ระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์

Computer Traffic Data Verification and Personal Data Import System via Computer Network

กิตติภูมิ จันพุดซา

โปรแกรมวิชาวิศวกรรมคอมพิวเตอร์

คณะเทคโนโลยีดิจิทัล

มหาวิทยาลัยราชภัฏเชียงราย

ปีการศึกษา 2566

(ลิขสิทธิ์ของมหาวิทยาลัยราชภัฏเชียงราย)

ระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์

ระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์

Computer Traffic Data Verification and Personal Data Import System via Computer Network

กิตติภูมิ จันพุดซา

โปรแกรมวิชาวิศวกรรมคอมพิวเตอร์

คณะเทคโนโลยีดิจิทัล

มหาวิทยาลัยราชภัฏเชียงราย

ปีการศึกษา 2566

ใบรับรองงานวิจัย

คณะเทคโนโลยีดิจิทัล

เรื่อง ระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์

โดย นายกิตติภูมิ จันพุดซา

ได้รับอนุมัติให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิตสาขาวิศวกรรมคอมพิวเตอร์

_____ รองคณบดีฝ่ายวิชาการ

(ผู้ช่วยศาสตราจารย์ อนุสรณ์ ใจแก้ว)

คณะกรรมการสอบงานวิจัย

_____ ประธานกรรมการ

(อาจารย์ อธิคม ศิริ)

_____ กรรมการ

(ผู้ช่วยศาสตราอาจารย์ ดร. ธนาวุฒิ ธนวาณิชย์)

_____ กรรมการ

(ผู้ช่วยศาสตราอาจารย์ ดร.กฤตกรณ์ ศรีวันนา)

_____ กรรมการ

(ผู้ช่วยศาสตราอาจารย์ ดร. มยูร ใยบัวเทศ)

บทคัดย่อ

ชื่องานวิจัย :	ระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์
ชื่อผู้วิจัย :	นายกิตติภูมิ จันพุดซา
สาขาวิชา :	วิศวกรรมคอมพิวเตอร์
ปีการศึกษา :	2566
อาจารย์ที่ปรึกษา :	อาจารย์อธิคม ศิริ

พระราชบัญญัติในเรื่องของการจัดเก็บข้อมูลจราจรคอมพิวเตอร์และคุ้มครองข้อมูลส่วนบุคคล พระราชบัญญัติว่าด้วยการกระทำความผิดเกี่ยวกับคอมพิวเตอร์ พ.ศ. 2560 และพระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. 2562 ได้เข้ามามีบทบาทในชีวิตประจำวันเนื่องจากทุกซอฟต์แวร์ในปัจจุบันได้มีการเข้าถึงข้อมูลของผู้ใช้งาน จึงจำเป็นต้องมีการตรวจสอบและป้องกันเป็นสิ่งที่ทำได้ลำบากจึงได้มีการเกิดขึ้นของระบบจัดการข้อมูลประเภทข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลขึ้น เพียงยังขาดในส่วนของรูปแบบการนำข้อมูลเข้า

ระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์จึงได้ถูกคิดขึ้น โดยมีการหลักการทำงานแบ่งเป็น 2 รูปแบบคือแบบเครื่องแม่ข่ายและเครื่องลูกข่าย เพื่อช่วยนำข้อมูลชนิดจราจรคอมพิวเตอร์หรือชนิดฐานข้อมูลจัดเก็บลงฐานข้อมูลเครื่องแม่ข่าย, นำไฟล์ที่เกี่ยวข้องข้อมูลจราจรคอมพิวเตอร์หรือข้อมูลส่วนบุคคลจัดเก็บไว้ยังเครื่องแม่ข่าย, หรือจะเป็นการนำเข้าข้อมูลกิจกรรมของการใช้งานอินเตอร์เน็ตโดยสามารถนำข้อมูลไปตรวจสอบต่อได้ และสุดท้ายการจัดการเครื่องลูกข่ายโดยผ่านเว็บแอพพลิเคชัน โดยมีทดลองการทำงานทั้งหมด 5 ส่วนดังนี้ส่วนที่ 1 การตรวจสอบความครบถ้วนตามข้อกำหนด ส่วนที่ 2 การตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ในแต่ละความเร็ว ส่วนที่ 3 การหาประสิทธิภาพของเวลาที่ใช้งานทุกฟังก์ชันในแต่ละประเภท ส่วนที่ 4 การหาประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่อง ส่วนที่ 5 การทดลองรองรับจำนวนโหลดของเครื่องลูกข่ายและความเร็วในเสร็จสิ้นการทำงาน ต่อ 1 ครั้ง

จากการทดลองระบบทำให้ได้ผลการทดสอบดังนี้ ส่วนที่ 1 การตรวจสอบความครบถ้วนตามข้อกำหนดได้ผลลัพธ์ ผ่านเงื่อนไขครบถ้วน ส่วนที่ 2 การตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ในแต่ละความเร็วได้ผลลัพธ์ จากทั้งหมด 2 รูปแบบ 100% ส่วนที่ 3 การตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานทุกฟังก์ชันในแต่ละประเภท โดยได้ผลลัพธ์ค่าเฉลี่ยรวม 7.1136 วินาที ส่วนที่ 4 การหาประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องในหน่วย ซีพียู 4.1%, แรม 2.84 Megabytes, และการอ่านดิสก์ 946.86 Megabytes per second การเขียนดิสก์ 365.69 Megabytes per second ส่วนที่ 5 การทดลองรองรับจำนวนโหลดของเครื่องลูกข่ายและความเร็วในเสร็จสิ้นการทำงาน ต่อ 1 ครั้ง ได้ผลลัพธ์เฉลี่ยรวมทั้งหมด 12.44 วินาที

กิตติกรรมประกาศ

การพัฒนาระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์ จะสำเร็จลุล่วงไปด้วยดีไม่ได้ หากไม่ได้รับความกรุณาจากบุคคลหลาย ๆ ท่าน ข้าพเจ้าขอขอบคุณบุคคลสำคัญ ดังที่จะกล่าวถึงดังต่อไปนี้เป็นอย่างยิ่งที่ช่วยให้คำแนะนำแก้ไขปัญหา ตลอดจนช่วยให้ข้าพเจ้าทำระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล สำเร็จลุล่วงไปได้ด้วยดี ซึ่งได้แก่ บิดามารดา และบริษัท เอสจีซี เซอร์วิส จำกัด ที่ให้การสนับสนุน และช่วยเหลือในการออกค่าใช้จ่ายมาโดยตลอดจน อาจารย์ อธิคม ศิริ อาจารย์ที่ปรึกษา, ผู้ช่วยศาสตราจารย์ ดร.ภูมิพงษ์ ดวงตั้ง, ผู้ช่วยศาสตราจารย์ ดร.ธนาวุฒิ ธนวาณิชย์, ผู้ช่วยศาสตราจารย์ กมล บุญล้อม, อาจารย์ ดร.กฤตกรณ์ ศรีวันนา, ผศ.ดร.มยูร ใยบัวเทศ ที่ค่อยสั่งสอนวิชาเพื่อนำไปใช้งาน ให้คำปรึกษาและคำแนะนำในการพัฒนาระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูลตั้งแต่เริ่มต้นจนเสร็จสิ้นกระบวนการทำงาน

สำหรับคุณงามความดีใดที่เกิดจากการพัฒนาระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล ข้าพเจ้าขอมอบให้กับบิดามารดา และบริษัท เอสจีซี เซอร์วิส ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้ และถ่ายทอดประสบการณ์ที่ดีต่อข้าพเจ้าช่วยเหลือและแนะนำสิ่งต่าง ๆ ให้กับ ข้าพเจ้า

นายกิตติภูมิ จันพุดซา

1 พฤศจิกายน 2566

สารบัญ

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญตาราง

สารบัญตาราง (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันซอฟต์แวร์คอมพิวเตอร์ได้เข้ามาบทบาทต่อการดำเนินชีวิตของมนุษย์เป็นอย่างมาก ซึ่งซอฟต์แวร์ได้กลายเป็นปัจจัยที่สำคัญที่มีผลต่อการทำงานขององค์กรและ หน่วยงานต่างๆ ทำให้ประสิทธิภาพในการทำงานมากยิ่งขึ้น ในยุคของข้อมูลข่าวสารที่มีความสำคัญ จนได้เกิดพระราชบัญญัติคอมพิวเตอร์ที่ต้องคุ้มครองข้อมูลไม่ว่าจะเป็นรูปแบบกิจกรรมที่อุปกรณ์ได้มีการใช้งานอินเตอร์เน็ตและข้อมูลส่วนบุคคลที่มีการเข้าถึง ที่ได้นำซอฟต์แวร์ไปใช้ในหน่วยงานสถานที่ต่างๆ เช่น เว็บไซต์ข่าวสารทางโรงเรียน, เว็บไซต์การตรวจสอบชื่อนักศึกษาในมหาวิทยาลัย, และเว็บไซต์จัดการข้อมูลจราจรคอมพิวเตอร์ ตลอดจนการนำไปใช้ในหน่วยงานราชการ นอกจากนี้ซอฟต์แวร์คอมพิวเตอร์เปรียบเสมือนเครื่องมือสำหรับช่วยให้การทำงานสะดวกสบายขึ้น ทั้งนี้ ซอฟต์แวร์คอมพิวเตอร์จำเป็นต้องอาศัยเครื่องมือ อุปกรณ์ ที่ทำให้ซอฟต์แวร์คอมพิวเตอร์สามารถใช้งานได้ เพื่อให้เข้าถึงแหล่งข้อมูลต่างๆ ในองค์กร หรือทั่วโลก โดยอุปกรณ์เครือข่ายต่างๆ และบริการต่างๆของเซิร์ฟเวอร์ แต่เนื่องจากทุกการกระทำของซอฟต์แวร์ ย่อมมีผลลัพธ์ในแบบที่ถูกกำหนดมา แต่เมื่อซอฟต์แวร์มีสิ่งหนึ่งที่บ่งบอกว่าซอฟต์แวร์ทำงานเกี่ยวข้องกับข้อมูลจราจรและข้อมูลส่วนบุคคลในการจัดเก็บข้อมูลนั้น จึงทำให้ซอฟต์แวร์คอมพิวเตอร์นั้นจะต้องมีการจัดการอย่างถูกวิธีและถูกต้องปลอดภัยให้การจัดการเป็นไปตามพระราชบัญญัติ แต่ถึงอย่างนั้นการจัดการซอฟต์แวร์คอมพิวเตอร์ก็ยังเป็นเรื่องที่ยุ่งยาก

ในยุคที่เทคโนโลยีสารสนเทศและอินเทอร์เน็ต ที่มีการพัฒนาอย่างรวดเร็ว จึงทำให้ซอฟต์แวร์คอมพิวเตอร์เข้าถึงข้อมูลส่วนบุคคลมากขึ้น ไม่ว่าจะเป็นซอฟต์แวร์แบบไหนก็ตาม อาจจะเป็นช่องโหว่ของผู้ไม่หวังดี เว็บไซต์ของทางราชการที่จะต้องมีการเก็บข้อมูลของประชากรในประเทศ ดังนั้นจึงได้มีพระราชบัญญัติเรื่องการเก็บรักษาข้อมูลจราจรคอมพิวเตอร์และคุ้มครองข้อมูลส่วนบุคคล เข้ามาช่วยควบคุมป้องกัน แต่ในปัจจุบันขาดเครื่องมือหรือซอฟต์แวร์ในการแก้ไขปัญหานี้

จากปัญหาที่กล่าวมานี้จึงได้คิดวิธีการแก้ไขปัญหาโดยพัฒนาซอฟต์แวร์เพื่อช่วยแก้ไขปัญหาและจัดการในส่วนของข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคล ส่วนของผู้วิจัยได้เข้าไปร่วมพัฒนาในส่วนของทางเลือกในการนำเข้าข้อมูลเพื่อป้องกันข้อมูลในรูปแบบเครือข่ายคอมพิวเตอร์ ช่วยตอบโจทย์ใน กรณีผู้ใช้งานซอฟต์แวร์ต้องการเห็นผลลัพธ์ของการป้องกันในรูปแบบการนำเข้าข้อมูลผ่านทาง โปรโตคอลต่างๆ ทางคอมพิวเตอร์เพื่อให้บริการให้ตรงกับความต้องการของพระราชบัญญัติคอมพิวเตอร์ในหัวข้อของการคุ้มครองข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคล (PDPA) โดยทางซอฟต์แวร์ได้ออกแบบรูปแบบการใช้งานสอดคล้องกับพระราชบัญญัติ เพื่อการนำเข้าข้อมูลที่ถูกเข้ารหัสของข้อมูลจราจรคอมพิวเตอร์, ไฟล์เอกสาร, หรือฐานข้อมูลประเภทต่าง ๆ ผ่าน

เทคโนโลยีการส่งไฟล์ผ่านรูปแบบของโปรโตคอลเอฟพีที (FTP Protocol) ลักษณะการทำงานของการส่งไฟล์ในรูปแบบนี้ เป็นการที่มีเครื่องเซิร์ฟเวอร์ที่เป็นตัวกลางที่รวบรวมไฟล์ต่างๆ ส่วนเครื่องลูกข่ายสามารถที่จะเข้าถึงไฟล์ได้โดยการช่องต่อผ่านหมายเลขอ้างอิง หรือ พอร์ต (Port) เมื่อเครื่องลูกข่ายสามารถเชื่อมต่อได้ ก็สามารถจัดการไฟล์ต่างๆที่อยู่บนเซิร์ฟเวอร์ แต่ต้องอยู่ในข้อกำหนดของเครื่องเซิร์ฟเวอร์ด้วยเช่นกัน และการติดต่อสื่อสารกันระหว่างเครื่องแม่ข่ายและเครื่องลูกข่ายผ่านรูปแบบของโปรโตคอลเน็ตเวิร์คซ็อกเก็ต (Socket) ลักษณะการทำงานของการติดต่อสื่อสารกันระหว่างเครื่องแม่ข่ายและลูกข่ายนั้น คือเครื่องแม่ข่ายนำเอาหมายเลขที่อยู่ของเน็ตเวิร์ค (IP Address) เปิดช่องทางให้สามารถเปิดรับข้อมูลผ่านหมายเลขอ้างอิง หรือ พอร์ต (Port) ส่วนเครื่องลูกข่ายก็ทำการเชื่อมต่อไปยังเครื่องแม่ข่ายผ่านหมายเลขที่อยู่ของเน็ตเวิร์ค (IP Address) หรือ พอร์ต (Port) ที่ตรงกับเครื่องแม่ข่าย เมื่อเชื่อมต่อกันได้ทั้งสองฝ่ายก็สามารถที่จะส่งข้อมูลติดต่อสื่อสารกันได้ ส่วนต่อมาเทคโนโลยีการวิเคราะห์โปรโตคอลบนอินเตอร์เน็ต ลักษณะการทำงานเป็นการดูว่าหมายเลขเน็ตเวิร์ค (IP Address) ที่ต้องการได้มีกิจกรรมอะไรเกิดขึ้นบ้าง และสามารถนำไปสู่การตรวจสอบได้ สุดท้ายจากเทคโนโลยีที่กล่าวมาเมื่อถึงขั้นตอนของการจัดการโดยผ่านเว็บแอพพลิเคชันที่ได้จัดทำขึ้นมา

