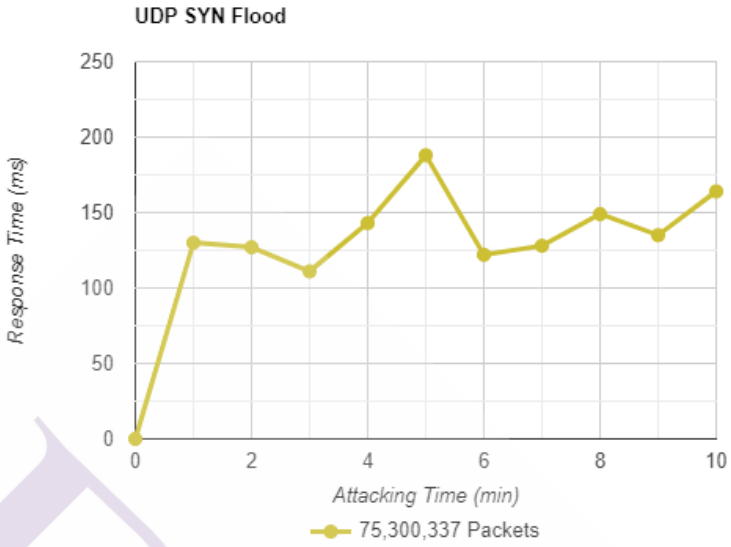
Pi Mem(%) : ปริมาณการใช้หน่วยความจำของ Raspberry Pi

	WebPage	Web Load Time(ms)	LED	Humidity	ESP Humidity	PI CPU (%)	PI Mem (%)
Time							
1min	Y	130	Y	Y	Y	12.2	0.9
2min	Y	127	Y	Y	Y	14.6	0.9
3min	Y	111	Y	Y	Y	17.1	0.9
4min	Y	143	Y	Y	Y	13.2	0.9
5min	Y	188	Y	Y	Y	10.4	0.9
6min	Y	122	Y	Y	Y	20.8	0.9
7min	Y	128	Y	Y	Y	19.9	0.9
8min	Y	149	Y	Y	Y	18.5	0.9
9min	Y	135	Y	Y	Y	17.6	0.9
10min	Y	164	Y	Y	Y	18.3	0.8

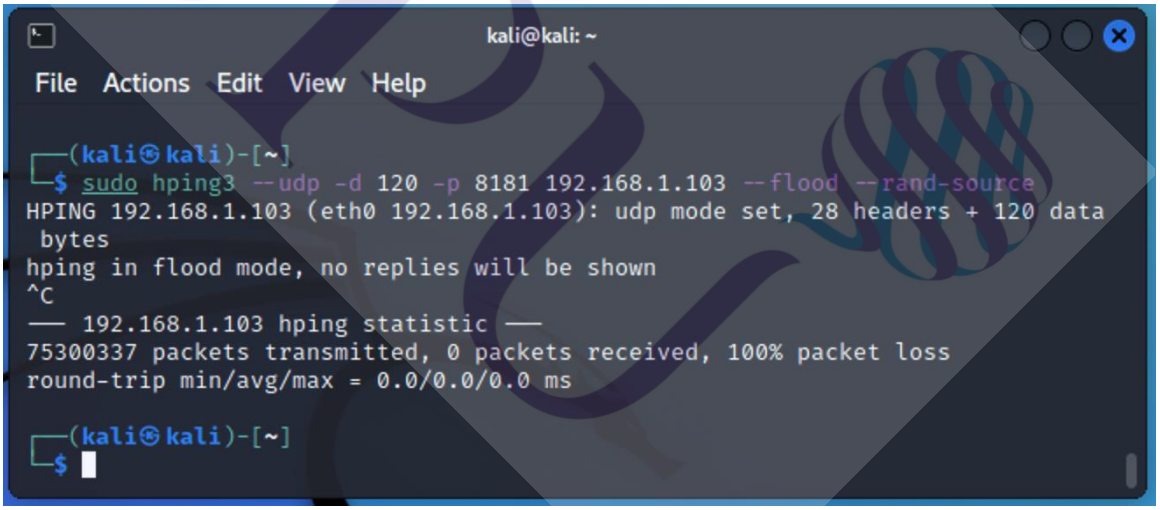
Raspberry Pi ที่ทำหน้าที่เป็น MQTT Broker มีปริมาณการใช้ CPU ที่สูงขึ้นเล็กน้อย และการเข้าถึง Web Application สำหรับความคุมแอคชูเอเตอร์ LED มีการทำงานที่ใกล้เคียงกับช่วงเวลาปกติ