ดังนั้นวิธีการรวบรวมข้อมูลมาจัดเก็บมีหลายวิธี ในการวิจัยนี้ เป็นหนึ่งในวิธีเพื่อช่วยแก้ไขปัญหาการรวบรวมข้อมูลที่ไม่สามารถใช้วิธีอื่นได้ เช่น การจัดเก็บข้อมูลการจราจร (Log) กรณีเป็น OS Windows อาจไม่สะดวกในการใช้ Syslog-NG ทางเราขอแนะนำอาจใช้วิธีตัวอย่างของงานวิจัยนี้เพื่อช่วยแก้ไขปัญหาได้ ส่วนวิธีการใช้ Hash 0 มาเพื่อตรวจสอบความถูกต้องของข้อมูล โดยในอดีต ตามมาตราฐานสากล นิยมใช้ค่า Hash 0 เป็นตัวยืนยันความถูกต้องของข้อมูลที่มาจัดเก็บ ผู้วิจัยจึงได้จำลองแบบการนำข้อมูลส่วนบุคคลที่ใช้วิธีรวบรวมตามแบบวิจัย มาทำการตรวจสอบ Hash 0 เพื่อช่วยในการยืนยันความถูกต้องของข้อมูลที่จัดเก็บตามมาตรฐานสากล เช่น มาตราฐาน Nectec ที่เป็นมาตราฐาน การจัดเก็บข้อมูลการจราจร เป็นต้น

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อสร้างระบบจัดการตรวจสอบและนำเข้าข้อมูลของข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์

1.2.2 เพื่อหาประสิทธิภาพการนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลของระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์

1.3 ขอบเขตการวิจัย

1.3.1 สามารถกำหนดประเภทของการตรวจสอบข้อมูลหรือนำเข้าข้อมูลที่กำหนดโดยเว็บแอพพลิเคชันตามข้อกำหนดพระราชบัญญัติการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ และคุ้มครองข้อมูลส่วนบุคคล

1.3.2 สามารถรองรับการทำงานบนระบบปฏิบัติ Oracle Linux version 8.7, Ubuntu Server 18.04, Windows 10, Windows 11รองรับฐานข้อมูล MySQL version 8.1.0, Oracle Database version 19c และรองรับภาษา Rust Language 1.63 หรือมากกว่า

1.4 ประโยชน์ที่ได้หวังว่าจะได้รับ

1.4.1 ระบบจัดการการนำเข้าข้อมูลของข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคล ข้อมูลค่าแฮช (Hash value), ไฟล์จราจรคอมพิวเตอร์, ไฟล์ข้อมูลส่วนบุคคล, และข้อมูลในฐานข้อมูล

1.4.2 รู้ถึงประสิทธิภาพที่ของระบบตรวจสอบและนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคลผ่านเครือข่ายคอมพิวเตอร์ ที่สามารถทำงานได้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

วิจัยเรื่องระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล ในการศึกษาครั้งนี้ ผู้วิจัยได้ศึกษาค้นคว้า และงานวิจัยที่เกี่ยวข้องเพื่อนำมาใช้สำหรับการกำหนดกรอบความคิด หลักการ ทฤษฎี เครื่องมือ การรวบรวมข้อมูล การวิเคราะห์และการอภิปรายผลการศึกษา ซึ่งประกอบไปด้วย เนื้อหา ดังต่อไปนี้

1. พระราชบัญญัติคอมพิวเตอร์ เรื่องการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ (Log)
2. พระราชบัญญัติข้อมูลส่วนบุคคล เรื่องคุ้มครองข้อมูลส่วนบุคคล (PDPA)
3. ระบบการนำข้อมูล (Data input system)
4. การตรวจสอบความถูกต้องของข้อมูล (Data verification)
5. เน็ตเวิร์คซ็อกเก็ต (Network Socket)
6. การส่งไฟล์บนระบบคอมพิวเตอร์ (File Transfer)
7. เครื่องมือวิเคราะห์แพ็คเก็ตอินเทอร์เน็ต (TCPDUMP)
8. หลักการทำงานให้สอดคล้องกับพระราชบัญญัติการคุ้มครองข้อมูลส่วนบุคคล (PDPA Process)

2.1 พระราชบัญญัติ เรื่องการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์

พระราชบัญญัติ เรื่องการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ (Log) หลักเกณฑ์การเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ (Data Retention Principles) เป็นระเบียบแนวทางที่กำหนดขั้นตอนและมาตรฐานสำหรับการเก็บรักษาข้อมูลที่เกี่ยวข้องกับการสื่อสารและการโอนข้อมูลผ่านระบบคอมพิวเตอร์ โดยเป้าหมายหลักคือการรักษาความเป็นส่วนตัวและความปลอดภัยของข้อมูลที่สำคัญ ดังนี้

2.1.1 การกำหนดระยะเวลาในการเก็บรักษาข้อมูล ผู้ให้บริการควรกำหนดระยะเวลาที่เหมาะสมในการเก็บรักษาข้อมูล โดยคำนึงถึงความจำเป็นของข้อมูลและข้อกำหนดกฎหมายที่เกี่ยวข้อง เช่น บางประเทศกฎหมายอาจกำหนดให้เก็บรักษาข้อมูลการสื่อสารเท่านั้นเป็นเวลา 90 วัน ตัวอย่าง บริษัทอินเทอร์เน็ตมือถืออาจกำหนดให้เก็บรักษาข้อมูลการโทรอาจารย์เป็นเวลา 180 วันหลังการโทรสิ้นสุด

2.1.2 การคำนึงถึงวัตถุประสงค์ในการเก็บรักษาข้อมูล ผู้ให้บริการควรระบุวัตถุประสงค์ที่ชัดเจนในการเก็บรักษาข้อมูล โดยการเก็บรักษาควรสอดคล้องกับวัตถุประสงค์นั้นๆ เท่านั้น ตัวอย่าง หากผู้ให้บริการใช้ข้อมูลการสื่อสารของลูกค้าเพื่อการประชาสัมพันธ์ ควรเก็บรักษาข้อมูลนี้เพียงเท่าที่จำเป็นเพื่อวัตถุประสงค์นี้

2.1.3 การควบคุมการเข้าถึงข้อมูล ผู้ให้บริการควรมีมาตรการในการควบคุมการเข้าถึงข้อมูลจราจรทางคอมพิวเตอร์ โดยจะต้องมีการบันทึกข้อมูลเกี่ยวกับผู้ที่มีสิทธิ์เข้าถึงข้อมูลนั้น ตัวอย่าง ผู้ให้บริการควรให้สิทธิ์ในการเข้าถึงข้อมูลจราจรทางคอมพิวเตอร์เฉพาะกับบุคคลที่มีความจำเป็นและให้มีระบบบันทึกการเข้าถึงข้อมูล

2.1.4 การควบคุมความปลอดภัย ผู้ให้บริการควรมีมาตรการความปลอดภัยที่เหมาะสมเพื่อป้องกันการเข้าถึงข้อมูลจราจรทางคอมพิวเตอร์โดยมิชอบ รวมถึงการเข้ารหัสข้อมูลที่เก็บรักษา ตัวอย่าง การใช้การเข้ารหัสข้อมูลที่เก็บรักษาเพื่อป้องกันการตรวจสอบข้อมูลจราจรทางคอมพิวเตอร์

2.1.5 การทำลายข้อมูล ผู้ให้บริการควรมีขั้นตอนการทำลายข้อมูลที่เก็บรักษาเมื่อไม่จำเป็นแล้ว และต้องทำตามข้อกำหนดของกฎหมายเกี่ยวกับการทำลายข้อมูล ตัวอย่าง หากข้อมูลการสื่อสารจราจรทางคอมพิวเตอร์ไม่จำเป็นต้องเก็บรักษาเนื่องจากหมดวัตถุประสงค์แล้ว ควรมีกระบวนการทำลายข้อมูลอย่างปลอดภัย

2.1.6 การบริการลูกค้า ผู้ให้บริการควรมีกระบวนการในการรับร้องเรียนและการติดต่อจากลูกค้าเกี่ยวกับข้อมูลที่เก็บรักษา ตัวอย่าง การตอบสนองต่อคำร้องเรียนเกี่ยวกับการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ในระยะเวลาที่เหมาะสม

2.1.7 การศึกษาและการพัฒนาต่อยอด ผู้ให้บริการควรคอยศึกษาและติดตามเทคโนโลยีและกฎหมายที่เปลี่ยนแปลงเกี่ยวกับการเก็บรักษาข้อมูล และพัฒนามากขึ้นตามประสิทธิภาพและความปลอดภัย ตัวอย่าง หลังจากการโจรกรรมหรือการแอบเข้าถึงข้อมูลจราจรทางคอมพิวเตอร์ผู้ให้บริการออนไลน์ควรปรับปรุงมาตรการความปลอดภัยเพิ่มเติมและจัดอบรมพนักงานในเรื่องความปลอดภัยข้อมูล

2.1.8 ติดต่อและร้องเรียน ผู้ใช้บริการควรมีวิธีติดต่อและร้องเรียนเกี่ยวกับการเก็บรักษาข้อมูล และผู้ให้บริการควรตอบสนองต่อคำร้องเรียนเหล่านี้ให้เร็วที่สุด ตัวอย่าง ผู้ใช้บริการควรสามารถติดต่อผู้ให้บริการผ่านช่องทางที่ระบุไว้ เช่น เบอร์โทรศัพท์หรืออีเมล

2.1.9 การปรับปรุงและการปรับแก้ไข ผู้ให้บริการควรปรับปรุงและอัพเดตนโยบายการเก็บรักษาข้อมูลให้เป็นไปตามกฎหมายและหลักเกณฑ์ที่เปลี่ยนแปลง ตัวอย่าง หากมีการเปลี่ยนแปลงในกฎหมายที่เกี่ยวข้องกับการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ ผู้ให้บริการควรปรับปรุงนโยบายตามกฎหมายใหม่

2.1.10 ความโปร่งใสและการเผยแพร่ ผู้ให้บริการควรเปิดเผยข้อมูลเกี่ยวกับนโยบายการเก็บรักษาข้อมูลและวิธีการใช้ข้อมูลแก่ผู้ใช้บริการโดยชัดเจน ตัวอย่าง การเผยแพร่นโยบายการเก็บรักษาข้อมูลบนเว็บไซต์ของผู้ให้บริการและในข้อตกลงการใช้บริการ

หลักเกณฑ์เหล่านี้เป็นสิ่งสำคัญที่ผู้ให้บริการควรปฏิบัติตามเพื่อรักษาความเป็นส่วนตัวและความปลอดภัยของข้อมูลจราจรทางคอมพิวเตอร์ โดยการประยุกต์ให้เหมาะสมกับความต้องการและกฎหมายของแต่ละสถานการณ์และองค์กรที่ใช้งานข้อมูลเหล่านี้

ต่อมาในเชิงของผู้ให้บริการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์เป็นกระบวนการที่สำคัญในการรักษาความปลอดภัยและความเป็นส่วนตัวของข้อมูลที่เกี่ยวข้องกับผู้ใช้บริการ การกำหนดหลักเกณฑ์ที่ดีในการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์มีความสำคัญอย่างมาก เพื่อให้มั่นใจได้ว่าข้อมูลจะถูกรักษาอย่างเหมาะสมและปลอดภัย ต่อไปนี้คือหลักเกณฑ์การเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ของผู้ให้บริการดังนี้

2.1.11 การรับรู้และการจัดเก็บข้อมูล ผู้ให้บริการควรระบุว่าข้อมูลจราจรทางคอมพิวเตอร์คืออะไร และในกรณีใดที่มีการเก็บรักษาข้อมูลนี้ เช่น บันทึกและเก็บข้อมูลการเข้าถึงเว็บไซต์ การสื่อสารผ่านระบบออนไลน์หรือข้อมูลการใช้บริการอื่นๆ ตัวอย่าง ผู้ให้บริการออนไลน์ระบุว่าพวกเขาจะเก็บรักษาประวัติการเข้าถึงเว็บไซต์และการใช้บริการเพื่อวัตถุประสงค์ในการวิเคราะห์และปรับปรุงประสิทธิภาพของเว็บไซต์

2.1.12 ระยะเวลาการเก็บรักษา ผู้ให้บริการควรระบุระยะเวลาที่ข้อมูลจราจรทางคอมพิวเตอร์จะถูกเก็บรักษา ระยะเวลานี้อาจแบ่งเป็นช่วงเวลาต่างๆ สำหรับข้อมูลที่ไม่เป็นประจำและข้อมูลที่มีความสำคัญมาก ตัวอย่าง บางข้อมูลอาจถูกเก็บรักษาเป็นระยะเวลา 30 วันเพื่อการตรวจสอบความปลอดภัย ในขณะที่ข้อมูลทางการบัญชีอาจถูกเก็บรักษาเป็นระยะเวลาหลายปีตามกฎหมายที่มีผลบังคับใช้

2.1.13 การประมวลผลข้อมูล ผู้ให้บริการควรระบุวัตถุประสงค์ของการเก็บรักษาข้อมูลและวิธีการประมวลผลข้อมูลนั้น รวมถึงผู้ที่มีสิทธิ์ในการเข้าถึงข้อมูล ตัวอย่าง ข้อมูลการใช้บริการออนไลน์อาจถูกใช้ในการปรับปรุงประสิทธิภาพของเว็บไซต์ และผู้ใช้บริการอาจได้รับข้อมูลผ่านทางอีเมลเกี่ยวกับโปรโมชั่นหรือข้อมูลที่เป็นประโยชน์

2.1.14 ความปลอดภัย การรักษาความปลอดภัยของข้อมูลจราจรทางคอมพิวเตอร์เป็นสิ่งสำคัญ ผู้ให้บริการควรมีมาตรการเพื่อป้องกันข้อมูลจราจรทางคอมพิวเตอร์จากการเข้าถึงและการนำข้อมูลไปใช้โดยไม่ได้รับอนุญาต ตัวอย่าง การใช้การเข้ารหัสข้อมูลและการใช้แฮชเพื่อป้องกันการโจรกรรมหรือการแอบเข้าถึงข้อมูลจราจรทางคอมพิวเตอร์

2.1.15 การเปิดเผยและการขายข้อมูล ผู้ให้บริการควรแสดงถึงนโยบายในการเปิดเผยข้อมูลแก่บุคคลที่สาม รวมถึงการขายข้อมูลแก่บุคคลที่สามถ้ามี ตัวอย่าง บริษัทออนไลน์ควรระบุว่าพวกเขาไม่ขายข้อมูลลูกค้าแก่บุคคลที่สามและมีนโยบายเปิดเผยข้อมูลลูกค้าอย่างชัดเจนในกรณีที่ต้องการทำเช่นนั้น

2.1.16 การตรวจสอบและการปรับแก้ไขนโยบาย ผู้ให้บริการควรมีกระบวนการตรวจสอบและการปรับแก้ไขนโยบายการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ให้สอดคล้องกับกฎหมายและข้อกำหนดที่เปลี่ยนแปลงได้ ตัวอย่าง บริษัทควรตรวจสอบและการปรับแก้ไขนโยบายการเก็บรักษาข้อมูลเมื่อมีการเปลี่ยนแปลงในกฎหมายที่เกี่ยวข้อง

2.1.17 การเรียนรู้และการปรับปรุง ผู้ให้บริการควรเรียนรู้จากการประสบความล้มเหลวและข้อผิดพลาดในการเก็บรักษาข้อมูล และพัฒนามากขึ้นตามประสิทธิภาพและความปลอดภัย ตัวอย่าง หลังจากการโจรกรรมหรือการแอบเข้าถึงข้อมูลจราจรทางคอมพิวเตอร์ผู้ให้บริการออนไลน์ควรปรับปรุงมาตรการความปลอดภัยเพิ่มเติมและจัดอบรมพนักงานในเรื่องความปลอดภัยข้อมูล

2.1.18 ติดต่อและร้องเรียน ผู้ใช้บริการควรมีวิธีติดต่อและร้องเรียนเกี่ยวกับการเก็บรักษาข้อมูล และผู้ให้บริการควรตอบสนองต่อคำร้องเรียนเหล่านี้ให้เร็วที่สุด ตัวอย่าง ผู้ใช้บริการควรสามารถติดต่อผู้ให้บริการผ่านทางอีเมลหรือแบบฟอร์มติดต่อเมื่อพบปัญหาเกี่ยวกับการเก็บรักษาข้อมูล

พระราชบัญญัติคอมพิวเตอร์ 2550 และ 2560 เป็นกฎหมายที่สำคัญในประเทศไทยที่เกี่ยวข้องกับการใช้และการจัดการข้อมูลจราจรของคอมพิวเตอร์ โดยเน้นไปที่ประเด็นของการควบคุมและป้องกันความผิดในด้านคอมพิวเตอร์ ต่อไปนี้ คือการอธิบายละเอียดเกี่ยวกับรูปแบบข้อมูลจราจรเชิงเทคนิคต่าง ๆ ที่เกี่ยวข้องกับพระราชบัญญัติ คอมพิวเตอร์ พร้อมกับตัวอย่างของแต่ละหัวข้อ

2.1.19 การบันทึกข้อมูลจราจร (Log Data) รูปแบบของข้อมูลจราจรที่ถูกบันทึกไว้ในระบบคอมพิวเตอร์ รวมถึงโครงสร้างข้อมูลและรูปแบบการบันทึกข้อมูลต่าง ๆ เช่น Text Logs, CSV Logs, JSON Logs และอื่น ๆ


```
{
        "timestamp": "2023-09-27T14:30:00",
        "source_ip": "192.168.1.1",
        "destination_ip": "203.0.113.1",
        "protocol": "HTTP",
        "request_method": "GET",
        "requested_url": "/example",
        "response_code": 200
}
```


ภาพที่ 2.1 ตัวอย่างของการบันทึกข้อมูลจราจรในรูปแบบของ JSON (Log Data)

2.1.20 การเข้ารหัสข้อมูลจราจร (Data Encryption) การใช้เทคนิคการเข้ารหัสข้อมูลเพื่อปกป้องความลับของข้อมูลจราจร ซึ่งรวมถึงการใช้วิธีการเข้ารหัสแบบทวิภาคและการเข้ารหัสแบบสมมุติ (Symmetric Encryption, Asymmetric Encryption)

2.1.21 การบรรจุข้อมูลจราจร (Data Packaging) วิธีการจัดรูปแบบข้อมูลจราจรเพื่อการจัดการและการส่งข้อมูล รวมถึงการใช้รูปแบบการบรรจุเชิง “ไบนารี” (Binary Packaging) และการบรรจุเชิงข้อความ (Text Packaging)

2.1.21 การตรวจสอบความถูกต้องของข้อมูล (Data Integrity) วิธีการตรวจสอบความถูกต้องของข้อมูลจราจร เพื่อป้องกันการเปลี่ยนแปลงข้อมูลที่ไม่ได้รับอนุญาต

2.1.22 การตรวจสอบความถูกต้องโดยแฮช (Data Integrity with Hash) เพื่อตรวจสอบความถูกต้องของข้อมูล รวมถึงการใช้ฟังก์ชันแฮชแบบต่าง ๆ เช่น SHA-256, MD5

2.1.23 การจัดการความปลอดภัยของข้อมูล (Data Security Management) มาตรการที่ใช้เพื่อความปลอดภัยของข้อมูลจราจร เช่น Firewalls, Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS)

2.1.24 การเข้าถึงและควบคุมสิทธิ์ (Access Control and Authorization) วิธีการควบคุมการเข้าถึงข้อมูลจราจรและการให้สิทธิ์การเข้าถึง เช่น การใช้งานรหัสผ่าน (Password), Access Control Lists (ACLs)

2.1.25 การเก็บรักษาข้อมูล (Data Retention) นโยบายและมาตรการในการเก็บรักษาข้อมูลจราจร เช่น ระยะเวลาการเก็บข้อมูล, การลบข้อมูลเมื่อไม่จำเป็น

2.1.26 การรายงานและการตรวจสอบ (Logging and Auditing) การบันทึกข้อมูลการทำงานและการตรวจสอบการใช้งานของระบบ เพื่อการติดตามและการตรวจสอบผู้ใช้งาน

2.1.27 การตอบสนองต่อการละเมิด (Incident Response) การตอบสนองในกรณีที่เกิดการละเมิดความปลอดภัยของข้อมูลจราจร รวมถึงกระบวนการและมาตรการในการตอบสนอง

ทั้งนี้การปฏิบัติที่ดีในการเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์จะช่วยให้ผู้ใช้บริการมีความมั่นใจในการใช้บริการและจะช่วยปรับปรุงประสิทธิภาพและความปลอดภัยของระบบในระยะยาวด้วย. ความเข้าใจและปฏิบัติตามหลักเกณฑ์เหล่านี้เป็นสิ่งสำคัญสำหรับผู้ให้บริการที่ต้องการรักษาความเชื่อมั่นของลูกค้าและปฏิบัติตามกฎหมายที่เกี่ยวข้องในการเก็บรักษาข้อมูล

2.2 พระราชบัญญัติ เรื่องคุ้มครองข้อมูลส่วนบุคคล

พระราชบัญญัติ เรื่องคุ้มครองข้อมูลส่วนบุคคล (PDPA) เป็นกฎหมายที่สร้างขึ้นเพื่อปกป้องความเป็นส่วนตัวและความปลอดภัยของข้อมูลส่วนบุคคลของบุคคลที่เกี่ยวข้อง การคุ้มครองข้อมูลส่วนบุคคลมีหลายแหล่งกฎหมายที่อาจแตกต่างกันตามประเทศ แต่มีหลักการและหัวข้อที่สำคัญที่ส่วนใหญ่ของกฎหมายคุ้มครองข้อมูลส่วนบุคคลมีอยู่ดังนี้

2.2.1 การรับรองความเป็นส่วนตัว กฎหมายคุ้มครองข้อมูลส่วนบุคคลกำหนดให้บุคคลที่ข้อมูลเกี่ยวข้องต้องให้ความยินยอมในการรับรองความเป็นส่วนตัวก่อนที่ข้อมูลจะถูกเก็บรวบรวม การรับรองความเป็นส่วนตัวนี้สามารถเป็นการยินยอมชัดแจ้งหรือการยินยอมที่ผู้เกี่ยวข้องรู้ได้ ตัวอย่าง การสร้างแบบฟอร์มการลงทะเบียนที่จะต้องรับรองความเป็นส่วนตัวจากผู้ใช้ก่อนที่ข้อมูลส่วนบุคคลของพวกเขาจะถูกบันทึก

2.2.2 การเก็บและการประมวลผลข้อมูล กฎหมายคุ้มครองข้อมูลส่วนบุคคลกำหนดกฎระเบียบเกี่ยวกับการเก็บและประมวลผลข้อมูลส่วนบุคคล รวมถึงความจำเป็นในการเก็บข้อมูลในระยะเวลาที่จำเป็นและความจำเป็นในการประมวลผลข้อมูลอย่างถูกต้อง ตัวอย่าง บริษัทต้องประมวลผลข้อมูลการชำระเงินของลูกค้าให้ถูกต้องเพื่อรักษาความเชื่อถือ

2.2.3 สิทธิของเจ้าของข้อมูล กฎหมายคุ้มครองข้อมูลส่วนบุคคลมอบสิทธิแก่บุคคลที่ข้อมูลเกี่ยวข้องในการเข้าถึง แก้ไข และลบข้อมูลส่วนบุคคลของตน และมีสิทธิในการร้องเรียนหรือปฏิเสธการประมวลผลข้อมูล ตัวอย่าง บุคคลที่ข้อมูลเกี่ยวข้องมีสิทธิในการขอให้บริษัทลบข้อมูลส่วนบุคคลของพวกเขาออกจากระบบ

2.2.4 ความปลอดภัยข้อมูล กฎหมายคุ้มครองข้อมูลส่วนบุคคลกำหนดความปลอดภัยข้อมูลและการป้องกันข้อมูลจากการเข้าถึง การทำลาย หรือการสูญหายโดยไม่ได้รับอนุญาต ตัวอย่าง การใช้เทคโนโลยีการเข้ารหัสข้อมูลเพื่อปกป้องข้อมูลส่วนบุคคลที่ถูกเก็บไว้ในระบบ

2.2.5 การแจ้งเตือนและการรายงานการขาดความปลอดภัยข้อมูล กฎหมายคุ้มครองข้อมูลส่วนบุคคลบังคับให้บริษัทแจ้งให้ทราบเมื่อมีการขาดความปลอดภัยข้อมูลและรายงานเหตุการณ์ดังกล่าวให้กับหน่วยงานที่

เกี่ยวข้องภายในระยะเวลาที่กำหนด ตัวอย่าง บริษัทต้องแจ้งให้ลูกค้าทราบเมื่อมีการละเมิดความปลอดภัยข้อมูลส่วนบุคคลและรายงานเหตุการณ์นี้ให้กับหน่วยงานคุมความปลอดภัย

2.2.6 ความสอดคล้องกับกฎหมาย กฎหมายคุ้มครองข้อมูลส่วนบุคคลกำหนดให้ผู้ให้บริการเป็นผู้รับผิดชอบในการดำเนินการตามกฎหมายคุ้มครองข้อมูลส่วนบุคคลที่เกี่ยวข้อง ตัวอย่าง การประมวลผลข้อมูลส่วนบุคคลตามข้อกำหนดของคำสั่งศาลหรือกฎหมายคุ้มครองข้อมูลส่วนบุคคลในประเทศที่ต้องปฏิบัติตามกฎหมาย

2.2.7 การพิจารณาเรื่องความสมบูรณ์และปราศจากข้อกังวล กฎหมายคุ้มครองข้อมูลส่วนบุคคลบังคับให้ผู้ให้บริการพิจารณาเรื่องความสมบูรณ์และปราศจากข้อกังวลของข้อมูลส่วนบุคคลที่เกี่ยวข้อง ตัวอย่าง การปรับปรุงนโยบายการเก็บรักษาข้อมูลเพื่อให้ความสมบูรณ์และปราศจากข้อกังวลในการใช้ข้อมูล

2.2.8 สอดคล้องกับความต้องการของผู้ให้บริการ การคุ้มครองข้อมูลส่วนบุคคลจะต้องสอดคล้องกับความต้องการและนโยบายของผู้ให้บริการและต้องปรับปรุงอย่างสม่ำเสมอเมื่อมีการเปลี่ยนแปลง ตัวอย่าง การปรับปรุงนโยบายความเป็นส่วนตัวเพื่อรวมข้อกำหนดใหม่หรือการเก็บรักษาข้อมูลตามความต้องการ

2.2.9 การจัดการข้อมูลส่วนบุคคลของบุคคลที่ข้อมูลเกี่ยวข้อง กฎหมายคุ้มครองข้อมูลส่วนบุคคลบังคับให้ผู้ให้บริการจัดการข้อมูลส่วนบุคคลของบุคคลที่ข้อมูลเกี่ยวข้องในลักษณะที่เหมาะสมและปลอดภัย ตัวอย่าง การจัดเก็บและการลบข้อมูลส่วนบุคคลตามคำขอของบุคคลที่ข้อมูลเกี่ยวข้อง

2.2.10 ความชดเชยและการใช้สัญญา กฎหมายคุ้มครองข้อมูลส่วนบุคคลอาจบังคับให้ผู้ให้บริการชดเชยบุคคลที่ข้อมูลเกี่ยวข้องในกรณีที่มีการละเมิดความเป็นส่วนตัวของพวกเขา และสามารถระบุข้อตกลงระหว่างบุคคลที่ข้อมูลเกี่ยวข้องและผู้ให้บริการ ตัวอย่าง การชดเชยลูกค้าที่ข้อมูลส่วนบุคคลของพวกเขาถูกทำลายโดยไม่ได้รับอนุญาต

2.2.11 การฝ่าฝืนและการละเมิด กฎหมายคุ้มครองข้อมูลส่วนบุคคลมีการกำหนดโทษและการละเมิดสำหรับผู้ให้บริการที่ไม่ปฏิบัติตามกฎหมายคุ้มครองข้อมูลส่วนบุคคล ตัวอย่าง การประมวลผลข้อมูลส่วนบุคคลโดยไม่ได้รับความยินยอมจากบุคคลที่ข้อมูลเกี่ยวข้อง

ดังนั้นกฎหมายคุ้มครองข้อมูลส่วนบุคคลเป็นเครื่องมือสำคัญในการปกป้องความเป็นส่วนตัวและความปลอดภัยของข้อมูลส่วนบุคคล และการปฏิบัติตามกฎหมายนี้มีผลกระทบต่อองค์กรและบุคคลที่ข้อมูลเกี่ยวข้องในลักษณะที่สำคัญ หากไม่ปฏิบัติตามกฎหมายคุ้มครองข้อมูลส่วนบุคคลอาจมีผลละเมิด การฟ้องร้อง และความเสียหายต่อธุรกิจและชื่อเสียงขององค์กร การเคลื่อนไหวในด้านความเป็นส่วนตัวและความปลอดภัยข้อมูลส่วนบุคคลอย่างรวดเร็วในปัจจุบันได้ส่งผลให้มีการปรับปรุงและการเข้มงวดกฎหมายคุ้มครองข้อมูลส่วนบุคคลในหลายประเทศทั่วโลก

2.3 ระบบการนำข้อมูล

ระบบการนำเข้าข้อมูล (Data Import System) เป็นระบบที่ถูกออกแบบเพื่อให้สามารถนำเข้าข้อมูลจากแหล่งต่าง ๆ เข้าสู่ระบบคอมพิวเตอร์หรือฐานข้อมูลเพื่อประมวลผลหรือการเก็บข้อมูลในรูปแบบที่เป็นประโยชน์ เช่น ข้อมูลลูกค้า, ข้อมูลการซื้อสินค้า, ข้อมูลการทำธุรกรรมการเงิน เป็นต้น

หลักการของระบบการนำเข้าข้อมูลมักจะเน้นการรักษาความถูกต้องและความครบถ้วนของข้อมูลที่ถูกนำเข้า เพื่อให้ข้อมูลที่ได้มีคุณภาพและน่าเชื่อถือ นอกจากนี้ยังรวมถึงการปรับเปลี่ยนรูปแบบข้อมูลหรือการแปลงข้อมูลให้อยู่ในรูปแบบที่เหมาะสมสำหรับการนำไปใช้งาน

การทำงานของระบบนี้มักเริ่มต้นด้วยกระบวนการตรวจสอบข้อมูลที่จะนำเข้า เช่น การตรวจสอบไฟล์ข้อมูลที่มีโครงสร้างถูกต้องหรือไม่ หรือการตรวจสอบรูปแบบของข้อมูล เพื่อให้มั่นใจว่าข้อมูลที่นำเข้าเป็นข้อมูลที่ถูกต้องและครบถ้วน

หลังจากนั้นระบบจะดำเนินการนำข้อมูลเข้าสู่ระบบ โดยมักจะมีกระบวนการแปลงข้อมูล (Data Transformation) หรือการทำความสะอาดข้อมูล (Data Cleansing) หากข้อมูลที่นำเข้ามีรูปแบบที่ไม่เหมาะสมหรือมีข้อมูลที่ไม่ถูกต้อง