ระหว่างการโจมตีตลอดระยะเวลา 10 นาที การเข้าถึง Web Application สามารถทำได้ปกติ การสั่งการคำสั่งเปิดและปิดแอคชูเอเตอร์ LED สามารถทำได้ปกติ การรับค่าอุณหภูมิและความชื้นจาก MQTT Broker มาแสดงบน Web Application สามารถแสดงได้ปกติ เซนเซอร์ DHT22 ที่เชื่อมต่อกับไมโครคอนโทรลเลอร์ ESP 8266 สามารถทำงานได้ปกติ

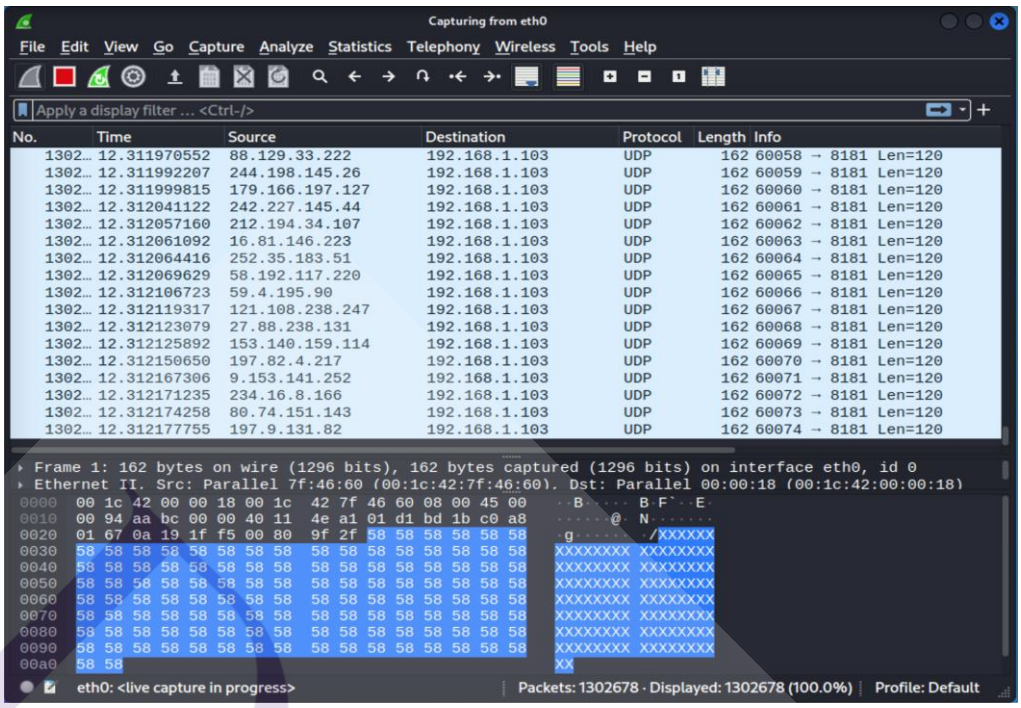
Raspberry Pi มีความสามารถในการกู้คืนตนเอง (Resilience) ที่ค่า Resource ต่าง ๆ เช่น CPU และ Memory กลับมาทำงานปกติทันทีหลังการโจมตีสิ้นสุดลง



ภาพที่ 4.8 กราฟแสดงระยะเวลาการตอบสนองของ Web Application เทียบต่อระยะเวลาการโจมตีแบบ UDP Flood Attack



ภาพที่ 4.9 คำสั่งสร้างการโจมตีแบบ UDP Flood Attack และจำนวน Packet ทั้งหมดที่ส่งไปโจมตี



ภาพที่ 4.10 ตรวจสอบ UDP Packet ที่ส่งออกไปทำการโจมตีโดยใช้ Wire shark

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

ระบบจำลองการทดสอบ IoT ในรูปแบบบ้านอัจฉริยะในงานวิจัยนี้แสดงให้เห็นถึงข้อจำกัดในด้านกำลังการประมวลของไมโครโพรเซสเซอร์ Raspberry Pi ที่นำมาประยุกต์ใช้ให้เป็นศูนย์ควบคุมกลาง (Control Center) และขีดจำกัดในด้านความมั่นคงปลอดภัย

HTTP GET Flood Attack

ไมโครโพรเซสเซอร์ Raspberry Pi มีข้อจำกัดด้านความสามารถประมวลผล GET Requests ได้เพียง 173 Requests ต่อวินาที และมีระยะเวลาการประมวลที่สูงขึ้น มีการใช้ปริมาณทรัพยากรในการประมวลที่สูงขึ้นแบบก้าวกระโดด มีอัตราการกู้คืนระบบด้วยตัวเองที่ใช้เวลาสูง