นอกจากนี้ยังมีกระบวนการการบันทึกข้อมูล (Data Logging) เพื่อบันทึกข้อมูลการนำเข้าเพื่อให้สามารถติดตามได้ว่าข้อมูลไหนถูกนำเข้าและเมื่อไหร่

ยกตัวอย่างของหัวข้อที่เกี่ยวข้อง

2.3.1 ตรวจสอบความถูกต้องของข้อมูล ระบบตรวจสอบว่าข้อมูลที่นำเข้ามีโครงสร้างและรูปแบบถูกต้องหรือไม่ และตรวจสอบความครบถ้วนของข้อมูลตามที่กำหนด

2.3.2 การแปลงรูปแบบข้อมูล (Data Transformation) ถ้ามีความต้องการให้ข้อมูลอยู่ในรูปแบบที่แตกต่าง ระบบจะทำการแปลงรูปแบบข้อมูลให้เหมาะสม

2.3.3 การทำความสะอาดข้อมูล (Data Cleansing) กระบวนการตรวจสอบและแก้ไขข้อมูลที่ไม่ถูกต้องหรือไม่สมบูรณ์ เช่น การลบช่องว่างที่ไม่จำเป็น หรือการแก้ไขรูปแบบที่ไม่ถูกต้อง

2.3.4 การบันทึกข้อมูล (Data Logging) บันทึกข้อมูลการนำเข้าเพื่อเก็บบันทึกประวัติและการตรวจสอบโปรแกรมการทำงาน

2.3.5 การจัดเก็บข้อมูล (Data Storage) ข้อมูลที่นำเข้าจะถูกจัดเก็บในระบบฐานข้อมูลหรือโครงสร้างข้อมูลที่เหมาะสมต่อการใช้งานต่อไป

โครงการหรือระบบการทำงานต่าง ๆ สามารถใช้เทคโนโลยีและเครื่องมือที่หลากหลายเพื่อสร้างระบบการนำเข้าข้อมูลที่ตอบสนองตามความต้องการของโครงการหรือธุรกิจที่ใช้งานอยู่ ตัวอย่างของเครื่องมือที่ใช้ในการนำเข้าข้อมูลได้แก่ Apache NiFi, Talend, Microsoft SQL Server Integration Services (SSIS) และ Pentaho

Data Integration (Kettle) เป็นต้น แต่ละเครื่องมือนี้มีความสามารถและคุณสมบัติที่แตกต่างกันตามความต้องการของโครงการและผู้ใช้งานแต่ละราย

ระบบการนำเข้าข้อมูลเชิงเทคนิค (Technical Data Import System) เป็นระบบที่ถูกออกแบบเพื่อให้สามารถนำเข้าข้อมูลจากแหล่งต่าง ๆ ลงในระบบคอมพิวเตอร์หรือฐานข้อมูลได้อย่างรวดเร็วและปลอดภัย ระบบนี้มีหลักการทำงานและคุณสมบัติที่ต้องการเพื่อให้การนำเข้าข้อมูลเป็นไปอย่างมีประสิทธิภาพ นี่คือลักษณะที่สำคัญของระบบการนำเข้าข้อมูลเชิงเทคนิค

2.3.6 การตรวจสอบและการเตรียมข้อมูล (Data Validation and Preparation) ตรวจสอบความถูกต้องและความสมบูรณ์ของข้อมูลที่นำเข้า และทำการปรับเปลี่ยนหรือแปลงข้อมูลให้เป็นรูปแบบที่ถูกต้องและเหมาะสมกับระบบที่จะนำเข้า เช่น ตรวจสอบรูปแบบของไฟล์, การแปลงข้อมูลเป็นรูปแบบที่ถูกต้อง (เช่น วันที่ให้เป็นรูปแบบที่ต้องการ)

2.3.7 การนำเข้าข้อมูล (Data Import) นำข้อมูลที่ผ่านการตรวจสอบและการเตรียมข้อมูลมาเข้าสู่ระบบคอมพิวเตอร์หรือฐานข้อมูล เช่น การนำเข้าข้อมูลผ่าน SQL หรือ API ไปยังฐานข้อมูล

2.3.8 การการตรวจสอบและการเข้ารหัสข้อมูล (Data Encryption and Verification) ใช้เทคนิคการเข้ารหัสข้อมูล (Encryption) เพื่อปกป้องความลับของข้อมูลที่ถูกนำเข้า และตรวจสอบความถูกต้องของข้อมูลที่ถูกนำเข้า เช่น ใช้ SSL/TLS เพื่อการเข้ารหัสข้อมูลที่ถูกส่งผ่านเครือข่าย และการใช้วิธีการตรวจสอบแบบ checksum

2.3.9 การบันทึกข้อมูลการนำเข้า (Data Logging) บันทึกข้อมูลการนำเข้าเพื่อการติดตามและการตรวจสอบข้อมูลที่ถูกนำเข้า เช่น บันทึกข้อมูลเวลาที่นำเข้า, ผู้ใช้งานที่ทำการนำเข้า, และผลการนำเข้า

2.3.10 การจัดการข้อมูล (Data Management) จัดการข้อมูลที่ถูกนำเข้าในรูปแบบที่มีความเรียบร้อยและสามารถเข้าถึงได้ง่าย เช่น การจัดทำดัชนี (Indexing) เพื่อเพิ่มความเร็วในการค้นหาข้อมูล, การกำหนดสิทธิ์ในการเข้าถึงข้อมูล

2.3.11 การทำความสะอาดข้อมูล (Data Cleansing) กระบวนการตรวจสอบและแก้ไขข้อมูลที่ไม่ถูกต้องหรือไม่สมบูรณ์ เช่น การลบช่องว่างที่ไม่จำเป็น หรือการแก้ไขรูปแบบที่ไม่ถูกต้อง เช่น ตรวจสอบและแก้ไขข้อมูลที่มีรูปแบบไม่ถูกต้องหรือข้อมูลที่ขาดหาย

2.3.12 การจัดเก็บข้อมูล (Data Storage) ข้อมูลที่นำเข้าจะถูกจัดเก็บในระบบฐานข้อมูลหรือโครงสร้างข้อมูลที่เหมาะสมต่อการใช้งานต่อไป เช่น การจัดทำดัชนี (Indexing) เพื่อเพิ่มความเร็วในการค้นหาข้อมูล, การกำหนดสิทธิ์ในการเข้าถึงข้อมูล

2.4 การตรวจสอบความถูกต้องของข้อมูล

การตรวจสอบความถูกต้องของข้อมูล (Data verification) ยืนยันความถูกต้องของข้อมูลที่ถูกส่งหรือบันทึกเพื่อให้มั่นใจว่าข้อมูลดังกล่าวตรงตามรูปแบบและคุณภาพที่คาดหวัง กระบวนการนี้เป็นส่วนสำคัญในการ

รักษาความถูกต้องและความน่าเชื่อถือของข้อมูลที่นำมาใช้ในระบบการตรวจสอบความถูกต้องของข้อมูล (Data verification) เป็นกระบวนการที่มุ่งเน้นการตรวจสอบและหรือกระบวนการต่าง ๆ ในองค์กร ดังนั้นการตรวจสอบความถูกต้องของข้อมูลมีความสำคัญอย่างมากโดยมีหลักการของการตรวจสอบความถูกต้องของข้อมูล (Data verification) ดังนี้

2.4.1 ความสมบูรณ์ (Integrity) ของข้อมูล การตรวจสอบความสมบูรณ์ของข้อมูลเป็นกระบวนการที่ใช้เช็คว่าข้อมูลไม่ถูกเปลี่ยนแปลงหรือเสียหายในระหว่างการส่งหรือบันทึก โดยใช้การตรวจสอบค่าความสมบูรณ์เช่น การใช้เช็คซับแฮชหรือเช็คแสตมป์ (Checksum) เพื่อให้แน่ใจว่าข้อมูลไม่ถูกทำให้เสียหายในระหว่างการส่งผ่านเครือข่ายหรือการเก็บรักษา

2.4.2 ความถูกต้อง (Accuracy) ของข้อมูล การตรวจสอบความถูกต้องของข้อมูลเป็นกระบวนการที่เช็คว่าข้อมูลตรงตามรูปแบบและข้อมูลที่ถูกต้อง ยกตัวอย่างเช่น การตรวจสอบรูปแบบของเลขประจำตัวประชาชนหรือเลขโทรศัพท์ การตรวจสอบรูปแบบของอีเมล์, การตรวจสอบความถูกต้องของค่าตัวเลข เป็นต้น

2.4.3 ความครบถ้วน (Completeness) ของข้อมูล การตรวจสอบความครบถ้วนของข้อมูลหมายความว่าข้อมูลต้องไม่ขาดหายหรือหายไปในระหว่างกระบวนการการส่งหรือบันทึก ควรตรวจสอบว่าข้อมูลทุกส่วนที่จำเป็นสำหรับการประมวลผลถูกส่งหรือบันทึก

2.4.4 ความน่าเชื่อถือ (Reliability) ของข้อมูล การตรวจสอบความน่าเชื่อถือของข้อมูลหมายความว่าข้อมูลนั้นมีความน่าเชื่อถือและมีแหล่งกำเนิดที่น่าเชื่อถือ เช่น การตรวจสอบแหล่งที่มาของข้อมูล การตรวจสอบความน่าเชื่อถือของแหล่งข้อมูลหรือการตรวจสอบเอกสารยืนยัน

2.4.5 ความเป็นทรัพยากร (Resourcefulness) ของข้อมูล การตรวจสอบความเป็นทรัพยากรของข้อมูลหมายความว่าข้อมูลสามารถนำไปใช้งานได้อย่างมีประสิทธิภาพ ยกตัวอย่างเช่น การตรวจสอบว่าข้อมูลมีความถูกต้องและเข้ากับระบบที่ใช้งานได้

2.4.6 ความปลอดภัย (Security) ของข้อมูล การตรวจสอบความปลอดภัยของข้อมูลเป็นกระบวนการที่ใช้เช็คว่าข้อมูลไม่ถูกเข้าถึงหรือโจมตีโดยบุคคลที่ไม่พึงประสงค์ การตรวจสอบระบบความปลอดภัยและการเข้ารหัสข้อมูลเป็นส่วนหนึ่งของการตรวจสอบความปลอดภัยของข้อมูล

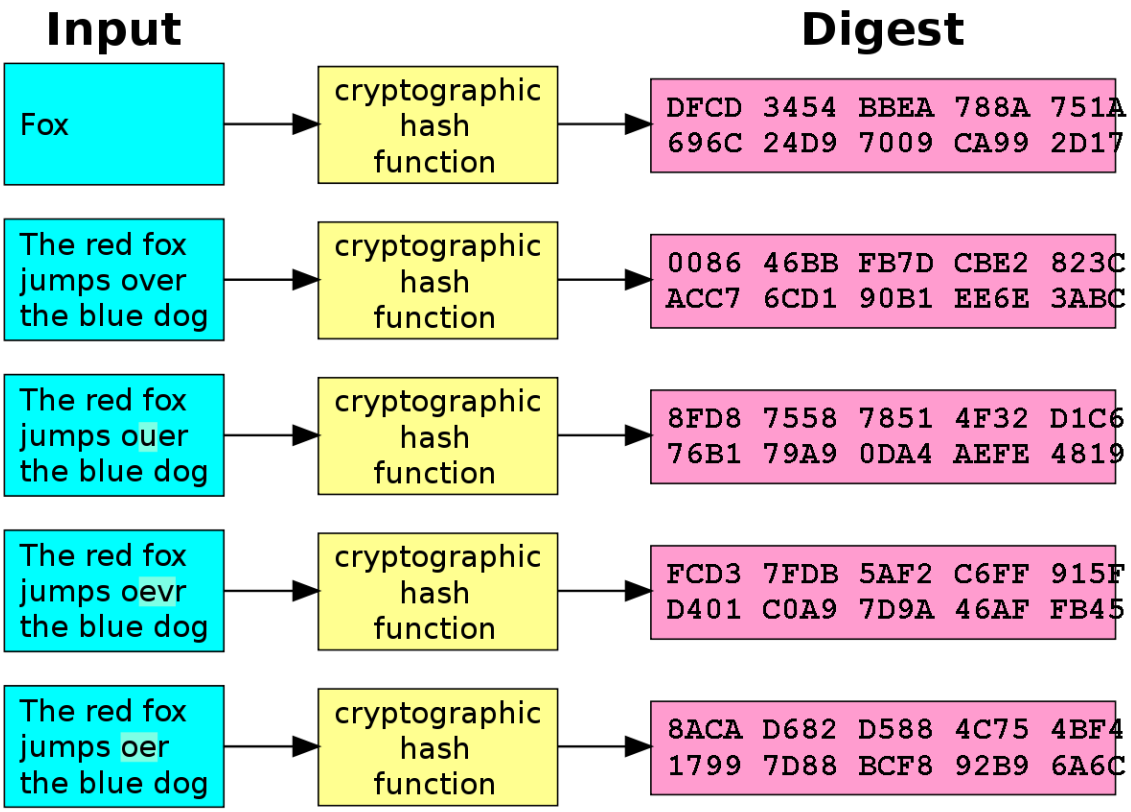
ดังนั้นการตรวจสอบความถูกต้องของข้อมูล (Data verification) จากหลักการต่างๆนำมาเริ่มกันนำมายกตัวอย่างการตรวจสอบความถูกต้องของข้อมูลบัญชีธนาคารโดยการตรวจสอบเลขบัญชีและยอดคงเหลือที่ตรงกับข้อมูลที่ระบบบันทึก การตรวจสอบความถูกต้องของเลขคำสั่งซื้อสินค้าในระบบการค้าออนไลน์ หรือการตรวจสอบความสมบูรณ์ของข้อมูลการสำรองข้อมูลเพื่อให้มั่นใจว่าข้อมูลสำรองเสร็จสมบูรณ์และสามารถกู้คืนได้

การตรวจสอบความถูกต้องและความครบถ้วนของข้อมูลมีบทบาทสำคัญในการรักษาคุณภาพและความน่าเชื่อถือของข้อมูลในระบบและกระบวนการต่าง ๆ และช่วยลดความเสี่ยงในการตัดสินใจและการดำเนินงานของ

องค์กรหรือระบบใด ๆ ที่เกี่ยวข้องกับข้อมูล และในงานของผู้วิจัยได้นำเสนอในส่วนของการตรวจสอบความถูกต้องข้อมูลโดยใช้วิธคือ แฮชฟังก์ชัน (Hash function)

โดยแฮชฟังก์ชัน (Hash Function) และอัลกอริทึมแฮช (Hash Algorithm) เป็นสองส่วนสำคัญในกระบวนการสร้างค่าแฮช (Hash Value) จากข้อมูลอินพุต (Input Data) โดยมีความเกี่ยวข้องกันอย่างมาก จะอธิบายเป็น 2 ส่วนคือรูปแบบดังนี้

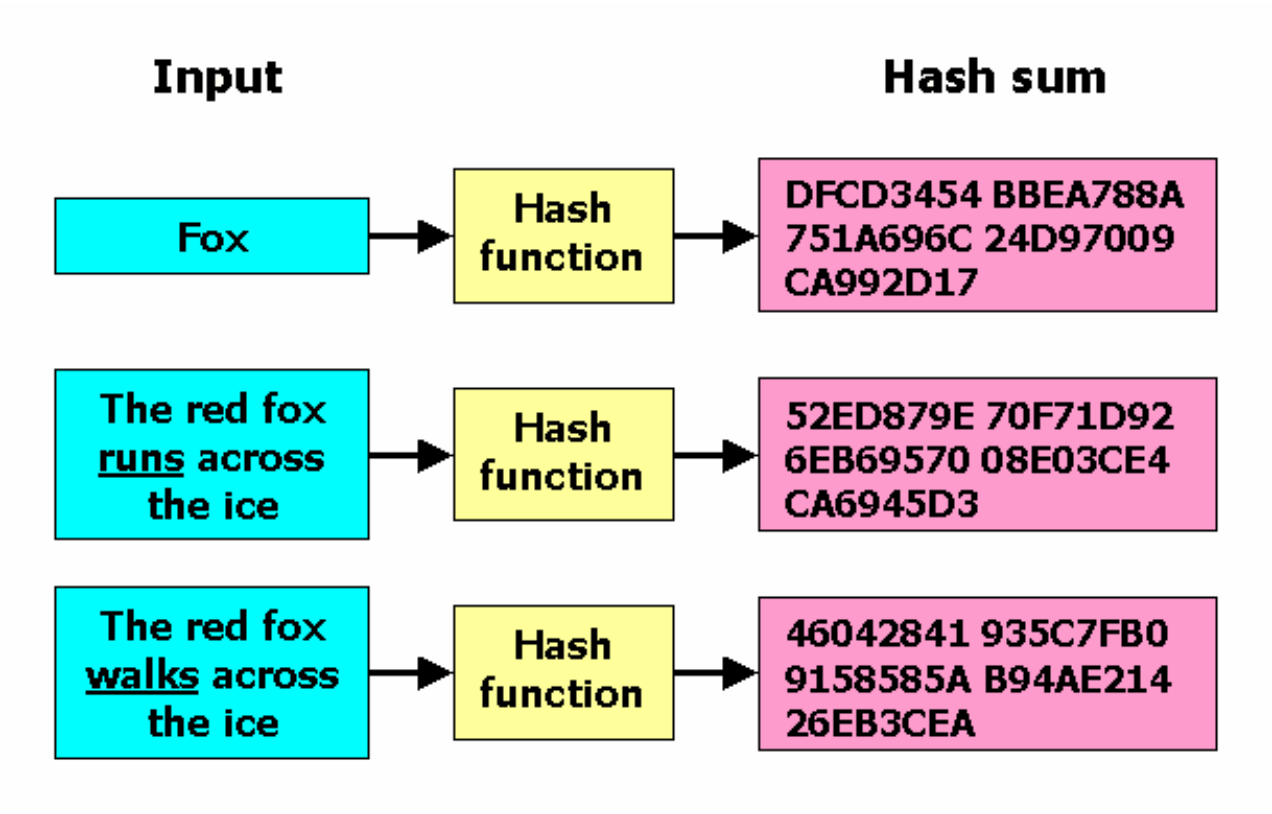
แฮชฟังก์ชัน (Hash Function) แฮชฟังก์ชันเป็นกระบวนการทางคณิตศาสตร์ที่รับข้อมูลอินพุตและแปลงมันเป็นค่าแฮชที่มีขนาดคงที่ (fixed-size) ซึ่งเรียกว่าค่าแฮชหรือแค่แฮช ค่าแฮชที่สร้างขึ้นจะมีขนาดเท่ากันไม่ว่าข้อมูลอินพุตจะยาวเท่าไร แฮชฟังก์ชันมีคุณสมบัติที่สำคัญคือถ้าข้อมูลอินพุตไม่เปลี่ยนแปลงเลย ค่าแฮชที่สร้างขึ้นจะเหมือนเดิมเสมอ (Deterministic) และถ้าข้อมูลอินพุตเปลี่ยนแปลงแม้เพียงนิดเดียว ค่าแฮชจะเปลี่ยนไปอย่างสิ้นเชิง แต่ควรจะเป็นกระบวนการสร้างค่าแฮชที่เร็วและคงที่ ตัวอย่างของแฮชฟังก์ชันที่นิยมได้แก่ SHA-256, SHA-1, MD5 เป็นต้น โดยการทำงานของการเข้ารหัสหรือแฮช (Hash) ดังภาพที่ 2.2



ภาพที่ 2.2 ตัวอย่างของการแปลงค่าแฮช (Hash) เบื้องต้น

อัลกอริทึมแฮช (Hash Algorithm) อัลกอริทึมแฮชคือขั้นตอนและวิธีการที่ใช้ในการคำนวณค่าแฮชจากข้อมูลอินพุต อัลกอริทึมแฮชรับข้อมูลอินพุตและทำหลายกระบวนการทางคณิตศาสตร์ต่าง ๆ เพื่อสร้างค่าแฮช

อัลกอริทึมแฮชควรมีคุณสมบัติที่สำคัญเช่นความปลอดภัยจากการชน (Collision Resistance) ซึ่งหมายถึงความยากในการหาข้อมูลอินพุตสองข้อมูลที่มีค่าแฮชเหมือนกัน ตัวอย่างของอัลกอริทึมแฮชที่นิยมได้แก่ SHA-256 (ซึ่งใช้แฮชฟังก์ชัน SHA-256) และ SHA-1 (ซึ่งใช้แฮชฟังก์ชัน SHA-1) เป็นต้น ดังภาพmที่ 2.3



ภาพที่ 2.3 ตัวอย่างการทำงานของอัลกอริทึมแฮช (Hash Algorithm) เบื้องต้น

ต่อมาการทำงานของแฮชฟังก์ชันและอัลกอริทึมแฮช

2.4.7 การทำงานของแฮชฟังก์ชัน แฮชฟังก์ชันรับข้อมูลอินพุตใด ๆ (ข้อความ, ไฟล์, ข้อมูลทุกชนิด) เป็นการโต้แย้ง (argument) แฮชฟังก์ชันจะทำการแปลงข้อมูลอินพุตเหล่านี้เป็นค่าแฮชที่มีขนาดคงที่ (fixed-size) ซึ่งเป็นสตริงของตัวเลขและตัวอักษร ข้อความแฮชที่สร้างขึ้นจะมีความยากต่อการถอดอ่านกลับเป็นข้อมูลอินพุตเดิมแบบที่ไม่ใช่ธรรมดา ความสำคัญของแฮชฟังก์ชันคือความปลอดภัยและความเร็วในการสร้างค่าแฮช

2.4.8 การทำงานของอัลกอริทึมแฮช อัลกอริทึมแฮชเป็นชุดของขั้นตอนทางคณิตศาสตร์และวิธีการที่ใช้ในการคำนวณค่าแฮช. อัลกอริทึมแฮชรับข้อมูลอินพุตและใช้ขั้นตอนทางคณิตศาสตร์เพื่อแปลงข้อมูลเหล่านี้เป็นค่าแฮชที่มีขนาดคงที่ อัลกอริทึมแฮชมีคุณสมบัติที่สำคัญเช่นความปลอดภัยจากการชน (collision-resistant) ซึ่งหมายถึงความยากในการหาข้อมูลอินพุตสองข้อมูลที่มีค่าแฮชเหมือนกัน ตัวอย่างของอัลกอริทึมแฮชเช่น SHA-256 มีขั้นตอนทางคณิตศาสตร์ที่ซับซ้อนเพื่อสร้างค่าแฮช

ดังนั้นจากที่อธิบายมาทั้งหมดก่อนหน้าของแฮชฟังก์ชันและอัลกอริทึมแฮชนั้นนำมา ยกตัวอย่างการใช้งาน

- การเก็บรหัสผ่าน ระบบรักษาความปลอดภัยจะเก็บรหัสผ่านของผู้ใช้เป็นค่าแฮชแทนข้อความรหัสผ่านที่เดิม เมื่อผู้ใช้ป้อนรหัสผ่านเพื่อเข้าสู่ระบบ ระบบจะทำการแปลงรหัสผ่านที่ผู้ใช้ป้อนเข้ามาเป็นค่าแฮช แล้วเปรียบเทียบกับค่าแฮชที่ถูกเก็บไว้ ถ้าตรงกันแสดงว่ารหัสผ่านถูกต้อง

- การตรวจสอบความถูกต้องของไฟล์ โปรแกรมดาวน์โหลดไฟล์จากอินเทอร์เน็ตสามารถใช้แฮชเพื่อตรวจสอบว่าไฟล์ที่ดาวน์โหลดมาไม่ถูกแก้ไขหรือเสียหายระหว่างการถ่ายโอน โดยค่าแฮชของไฟล์ที่ดาวน์โหลดมาจะต้องตรงกับค่าแฮชที่รายงานในแหล่งที่มา

- การตรวจสอบความถูกต้องของข้อมูลในฐานข้อมูล ในระบบฐานข้อมูลที่เก็บข้อมูลสำคัญ เช่นรายชื่อลูกค้า รายการสินค้า เป็นต้น อัลกอริทึมแฮชสามารถใช้ในการตรวจสอบความถูกต้องของข้อมูล และป้องกันการปลอมแปลงข้อมูล

- การตรวจสอบความถูกต้องของไฟล์ในระบบปฏิบัติการ ระบบปฏิบัติการอาจใช้อัลกอริทึมแฮชเพื่อตรวจสอบว่าไฟล์บางไฟล์ไม่ถูกเปลี่ยนแปลงโดยไม่ได้รับอนุญาต

อัลกอริทึมแฮช (Hash algorithm) มีการใช้งานหลากหลายในด้านความปลอดภัยและการรักษาความลับการตรวจสอบความถูกต้องของข้อมูล โดยผ่านกระบวนการแฮชฟังก์ชัน (Hash function) เพื่อการตรวจสอบความเป็นเจ้าของข้อมูล และอื่น ๆ อัลกอริทึมแฮชที่ใช้กันอย่างแพร่หลายรวมถึง MD5, SHA-1, SHA-256 เป็นต้น มักถูกใช้ในการทำแฮชเพื่อตรวจสอบความถูกต้องของข้อมูลและในการรักษาความปลอดภัยในหลายระบบ แต่ควรระมัดระวังเรื่องความปลอดภัยและเลือกอัลกอริทึมที่เหมาะสมกับการใช้งานและคุณสมบัติของอัลกอริทึมแฮช (Hash algorithm) เพิ่มเติมคือ

2.4.9 ความเร็ว อัลกอริทึมแฮชต้องมีประสิทธิภาพในการทำงานเพื่อให้สามารถคำนวณแฮชเวิร์ดได้เร็วที่สุด โดยไม่ขึ้นอยู่กับขนาดของข้อมูลที่นำเข้า

2.4.10 ความยากในการกลับคืนข้อมูล อัลกอริทึมแฮชจะแปลงข้อมูลต้นฉบับให้เป็นแฮชเวิร์ด และไม่สามารถกลับคืนข้อมูลเดิมได้จากแฮชเวิร์ด นั่นคือคุณไม่สามารถคำนวณข้อมูลต้นฉบับจากแฮชเวิร์ดได้

2.4.11 ความแปรปรวน การเปลี่ยนแปลงเล็กน้อยในข้อมูลต้นฉบับจะส่งผลให้แฮชเวิร์ดเปลี่ยนแปลงอย่างมาก และข้อมูลที่มีการเปลี่ยนแปลงแค่น้อยก็จะมีแฮชเวิร์ดที่แตกต่างกันอย่างชัดเจน

2.4.12 ความยากในการทำแฮชเวิร์ดที่ซ้ำกัน: อัลกอริทึมแฮชควรมีความยากในการคำนวณแฮชเวิร์ดที่ซ้ำกันสำหรับข้อมูลที่แตกต่างกัน ซึ่งหมายความว่าควรจะมีความน้อยที่ข้อมูลที่แตกต่างกันจะมีแฮชเวิร์ดเหมือนกัน

2.5 เน็ตเวิร์คซ็อกเก็ต

เน็ตเวิร์คโซเก็ต (Network Socket) เป็นแนวคิดและเทคโนโลยีที่สำคัญในการสื่อสารผ่านเครือข่ายคอมพิวเตอร์ โดยเฉพาะในระบบเครือข่าย TCP/IP ซึ่งเป็นโปรโตคอล (Protocol) หลักในอินเทอร์เน็ตและ

เครือข่ายคอมพิวเตอร์ที่ใช้กันทั่วไปในปัจจุบัน เน็ตเวิร์คโซเก็ตมีความหมายคือ แนวคิดและมาตรฐานการสื่อสารที่ถูกกำหนดขึ้นเพื่อให้คอมพิวเตอร์ต่าง ๆ สามารถสื่อสารกันผ่านเครือข่ายคอมพิวเตอร์ มันคล้ายกับประตูหรือช่องทางที่คอมพิวเตอร์ใช้ในการส่งและรับข้อมูลผ่านเครือข่าย ซึ่งเปรียบเสมือนกับโทรศัพท์หรือท่อทางน้ำในโลกแห่งเครือข่ายคอมพิวเตอร์

2.5.1 ประเภทของเน็ตเวิร์คโซเก็ต มีสองประเภทหลัก คือ TCP (Transmission Control Protocol) และ UDP (User Datagram Protocol)

- TCP คือโปรโตคอลที่มั่นคงและมีการจัดการข้อผิดพลาดในการสื่อสาร มักถูกใช้สำหรับการส่งข้อมูลที่ต้องการความน่าเชื่อถือและแน่นอน เช่น การส่งอีเมลและดาวน์โหลดเว็บเพจ
- UDP คือโปรโตคอลที่มีการส่งข้อมูลได้เร็วแต่ไม่มั่นคง ไม่มีการจัดการข้อผิดพลาดในการสื่อสาร มักถูกใช้สำหรับแอปพลิเคชันที่ต้องการความเร็วในการส่งข้อมูลมากกว่าความน่าเชื่อถือ เช่น การสตรีมวิดีโอและเกมออนไลน์

2.5.2 เลขที่อยู่ไอพี และหมายเลขของช่องทาง (IP Address และ Port Number) ในการสื่อสารผ่านเน็ตเวิร์คโซเก็ต แต่ละเครื่องคอมพิวเตอร์จะมีที่อยู่ IP ที่แตกต่างกัน เพื่อระบุเครื่องปลายทางของข้อมูล หมายเลขช่องทาง (Port Number) ใช้ระบุแอปพลิเคชันหรือบริการบนเครื่องปลายทางที่จะรับข้อมูล เป็นตัวเลขที่ถูกกำหนดเป็นช่วง เช่น Port 80 ใช้สำหรับ HTTP (เว็บเบราว์ซิ่ง)

2.5.3 การสร้างและเปิดเซสชัน (Session) การสื่อสารผ่านเน็ตเวิร์คโซเก็ตเริ่มต้นด้วยการสร้างเซสชัน (Session) หรือการเชื่อมต่อระหว่างเครื่องส่งและเครื่องปลายทาง การสร้าง Socket ใช้ API ที่มีอยู่ในหลายภาษาโปรแกรม เช่น Python, Java, และ C++ เมื่อ Socket ถูกสร้างและเปิดใช้งาน มันสามารถใช้สำหรับการส่งข้อมูลไปยังเครื่องปลายทางหรือรับข้อมูลจากเครื่องปลายทาง

2.5.4 การสื่อสารแบบครั้งเดียวและการสื่อสารแบบความต่อเนื่อง Socket สามารถใช้สำหรับการสื่อสารแบบครั้งเดียว (one-shot) โดยที่การสื่อสารจะเกิดขึ้นเพียงครั้งเดียวและหลังจากนั้น Socket จะปิดลง หรือสามารถใช้สำหรับการสื่อสารแบบความต่อเนื่อง (persistent) โดยที่การสื่อสารจะเกิดขึ้นเรื่อย ๆ ไปเรื่อย ๆ จนกว่าการเชื่อมต่อจะถูกปิด