บทวิเคราะห์ HTTP GET Flood Attack

การโจมตีในรูปแบบ HTTP GET Flood Attack สามารถส่งผลกระทบกับการทำงานของระบบ IoT ได้อย่างชัดเจนหากปริมาณของ Packet GET Requests มีปริมาณที่สูงมาก อาจทำให้ระบบ IoT ไม่สามารถให้บริการได้ ซึ่งจะส่งผลกระทบให้เกิดความเสียหายต่อผู้ใช้งาน โดยแนวทางการป้องกันในการประยุกต์ใช้งานจริง หากเป็นระบบ IoT ที่ถูกใช้งานภายในองค์กรควรวางระบบ IoT ไว้ภายในเครือข่าย Intranet และมีใช้งาน Load Balance เพื่อช่วยจัดการกับปริมาณ Packet ที่มีจำนวนมาก หากเป็นการประยุกต์ใช้งานเองภายในครัวเรือน อาจเรียกใช้บริการ Commercial Cloud Services [2] เพื่อให้มีทรัพยากรที่มากพอกับการจัดการปริมาณ Packet ที่มีจำนวนมาก

TCP SYN Flood Attack

ไมโครโพรเซสเซอร์ Raspberry Pi มีความสามารถให้บริการ Web Application ในขณะเกิดการโจมตีได้เพียง 50% และไม่มีความสามารถในการควบคุมการทำงาน หรือการรับค่าข้อมูลจากอุปกรณ์เซนเซอร์ ตลอดระยะเวลาการโจมตี มีการใช้ปริมาณทรัพยากรในการประมวลที่สูงขึ้นเป็นอย่างมากเมื่อเปรียบเทียบกับช่วงเวลาการทำงานแบบปกติ มีอัตราการกู้คืนระบบด้วยตัวเองที่ใช้เวลานาน และอาจจำเป็นต้องมีการ Reboot ตัวอุปกรณ์เพื่อให้เริ่มทำงานใหม่

บทวิเคราะห์ TCP SYN Flood Attack

การโจมตีในรูปแบบ TCP SYN Flood Attack สามารถส่งผลกระทบทำให้ระบบ IoT ไม่สามารถให้บริการได้อย่างชัดเจน รวมถึงไม่สามารถออกคำสั่งเพื่อควบคุมอุปกรณ์เซนเซอร์และแอคชูเอเคอร์ได้ ซึ่งเป็นผลกระทบที่ร้ายแรงต่อผู้ใช้งาน เพื่อให้ระบบ IoT สามารถรับมือหรือทนต่อการโจมตีในรูปแบบนี้ ต้องมีการประยุกต์นำ IPS (Intrusion Prevention System) เข้ามาใช้ร่วมกับระบบ IoT โดยออกแบบในลักษณะแยกกันเพื่อให้ IPS และ IoT มีทรัพยากรของตัวเองที่เพียงพอต่อการใช้งาน หรือเลือกใช้งาน Commercial Cloud Services [2] เพื่อใช้บริการด้านความมั่นคงปลอดภัยจากผู้ให้บริการ

ICMP Flood Attack

ไมโครโพเซสเซอร์ Raspberry Pi มีความสามารถในการให้บริการ Web Application ที่ช้าลงกว่าช่วงเวลาปกติ และปริมารการใช้ CPU ของตัวอุปกรณ์มีแนวโน้มที่สูงขึ้น แต่ยังสามารถให้บริการการรับส่งข้อมูล หรือการควบคุมการสั่งการได้ มีอัตราการกู้คืนระบบด้วยตัวเองที่ค่อนข้างเร็ว

UDP Flood Attack

ไมโครโพเซสเซอร์ Raspberry Pi มีความสามารถในการให้บริการ Web Application ที่ช้าลงกว่าช่วงเวลาปกติ และปริมารการใช้ CPU ของตัวอุปกรณ์มีแนวโน้มที่สูงขึ้น แต่ยังสามารถให้บริการการรับส่งข้อมูล หรือการควบคุมการสั่งการได้ มีอัตราการกู้คืนระบบด้วยตัวเองที่ค่อนข้างเร็ว

บทวิเคราะห์ ICMP Flood Attack และ UDP Flood Attack

การโจมตีในรูปแบบทั้ง 2 นี้ไม่ค่อยส่งผลกระทบแบบเห็นได้ชัดกับการทำงานของระบบ IoT อาจเนื่องมาจาก ขนาดของ packet ที่ใช้ในการโจมตีของทั้ง 2 รูปแบบนี้มีขนาดเล็ก และระบบปฏิบัติการ Raspbian OS ของ Raspberry Pi มีการพัฒนาอย่างต่อเนื่องทำให้อาจมีการเตรียมการรองรับการใช้งาน Ping ที่มีประสิทธิภาพ และด้วยในปัจจุบันอุปกรณ์ที่เกี่ยวข้องกับเครือข่าย เช่น Routers มีปริมาณ Bandwidth ที่สูงมากหากภายในเครือข่ายมีอุปกรณ์ใช้งานที่ต่อพ่วงน้อย การโจมตีที่จะทำให้เกิด Overwhelming ใน Bandwidth ย่อมน้อยตามไปด้วย