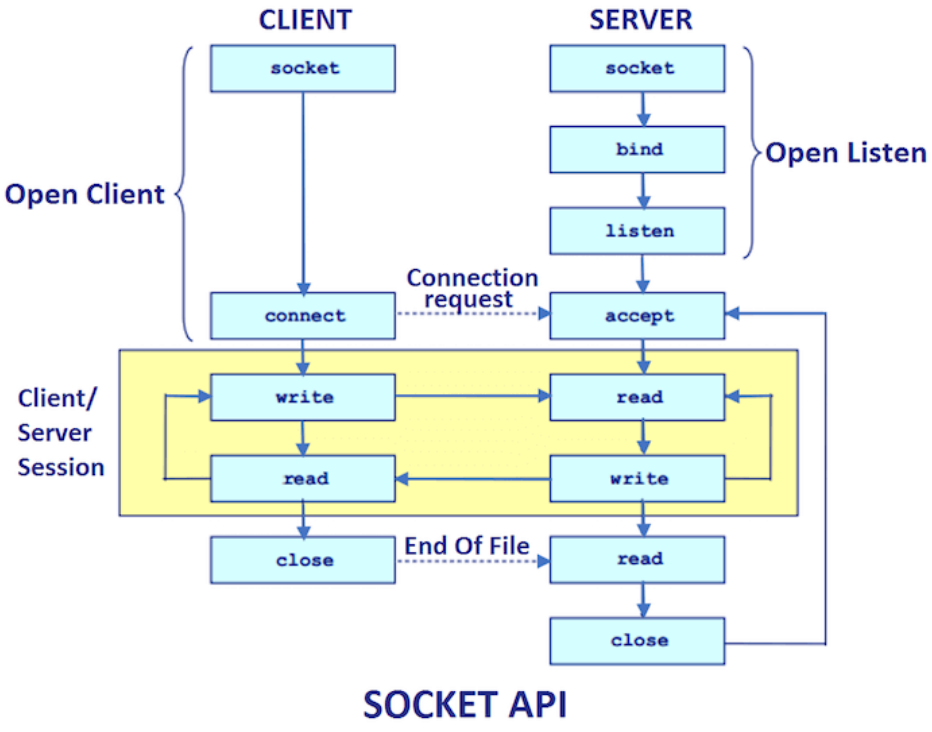
2.5.5 การติดตามสถานะการเชื่อมต่อ ในการสื่อสารแบบความต่อเนื่อง Socket สามารถติดตามสถานะการเชื่อมต่อได้ เช่น สามารถตรวจสอบว่าการส่งข้อมูลถึงเครื่องปลายทางสำเร็จหรือไม่การติดตามสถานะเป็นสิ่งสำคัญในการจัดการข้อผิดพลาดและการรับรองความถูกต้องของข้อมูล

2.5.6 ความปลอดภัย Socket สามารถใช้ในการสื่อสารระหว่างเครื่องที่มีความปลอดภัย โดยการเข้ารหัสและถอดรหัสข้อมูล (encryption/decryption) การใช้งาน SSL/TLS (Secure Sockets Layer/Transport Layer Security) สามารถเพิ่มความปลอดภัยให้กับการสื่อสารผ่าน Socket

เป็นแนวคิดและแนวทางที่ใช้ในการสื่อสารผ่านเน็ตเวิร์คโซเก็ต การเรียนรู้การใช้งาน Socket เป็นสิ่งสำคัญสำหรับนักพัฒนาซอฟต์แวร์ที่ต้องการสร้างแอปพลิเคชันที่สื่อสารผ่านเครือข่ายคอมพิวเตอร์ได้อย่างแม่นยำและปลอดภัย

ดังนั้นแล้วเน็ตเวิร์คโซเก็ต (Network Socket) เป็นแนวคิด และอินเตอร์เฟสที่ใช้ในการสื่อสารระหว่างโปรแกรมที่ทำงานบนคอมพิวเตอร์หรืออุปกรณ์เครือข่ายกับโปรแกรมหรืออุปกรณ์เครือข่ายอื่น ๆ ในเครือข่ายโดยใช้โปรโตคอลแบบแบ่งช่อง (Protocol Socket) เช่น TCP/IP หรือ UDP/IP โปรแกรมที่ทำงานบนคอมพิวเตอร์หรืออุปกรณ์เครือข่ายสามารถใช้ Network Socket เพื่อเปิดการเชื่อมต่อ (เป็นตัวแทนของการสื่อสาร) รับส่งข้อมูลระหว่างอุปกรณ์เครือข่ายที่เชื่อมต่อกัน

นอกจากนี้ Network Socket ยังให้บริการความปลอดภัยและความเชื่อถือได้ในการรับส่งข้อมูล ซึ่งสามารถใช้กำหนดโหมดการเชื่อมต่อแบบแสดงผลหรือแบบไม่แสดงผล (Blocking หรือ Non-blocking) และรูปแบบการสื่อสารที่ต้องการเช่นแบบแพร่เวลา (Broadcast) หรือแบบระบุเครื่องที่รับ (Unicast) การใช้งาน Network Socket มักใช้ในการสร้างแอปพลิเคชันเครือข่ายที่ต้องการสื่อสารระหว่างคอมพิวเตอร์หรืออุปกรณ์ที่เชื่อมต่อกัน อย่างเช่นเกมออนไลน์ที่เกี่ยวข้องกับการแลกเปลี่ยนข้อมูลระหว่างผู้เล่นหรือแอปพลิเคชันเว็บที่ต้องการรับส่งข้อมูลผ่านโปรโตคอลเครือข่าย เน็ตเวิร์คโซเก็ตจะเป็นตัวกลางที่ช่วยให้โปรแกรมสามารถสื่อสารและรับส่งข้อมูลได้ในรูปแบบที่เข้าใจง่ายและมีความยืดหยุ่นสูงในการเชื่อมต่อเครือข่าย ดังภาพ 2.1



ภาพที่ 2.4 การทำงานของโซเก็ตลูกข่าย และแม่ข่าย (Socket client and server)

2.6 การส่งไฟล์บนระบบคอมพิวเตอร์

การส่งข้อมูลบนคอมพิวเตอร์ (File Transfer) แบบข้ามเครื่องหมายถึงกระบวนการส่งข้อมูลระหว่างคอมพิวเตอร์สองเครื่องหรือมากกว่าที่ไม่อยู่ในเครือข่ายเดียวกัน นี่คือหลักการแบบเชิงลึกละเอียดเกี่ยวกับวิธีการส่งข้อมูลและเทคโนโลยีที่ใช้ในกระบวนการนี้

2.6.1 โปรโตคอลการสื่อสาร การส่งข้อมูลข้ามเครื่องส่วนใหญ่ใช้โปรโตคอลการสื่อสารเพื่อให้คอมพิวเตอร์สองเครื่องสื่อสารกันได้ โปรโตคอลเหล่านี้รวมถึง TCP/IP (Transmission Control Protocol/Internet Protocol) สำหรับการส่งข้อมูลผ่านอินเทอร์เน็ตและ UDP (User Datagram Protocol) สำหรับการสื่อสารแบบไร้การเชื่อมต่อเพียงครั้งเดียว

2.6.2 การเชื่อมต่อและการกำหนดค่าเครือข่าย เพื่อส่งข้อมูลไปยังคอมพิวเตอร์อื่น ๆ คุณจำเป็นต้องมีการเชื่อมต่อเครือข่ายระหว่างเครื่องคอมพิวเตอร์ทั้งสอง การเชื่อมต่อนี้สามารถเป็นแบบไร้สายหรือใช้สายก็ได้ นอกจากนี้ คุณต้องกำหนดค่าเครือข่ายให้กับทั้งสองเครื่องเพื่อให้สามารถสื่อสารกันได้อย่างถูกต้อง

2.6.3 พอร์ทและการรับส่งข้อมูล การส่งข้อมูลระหว่างคอมพิวเตอร์ข้ามเครื่องสามารถทำผ่านพอร์ทที่กำหนดไว้ พอร์ทเป็นช่องทางที่ใช้ในการรับและส่งข้อมูลระหว่างคอมพิวเตอร์ พอร์ทมักถูกกำหนดในรูปแบบตัวเลข เช่น พอร์ท 80 ใช้สำหรับเว็บเซิร์ฟเวอร์ HTTP

2.6.4 โปรโตคอลการส่งข้อมูล การส่งข้อมูลข้ามเครื่องอาจใช้โปรโตคอลที่แตกต่างกันตามลักษณะของข้อมูลและความต้องการเชิงพิเศษ ตัวอย่างของโปรโตคอลที่ใช้รวมถึง HTTP (Hypertext Transfer Protocol) สำหรับการส่งข้อมูลเว็บ, SMTP (Simple Mail Transfer Protocol) สำหรับการส่งอีเมล, และ FTP (File Transfer Protocol) สำหรับการถ่ายโอนไฟล์

2.6.5 การเข้ารหัสและความปลอดภัย การส่งข้อมูลข้ามเครื่องต้องใส่ความสำคัญในเรื่องความปลอดภัย เพื่อป้องกันการดักฟังและการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาต สิ่งที่สำคัญคือการใช้การเข้ารหัสเพื่อปกป้องข้อมูลขณะที่ถูกส่งผ่านเครือข่าย

ดังนั้นจากที่ได้อธิบายข้างต้น ตัวอย่างของการส่งข้อมูลแบบข้ามเครื่องได้ดังนี้

- การส่งไฟล์ผ่าน HTTP เว็บเบราว์เซอร์ของคุณเป็นเครื่องลูกข่ายและคอมพิวเตอร์เซิร์ฟเวอร์อื่น ๆ เป็นเครื่องเซิร์ฟเวอร์ คุณสามารถส่งข้อมูลจากคอมพิวเตอร์เซิร์ฟเวอร์ไปยังเครื่องลูกข่ายของคุณผ่าน HTTP โดยใช้การร้องขอ (Request) เพื่อดาวน์โหลดหรือบรรจุ (Upload) ไฟล์ขึ้น

- การส่งข้อมูลผ่านอีเมล คุณสามารถแนบไฟล์หรือข้อมูลอื่น ๆ ในอีเมลและส่งไปยังผู้รับ โดยทั่วไปแล้วอีเมลถูกส่งผ่านโปรโตคอล SMTP

- การถ่ายโอนไฟล์โดยใช้เอฟทีพี (FTP) ใช้สำหรับการถ่ายโอนไฟล์ระหว่างคอมพิวเตอร์ คุณสามารถเชื่อมต่อกับเซิร์ฟเวอร์เอฟทีพี (FTP) และส่งหรือรับไฟล์โดยใช้บัญชีเอฟทีพี (FTP)

การส่งข้อมูลแบบข้ามเครื่องมีหลายวิธีและโปรโตคอลที่ใช้กันทั่วไป ขึ้นอยู่กับวัตถุประสงค์และความต้องการของการสื่อสารระหว่างคอมพิวเตอร์การเลือกวิธีและโปรโตคอลที่เหมาะสมขึ้นอยู่กับแต่ละสถานการณ์และความต้องการในการสื่อสารโดยทางผู้วิจัยได้นำหลักการส่งข้อมูลแบบผ่านโปรโตคอลเอฟทีพี (FTP) และที่เกี่ยวกับโปรโตคอลเอฟทีพี (FTP) คือ โปรโตคอลเอสเอฟทีพี (SFTP) มาใช้งานร่วมกันดังการอธิบายเพิ่มเติมด้านล่าง

2.6.6 FTP (File Transfer Protocol) และ SFTP (SSH File Transfer Protocol) เป็นโปรโตคอลที่ใช้ในการส่งข้อมูลระหว่างคอมพิวเตอร์ผ่านเครือข่าย แต่มีความแตกต่างกันในด้านความปลอดภัยและวิธีการทำงานดังนี้

- เอฟทีพี (FTP หรือ File Transfer Protocol) เป็นโปรโตคอลสำหรับการถ่ายโอนไฟล์ระหว่างคอมพิวเตอร์ในเครือข่าย โดยใช้พอร์ต 21 ในการควบคุมการเชื่อมต่อและการส่งข้อมูล หลักการทำงาน การสื่อสารเอฟทีพี (FTP) นั้นมีขั้นตอนการทำงานเป็นแบบโปรโตคอลแบบไม่เป็นการสื่อสารแบบทันที (stateless) คือต้องเปิดการเชื่อมต่อเอฟทีพี (FTP) ก่อนแล้วค่อยๆ ส่งคำสั่งและข้อมูลต่าง ๆ ผ่านการเชื่อมต่อนี้ FTP ไม่ใช้การเข้ารหัสข้อมูลในการถ่ายโอนทำให้ข้อมูลที่ถูกส่งไปมาในรูปแบบข้อความ (plaintext) และมีความเสี่ยงที่ข้อมูลจะถูกดักฟังหรือโจมตีได้

- เอสเอฟพีที (SFTP หรือ SSH File Transfer Protocol) เป็นโปรโตคอลสำหรับการถ่ายโอนไฟล์ที่ปลอดภัยผ่านเครือข่าย โดยใช้เอสเอสเอช (SSH หรือ Secure Shell) เป็นเสมือนชั้นการรักษาความปลอดภัย หลักการทำงาน SFTP ใช้การเข้ารหัสข้อมูลและการสื่อสารผ่านการเชื่อมต่อ SSH ทำให้ข้อมูลที่ถูกส่งไปมีความปลอดภัยและเข้ารหัส การเชื่อมต่อ SFTP มีระบบการยืนยันและการสร้างเซสชัน (session) เพื่อส่งข้อมูลอย่างปลอดภัย

เหตุผลที่ SFTP เกิดขึ้นมาเนื่องจากความต้องการในความปลอดภัยของการสื่อสารและการถ่ายโอนไฟล์ผ่านเครือข่าย เนื่องจาก FTP มีปัญหาในด้านความปลอดภัย เนื่องจากข้อมูลที่ถูกส่งไปมีความเสี่ยงที่จะถูกดักฟังหรือถูกโจรกรรมหรือการแอบเข้าถึง (Hack) อันนี้เป็นตัวอย่างการทำงานของทั้งสองโปรโตคอล

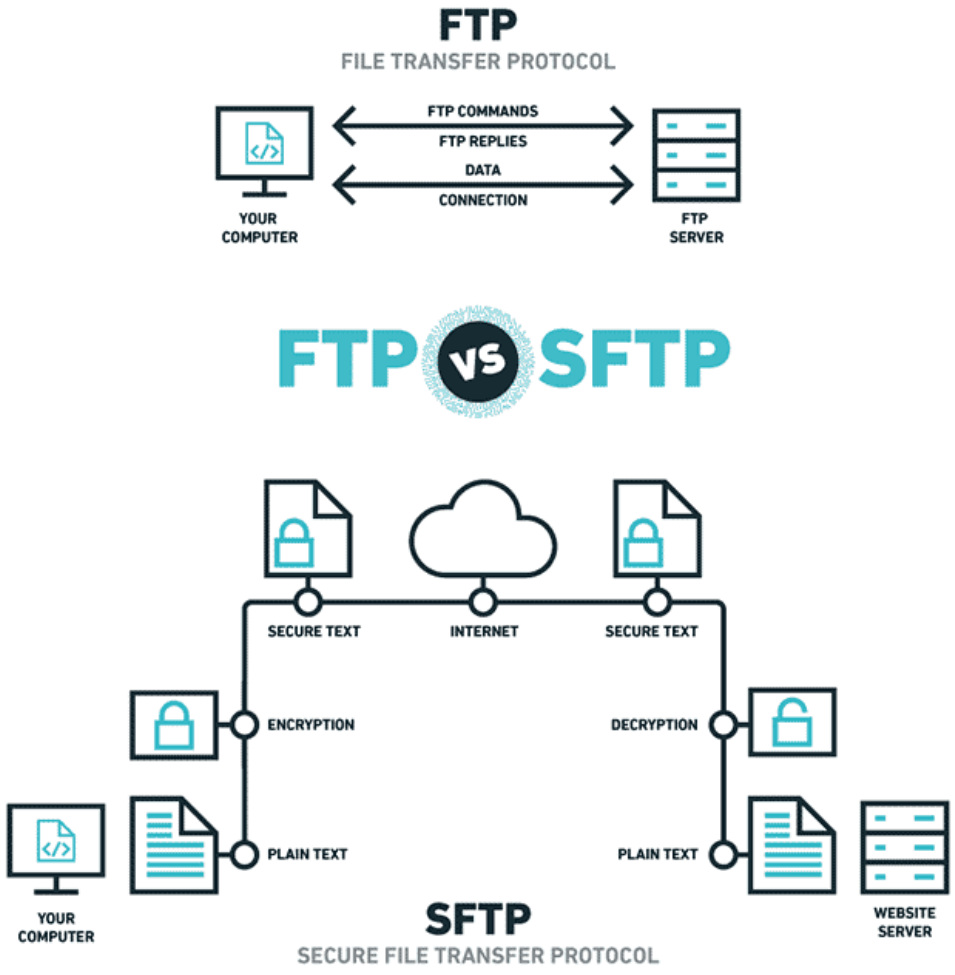
ตัวอย่างการเปรียบเทียบของทั้ง 2 รูปแบบการส่งข้อมูลเอฟทีพี และเอสเอฟทีพี (FTP and SFTP) การถ่ายโอนไฟล์ผ่าน FTP ดังนี้

- คอมพิวเตอร์เอ (A) เปิดการเชื่อมต่อเอฟทีพี (FTP) ไปยังคอมพิวเตอร์บี (B) ผ่านหมายเลขช่องทางหรือพอร์ต (Port) 21
- คอมพิวเตอร์เอ (A) ส่งคำสั่ง (FTP) เพื่อร้องขอการถ่ายโอนไฟล์ไปยังคอมพิวเตอร์บี (B)
- คอมพิวเตอร์บี (B) ตอบกลับด้วยการอนุญาตและรอรับข้อมูล
- คอมพิวเตอร์เอ (A) ส่งข้อมูลไฟล์ไปยังคอมพิวเตอร์บี (B) ในรูปแบบข้อความ (plaintext)
- คอมพิวเตอร์บี (B) รับข้อมูลและบันทึกไฟล์

อีกหนึ่งตัวอย่างในส่วนของการถ่ายโอนไฟล์ผ่านเอสเอฟทีพี (SFTP)

- คอมพิวเตอร์เอ (A) เปิดการเชื่อมต่อเอสเอสเอช (SSH) ไปยังคอมพิวเตอร์บี (B) ผ่านหมายเลขช่องทางหรือพอร์ต (Port) 22
- คอมพิวเตอร์เอ (A) ส่งคำสั่งเอสเอฟทีพี (SFTP) เพื่อร้องขอการถ่ายโอนไฟล์ไปยังคอมพิวเตอร์บี (B)
- คอมพิวเตอร์บี (B) ตอบกลับด้วยการอนุญาตและรอรับข้อมูล
- คอมพิวเตอร์เอ (A) ส่งข้อมูลไฟล์ไปยังคอมพิวเตอร์บี (B) โดยเข้ารหัสข้อมูลและสื่อสารผ่านการเชื่อมต่อเอสเอสเอช (SSH)
- คอมพิวเตอร์บี (B) รับข้อมูลและบันทึกไฟล์ที่ถูกถ่ายโอนแบบปลอดภัย

สรุปแล้วการถ่ายโอนไฟล์ผ่าน FTP เป็นการสื่อสารแบบไม่ปลอดภัยที่ใช้โปรโตคอลเดิม ในขณะที่ SFTP เป็นการสื่อสารที่มีระบบความปลอดภัยสูงและใช้การเข้ารหัสข้อมูลในการถ่ายโอน การเลือกใช้โปรโตคอลขึ้นอยู่กับความต้องการในความปลอดภัยและความสะดวกในการใช้งานข้อมูลข้ามเครื่องดังภาพที่ 2.5



ภาพที่ 2.5 อธิบายการทำงานของทั้งสองโปรโตคอลส่งไฟล์ข้อมูล (FTP and SFTP)

อธิบายเพิ่มเติมในส่วนของเอสเอสเอช (SSH หรือ Secure Shell) เป็นโปรโตคอลควบคุมระยะไกลและเป็นระบบสื่อสารที่ปลอดภัยที่ใช้ในการเชื่อมต่อและจัดการคอมพิวเตอร์ผ่านเครือข่ายในโหมดที่เข้ารหัสและป้องกันข้อมูลระหว่างการสื่อสาร นี่คือหลักการทำงานของเอสเอสเอช (SSH) และหน้าที่ของแต่ละส่วน

การพิสูจน์ตัวตน (Authentication) คือ การเชื่อมต่อเอสเอสเอช (SSH) ต้องมีการพิสูจน์ตัวตนเพื่อให้ความมั่นคงและป้องกันการเข้าถึงจากบุคคลที่ไม่ใช่. SSH รองรับหลายวิธีการพิสูจน์ตัวตน เช่น การใช้รหัสผ่าน, การใช้งานกุญแจสาธารณะและกุญแจส่วนตัว (public-key authentication), หรือวิธีการพิสูจน์ตัวตนอื่น ๆ ที่กำหนดเอง ตัวอย่างการเชื่อมต่อเอสเอสเอช (SSH) ไปยังเซิร์ฟเวอร์โดยใช้การพิสูจน์ตัวตนด้วยกุญแจสาธารณะและกุญแจส่วนตัว.

การเข้ารหัสข้อมูล (Encryption) คือ ใช้การเข้ารหัสข้อมูลในระหว่างการสื่อสารเพื่อป้องกันข้อมูลไม่ให้ถูกอ่านหรือแก้ไขโดยบุคคลที่ไม่ได้รับอนุญาต. ข้อมูลที่ส่งไปมาระหว่างคอมพิวเตอร์จะถูกเข้ารหัสก่อนการส่งและถูกถอดรหัสที่ปลายรับ ตัวอย่าง ข้อมูลที่ส่งผ่านการเชื่อมต่อเอสเอสเอช (SSH) จะถูกเข้ารหัสและถ้ามีการดักฟังการสื่อสาร, ผู้ดักฟังจะไม่สามารถอ่านข้อมูลได้

การแลกเปลี่ยนกุญแจ (Key Exchange) คือ การเริ่มต้นการเชื่อมต่อเอสเอสเอช (SSH) จะมีกระบวนการแลกเปลี่ยนกุญแจเพื่อใช้ในการเข้ารหัสและถอดรหัสข้อมูล. กระบวนการนี้จะมีการสร้างกุญแจร่วม (session key) ที่ใช้เฉพาะในการสื่อสารระหว่างเครื่องคอมพิวเตอร์ ตัวอย่าง การเริ่มต้นการเชื่อมต่อ SSH จะมีการแลกเปลี่ยนกุญแจแบบ Diffie-Hellman หรือ ECDH (Elliptic Curve Diffie-Hellman) เพื่อกำหนดกุญแจร่วม

การจัดการเซสชัน (Session Managemen) คือ บริหารจัดการเซสชันของการเชื่อมต่อ รวมถึงการเข้ารหัสและถอดรหัสข้อมูลในเซสชันนั้น ๆ และการควบคุมการเข้าถึงทรัพยากรของเครื่องคอมพิวเตอร์ที่เชื่อมต่อ ตัวอย่าง การจัดการเซสชันเอสเอสเอช (SSH) ระหว่างผู้ใช้และเซิร์ฟเวอร์รวมถึงการตั้งค่าการเข้าถึงทรัพยากรบนเครื่องเซิร์ฟเวอร์

การส่งพอร์ต (Port Forwarding) คือ สามารถใช้ในการส่งพอร์ต (port forwarding) เพื่อเชื่อมต่อเครื่องคอมพิวเตอร์ระยะไกลกับทรัพยากรบนเครื่องคอมพิวเตอร์ในเครือข่ายภายใน ตัวอย่าง ผู้ใช้เอสเอสเอช (SSH) สามารถสร้างการส่งพอร์ตเพื่อเชื่อมต่อกับเครื่องเซิร์ฟเวอร์ฐานข้อมูลในเครือข่ายภายในผ่านการเชื่อมต่อเอสเอสเอช (SSH)

ควบคุมการเข้าถึง (Access Control) คือ สามารถใช้ในการควบคุมการเข้าถึงและการอนุญาตให้เฉพาะผู้ใช้ที่มีสิทธิ์เข้าถึงบนเครื่องเซิร์ฟเวอร์ ตัวอย่าง การกำหนดกุญแจการเข้าถึงที่มีสิทธิ์ในไฟล์การตั้งค่าเอสเอสเอช (SSH) เพื่อควบคุมผู้ใช้ที่มีสิทธิ์เข้าถึงบนเครื่องเซิร์ฟเวอร์

ดังนั้นแล้วเอสเอสเอช (SSH) เป็นโปรโตคอลที่สำคัญในการประกันความปลอดภัยและการควบคุมการเข้าถึงในระบบคอมพิวเตอร์ระยะไกล โดยมีหลักการทำงานและคุณสมบัติที่ช่วยให้ข้อมูลและการสื่อสารระหว่างเครื่องคอมพิวเตอร์ปลอดภัยและเชื่อถือได้

2.7 เครื่องมือวิเคราะห์แพ็คเก็ตอินเทอร์เน็ต

เครื่องมือวิเคราะห์แพ็คเก็ตอินเทอร์เน็ต (TCPDUMP) หรือ ตัววิเคราะห์โปรโตคอล (Protocol Analyzer หรือ Sniffer) เป็นเครื่องมือหรือโปรแกรมคอมพิวเตอร์ที่ใช้ในการวิเคราะห์และตรวจสอบการสื่อสารผ่านเครือข่ายคอมพิวเตอร์ เพื่อเข้าใจหรือตรวจสอบโปรโตคอลการสื่อสาร เหตุผลและหลักการทำงานของตัววิเคราะห์โปรโตคอล (Sniffer) มีดังนี้

2.7.1 เหตุผลที่เกิดขึ้น

- ตรวจสอบปัญหา สามารถใช้ตัววิเคราะห์โปรโตคอล (Sniffer) เพื่อตรวจสอบปัญหาในเครือข่าย เช่น การสื่อสารที่ผิดพลาดหรือปริมาณข้อมูลที่มากเกินไป ซึ่งช่วยในการแก้ไขปัญหาได้อย่างมีประสิทธิภาพ

- ควบคุมความปลอดภัยตัววิเคราะห์โปรโตคอล (Sniffer) สามารถใช้ในการตรวจจับการบุกรุกหรือการโจรกรรมหรือแอบเข้าถึง (Hack) เข้าสู่ระบบคอมพิวเตอร์ เพื่อควบคุมความปลอดภัยของเครือข่าย

- วิเคราะห์การใช้งานเครือข่าย ใช้ในการตรวจสอบการใช้งานเครือข่าย เพื่อวิเคราะห์การใช้งานและปรับปรุงประสิทธิภาพของระบบ

2.7.2 หลักการทำงาน การตรวจสอบแพ็คเก็ต (Packet Inspection) ตัววิเคราะห์โปรโตคอล (Sniffer) จะตรวจจับและบันทึกแพ็คเก็ตข้อมูลที่ส่งผ่านเครือข่าย โดยรวบรวมข้อมูลเช่นแหล่งที่มาและปลายทาง, ข้อมูลในแพ็คเก็ต, พอร์ตต้นทางและปลายทาง, และเวลาการส่งข้อมูล

- วิเคราะห์โครงสร้างโปรโตคอล (Sniffer) จะวิเคราะห์โครงสร้างของข้อมูลในแพ็คเก็ตเพื่อรู้ว่ามีการใช้โปรโตคอลใด ๆ ในการสื่อสาร เช่น HTTP, FTP, TCP, UDP, เป็นต้น

- การแสดงผล ข้อมูลที่ถูกตรวจจับและวิเคราะห์จะถูกแสดงผลในรูปแบบที่เข้าใจง่ายเช่นแสดงข้อมูลในรูปแบบของตารางหรือกราฟ

2.7.3 การทำงานของตัววิเคราะห์โปรโตคอล (Sniffer)

- การจับแพ็คเก็ตตัววิเคราะห์โปรโตคอล (Sniffer) จะเปิดการที่จับข้อมูลบนเครือข่ายและตรวจจับแพ็คเก็ตข้อมูลที่ถูกส่งผ่าน

- การตรวจสอบแพ็คเก็ต หลังจากจับแพ็คเก็ต Sniffer จะตรวจสอบโครงสร้างและเนื้อหาของแพ็คเก็ตเพื่อระบุประเภทของโปรโตคอลและข้อมูลที่อยู่ในนั้น

- การแสดงผล ข้อมูลที่ถูกตรวจจับและวิเคราะห์จะถูกแสดงผลให้ผู้ใช้ดู เพื่อให้เข้าใจการสื่อสารที่เกิดขึ้นในเครือข่าย

เพิ่มเติมตัววิเคราะห์โปรโตคอล (Sniffer) ในการตรวจสอบและวิเคราะห์การสื่อสารระหว่างอุปกรณ์ในเครือข่ายอินเทอร์เน็ต โดยหมายถึงการอ่านข้อมูลที่ถูกส่งผ่านเครือข่าย เพื่อทราบรายละเอียดเกี่ยวกับการสื่อสารและการตรวจสอบปัญหาหรือระบุความเสี่ยงทางความปลอดภัยการทำงานของตัววิเคราะห์โปรโตคอลมีลักษณะดังนี้

2.6.1 การตรวจสอบแพ็คเก็ต (Packet Sniffing) คือ ตัววิเคราะห์โปรโตคอลจะอ่านแพคเก็ตข้อมูลที่ถูกส่งผ่านเครือข่าย โดยรวบรวมข้อมูลเช่น ที่อยู่ต้นทางและปลายทาง ข้อมูลเสริม เนื้อหาข้อมูล และข้อมูลการควบคุมอื่น ๆ