สำหรับงานวิจัยต่อไปในอนาคต มีความเป็นไปได้ที่จะนำเอาความสามารถด้านความมั่นคงปลอดภัยเพิ่มเติม เช่น อุปกรณ์ IPS มาประยุกต์ร่วมใช้ภายในระบบ IoT แบบบ้านอัจฉริยะ (Smart Home) หรือการสร้างกฎของ Firewall ที่อุปกรณ์ไมโครโพรเซสเซอร์ Raspberry Pi เพื่อเพิ่มความมั่นคงปลอดภัยของระบบต่อไปได้

5.2 ปัญหาและข้อเสนอแนะ

5.2.1 ปัญหาที่พบในงานวิจัย

ชุดคำสั่งที่ใช้สร้างการโจมตีในรูปแบบ HTTP GET Flood Attack, TCP SYN Flood Attack, ICMP Flood Attack และ UDP Flood Attack ภายในงานวิจัยนี้มีการกำหนดขนาดของ Packet ที่ 120 Byte เพื่อไม่ให้เกินกำลังการทำงานของอุปกรณ์ Packet Generator หากมีการปรับขนาดของ Packet ที่ใช้สำหรับการโจมตีให้มีขนาดที่ใหญ่ขึ้น อาจจะส่งผลกระทบที่รุนแรงกับระบบ IoT แบบบ้านอัจฉริยะ (Smart Home) ของงานวิจัยนี้ได้มากขึ้น

5.2.2 ข้อเสนอแนะ

ภายในงานวิจัยนี้ได้ออกแบบระบบ IoT ในเชิงที่สามารถจัดหาอุปกรณ์ที่มีในท้องตลาดมาพัฒนาระบบด้วยตนเองได้ หากมีการนำอุปกรณ์ของค่ายผลิตภัณฑ์ที่มีวางขายในท้องตลาดมาประยุกต์ใช้ด้วย อาจทำให้ระบบมีความสามารถด้านความมั่นคงปลอดภัยที่สูงขึ้น

บรรณานุกรม

บรรณานุกรม

[1] A Survey: DDoS Attack on Internet of Things, Krushang Sonar, Hardik Upadhyay, Gujarat Technology University, India, International Journal of Engineering Research and Development, e-ISSN: 2278-067X,p-ISSN: 2278-800X
Internet of Things(IoT),(16 กันยายน 2563). สืบค้นจาก Google Scholar.

[2] Resistance of IoT Sensors against DDoS Attack in Smart Home Environment, Ladislav Huraj, Marek Simon, Department of Applied Informatics, University of SS. Cyril and Methodius ,Tibor Horak, Institute of Applied Informatics, Automation and Mechatronics, Faculty of Materials Science and Technology, Trnava, Slovak University of Technology, Slovakia, Sensors MDPI, 2022
Internet of Things(IoT),(4 กุมภาพันธ์ 2564). สืบค้นจาก Google Scholar.

[3] Vulnerability of Smart IoT-Based Automation and Control Devices to Cyber Attack, Tibor Horak, Marek Simon, Ladislav Huraj, and Roman Budjac, Institute of Applied Informatics, Automation and Mechatronics, Faculty of Materials Science and Technology, Trnava, Slovak University of Technology, Slovakia, Sensors MDPI, 2020
Internet of Things(IoT),(30 เมษายน 2561). สืบค้นจาก IEEE.

[4] A DDoS Attack Detection and Mitigation With Software-Defined Internet of Things Framework, Da Yin, Lianming Zhang, Kun Yang, College of Information Science and Engineering, Hunan Normal University, Changsha, IEEE Access, 2018
Security, (26 มิถุนายน 2561). สืบค้นจาก ACM Digital Library.

[5] DDoS Attacks on the Internet of Things and their prevention methods, Hanan Mustapha, Ahmed M Alghamdi, School of Computing, Mathematics, & Digital Technology, Manchester Metropolitan University, UK, 2nd International Conference on Future Networks and Distributed Systems.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายอินทัชพงค์ รัตนดิลก ณ ภูเก็ต
ประวัติการศึกษา	วิทยาศาสตร์บัณฑิต สาขาวิศวกรรมซอฟต์แวร์ มหาวิทยาลัยเชียงใหม่
ตำแหน่งและสถานที่ทำงานปัจจุบัน	นักเทคโนโลยีสารสนเทศปฏิบัติการระดับกลาง สำนักงานคณะกรรมการกิจการกระจายเสียง กิจการโทรทัศน์และกิจการโทรคมนาคมแห่งชาติ