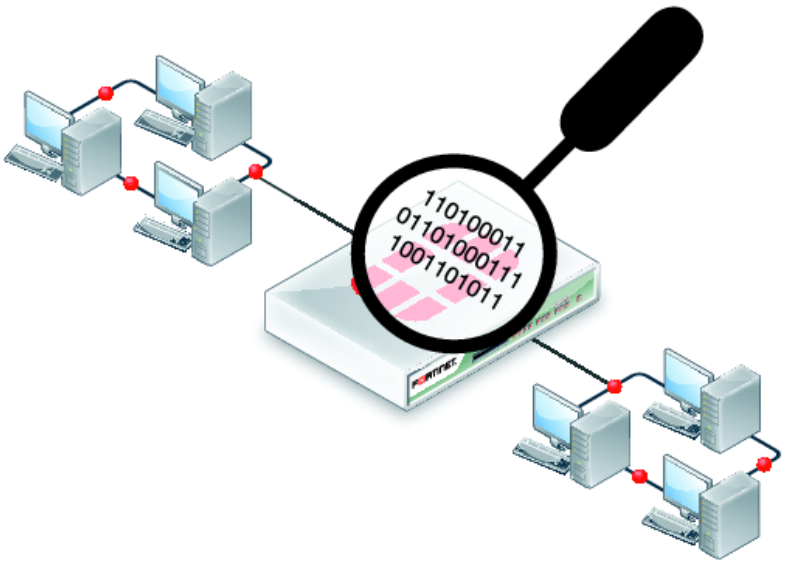
2.6.2 การวิเคราะห์โปรโตคอล (Protocol Analysis) คือ ตัววิเคราะห์โปรโตคอลจะวิเคราะห์และตีความข้อมูลที่ถูกตรวจสอบเพื่อรู้เกี่ยวกับโครงสร้างของข้อมูลและโปรโตคอลที่ใช้ในการสื่อสาร โดยการตรวจสอบเฟรมข้อมูล การเปรียบเทียบเทมเพลต และสถานะของการสื่อสาร

2.6.3 การตรวจสอบปัญหาและความเสี่ยง คือ ตัววิเคราะห์โปรโตคอลสามารถตรวจสอบปัญหาหรือความเสี่ยงทางความปลอดภัยที่เกี่ยวข้องกับการสื่อสาร อาทิเช่นการตรวจสอบการแฮก (hacking) หรือการโจมตีแบบดอส (DoS attacks) เพื่อช่วยในการตรวจจับและแก้ไขปัญหาต่าง ๆ

2.6.4 การบันทึกข้อมูล: ตัววิเคราะห์โปรโตคอลสามารถบันทึกและเก็บรวบรวมข้อมูลที่ถูกตรวจสอบได้ เพื่อให้สามารถดูข้อมูลที่ถูกตรวจสอบได้ในภายหลัง หรือนำไปใช้ในการวิเคราะห์เพิ่มเติม

ตัววิเคราะห์โปรโตคอลมีการใช้งานในหลายด้าน เช่น การตรวจสอบและวิเคราะห์การสื่อสารในเครือข่ายเพื่อความปลอดภัยและปัญหาทางเทคนิค, การวิเคราะห์และตรวจสอบปัญหาเครือข่าย เช่น ปัญหาการส่งข้อมูลที่ผิดพลาดหรือช้า, และการพัฒนาและทดสอบโปรโตคอลใหม่หรือการปรับปรุงโปรโตคอลที่มีอยู่

สำหรับการใช้งานตัววิเคราะห์โปรโตคอล จำเป็นต้องมีความระมัดระวังและคำนึงถึงความเป็นส่วนตัวและกฎหมายที่เกี่ยวข้อง เนื่องจากมีความสามารถในการตรวจสอบและอ่านข้อมูลที่ถูกส่งผ่านเครือข่าย ควรใช้งานตามเป้าหมายและในขอบเขตที่ถูกต้องเท่านั้น ดังนั้น ตัวอย่างของ Sniffer ที่มีชื่อเสียงรวมถึง Wireshark, tcpdump, และ Microsoft Network Monitor ซึ่งใช้ในการวิเคราะห์และตรวจสอบการสื่อสารในเครือข่าย และเป็นเครื่องมือสำคัญในการดำเนินการด้านความปลอดภัยและการดูแลระบบเครือข่ายในองค์กรและองค์กรต่าง ๆ ทั่วโลก



ภาพที่ 2.6 ตัวอย่างของเครื่องมือวิเคราะห์แพ็คเก็ตอินเทอร์เน็ต (TCPDUMP)

2.8 ช่องทางในการส่งผ่านข้อมูลประมวลผล

ช่องทางในการส่งผ่านข้อมูลประมวลผล (Thread) คอมพิวเตอร์คือกลไกระดับโปรแกรมที่ทำให้โปรแกรมสามารถทำงานพร้อม ๆ กันได้ในคอมพิวเตอร์หนึ่งเครื่อง โดยมีหน้าที่ในการแบ่งเวลาประมวลผลของซีพียู (CPU) ให้กับโปรแกรมหลาย ๆ ตัวที่ทำงานพร้อม ๆ กัน ช่องทางในการส่งผ่านข้อมูลประมวล (Thread) ช่วยเพิ่มประสิทธิภาพในการประมวลผลข้อมูลและทำให้การทำงานของโปรแกรมเป็นไปอย่างมีประสิทธิภาพ

2.8.1 หลักการของ ช่องทางในการส่งผ่านข้อมูลประมวล (Thread) ดังนี้

- การทำงานพร้อม ๆ กัน (Concurrency) ช่องทางในการส่งผ่านข้อมูลประมวลผล (Threads) ทำงานพร้อม ๆ กันในเวลาเดียวกันบนเครื่องคอมพิวเตอร์เพื่อทำให้โปรแกรมทำงานได้เร็วขึ้น

- การแชร์ข้อมูล (Sharing) ช่องทางในการส่งผ่านข้อมูลประมวล (Threads) สามารถแชร์ข้อมูลและแหล่งทรัพยากรต่าง ๆ กันได้ เช่น แชร์ตัวแปร, ไฟล์, หรือคอนเน็คชัน (Connection)

- การประสานงาน (Synchronization) ช่องทางในการส่งผ่านข้อมูลประมวล (Threads) ต้องถูกประสานงานให้ทำงานพร้อม ๆ กันได้โดยไม่เกิดข้อผิดพลาด เช่น การใช้ ล็อค, เซมาฟอร์ส (Locks, Semaphores), หรือ มิวเท็กซ์ (Mutexes)

- การสื่อสาร (Communication) ช่องทางในการส่งผ่านข้อมูลประมวล (Threads) ต้องสื่อสารกันในกระบวนการทำงาน เพื่อให้ข้อมูลถูกส่งไปมาระหว่าง ช่องทางในการส่งผ่านข้อมูลประมวล (Threads)

2.8.2 การทำงานของ Threads แบ่งออกเป็น 2 รูปแบบคือ

- เครื่องคอมพิวเตอร์แบบแกนเดี่ยว (Single-Core) ในเครื่องคอมพิวเตอร์ที่มีเพียงแกน (Core) เดียว, ช่องทางในการส่งผ่านข้อมูลประมวล (Threads) จะถูกทำงานโดยใช้หลักการของการสลับบริบท (Context Switching) ที่ช่วยให้ทุก Thread ได้รับโอกาสทำงาน

- เครื่องคอมพิวเตอร์แบบหลายแกน (Multi-Core) ในเครื่องคอมพิวเตอร์ที่มีหลายแกน (Core), ช่องทางในการส่งผ่านข้อมูลประมวล (Threads) แต่ละตัวสามารถทำงานพร้อม ๆ กันบนแกน (Core) ต่าง ๆ โดยไม่ต้องใช้การสลับบริบท (Context Switching) ทุกครั้ง

2.9 หลักการทำงานให้สอดคล้องกับพระราชบัญญัติการคุ้มครองข้อมูลส่วนบุคคล

การทำงานให้สอดคล้องกับกฎหมายข้อมูลส่วนบุคคล (PDPA) ของประเทศไทยควรปฏิบัติตามหลักการและขั้นตอนต่าง ๆ ดังนี้

2.8.1 การรับรู้และการตระหนักถึงกฎหมายข้อมูลส่วนบุคคล (Awareness) องค์กรควรสร้างความตระหนักในพนักงานทุกคนเกี่ยวกับกฎหมายข้อมูลส่วนบุคคล (PDPA) และควรมีการอบรมพนักงานในเรื่องนี้

2.8.2 การกำหนดผู้รับผิดชอบ (Data Controller) องค์กรควรระบุผู้รับผิดชอบในการดำเนินงานที่เกี่ยวข้องกับข้อมูลส่วนบุคคลและความเป็นส่วนตัว

2.8.3 การเก็บรักษาข้อมูล (Data Collection) การเก็บรักษาข้อมูลส่วนบุคคลควรมีวัตถุประสงค์ชัดเจนและไม่ควรเก็บเกินขอบเขตของวัตถุประสงค์นั้น ๆ และควรให้ข้อมูลแก่เจ้าของข้อมูลทราบถึงวัตถุประสงค์และวิธีการใช้ข้อมูล

2.8.4 การประมวลผลข้อมูล (Data Processing) การประมวลผลข้อมูลส่วนบุคคลควรเป็นไปตามวัตถุประสงค์ที่ระบุไว้ และควรมีมาตรการความปลอดภัยที่เพียงพอในการปกป้องข้อมูล

2.8.5 การขออนุญาต (Consent) ในบางกรณี การขออนุญาตจากเจ้าของข้อมูลส่วนบุคคลอาจจำเป็น โดยองค์กรควรแจ้งเจ้าของข้อมูลเกี่ยวกับวัตถุประสงค์และวิธีการใช้ข้อมูลอย่างชัดเจน

2.8.6 การแจ้งให้ทราบ (Notice) องค์กรควรระบุข้อมูลที่จะถูกเก็บรักษาและใช้งานในข้อตกลง PDPA และต้องแจ้งเจ้าของข้อมูลถึงข้อมูลเหล่านี้

2.8.7 สิทธิของบุคคล (Data Subject Rights) เจ้าของข้อมูลส่วนบุคคลมีสิทธิ์ในการเข้าถึงข้อมูล, ให้แก้ไข, ร้องเรียน, และให้ลบข้อมูลของตนเอง

2.8.8 ความปลอดภัยของข้อมูล (Data Security): องค์กรควรใช้มาตรการเพื่อปกป้องข้อมูลส่วนบุคคลไม่ให้ถูกเข้าถึงหรือเปิดเผยโดยไม่ได้รับอนุญาต

2.8.9 การส่งข้อมูลต่างประเทศ (Cross-Border Data Transfer) หากมีการส่งข้อมูลไปต่างประเทศ องค์กรควรปฏิบัติตามกฎหมายที่เกี่ยวข้องและต้องรักษาความปลอดภัย

2.8.10 การควบคุมองค์กรย่อย (Subsidiary Control) หากองค์กรมีองค์กรย่อยหรือฝ่ายงานที่เกี่ยวข้องจะต้องมีการควบคุมให้สอดคล้องกับกฎหมาย

2.8.11 การควบคุมและการตรวจสอบ (Compliance and Audit) องค์กรควรทำการตรวจสอบและตรวจสอบการปฏิบัติตามกฎหมายข้อมูลส่วนบุคคลเพื่อดูว่ากฎหมายถูกปฏิบัติตามอย่างถูกต้อง

2.8.12 การรายงานเหตุการณ์ข้อมูลส่วนบุคคล (Data Breach Reporting) หากมีการรั่วไหลของข้อมูลส่วนบุคคล องค์กรต้องรายงานให้กับหน่วยงานที่เกี่ยวข้องและผู้สูญเสียข้อมูล

2.8.13 การสร้างนโยบายและคู่มือ (Policies and Guidelines) องค์กรควรสร้างนโยบายและคู่มือเพื่อแนะนำและกำหนดวิธีการที่ควรปฏิบัติ

2.8.14 การฝึกอบรม (Training and Education) บุคลากรควรรับการฝึกอบรมเกี่ยวกับกฎหมายข้อมูลส่วนบุคคลและการปฏิบัติที่ถูกต้อง

2.8.15 การรักษาเอกสาร (Documentation): องค์กรควรรักษาเอกสารที่เกี่ยวข้องกับการปฏิบัติตามกฎหมาย

ดังนั้นจากข้อทั้งหมดที่ได้กล่าวมาข้างต้น ยกตัวอย่าง บริษัท XYZ ต้องการใช้ข้อมูลสมาชิกในการติดต่อและสร้างกิจกรรมต่าง ๆ ที่เกี่ยวข้อง องค์กรจึงจะต้องขออนุญาตจากสมาชิกและแจ้งวัตถุประสงค์และวิธีการใช้ข้อมูลให้เจ้าของข้อมูลทราบ และมีมาตรการความปลอดภัยในการรักษาข้อมูลให้ปลอดภัย

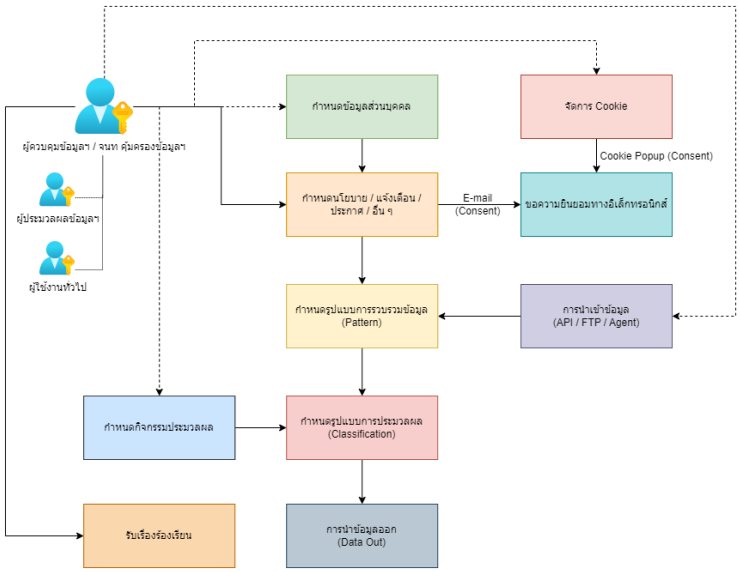
ปัจจัยสู่ความสำเร็จในการปฏิบัติตาม PDPA (Key success factor) องค์กรต่างๆ จะประสบความสำเร็จในการปฏิบัติตามพระราชบัญญัติ คุ้มครองข้อมูลส่วนบุคคล (PDPA) ได้ก็ต่อเมื่อ องค์กรนั้นๆ ประสบความสำเร็จในการบริหารจัดการกับปัจจัยทั้ง 3 ปัจจัย ซึ่งประกอบไปด้วย

- ปัจจัยด้านบุคคล (People) – องค์กรต้องดูแลและควบคุมบุคคลให้ปฏิบัติตาม PDPA โดยอาจทำได้หลายวิธี เช่น การให้ความรู้ การอบรม การให้คำแนะนำ รวมถึงการใช้หนังสือหรือเอกสารทางอิเล็กทรอนิกส์เพื่อขอความยินยอมของบุคคลนั้นๆ

- ปัจจัยด้านกระบวนการจัดการ (Process) – องค์กรต้องตรวจสอบการเก็บรวบรวม ใช้ หรือเปิดเผยข้อมูลส่วนบุคคลในปัจจุบันเพื่อระบุถึงที่มาของข้อมูล ประเภทของข้อมูล จัดกลุ่มข้อมูล ระบุถึงความเสี่ยงของข้อมูล และปรับเปลี่ยนวิธีการเมื่อพบว่าวิธีการนั้นมีความเสี่ยงต่อการละเมิดสิทธิข้อมูลส่วนบุคคล

- ปัจจัยด้านเทคโนโลยี (Technology) - องค์กรต้องใช้เครื่องมือที่ทันสมัยเพื่อมารองรับในการดำเนินงาน ให้การจัดเก็บ ใช้ หรือเปิดเผยข้อมูลส่วนบุคคลเป็นไปตามพระราชบัญญัติ คุ้มครองข้อมูลส่วนบุคคล เช่น ระบบป้องกันข้อมูลรั่วไหล (Data loss prevention) ระบบแจ้งเตือน หรือระบบการเก็บข้อมูล เป็นต้น

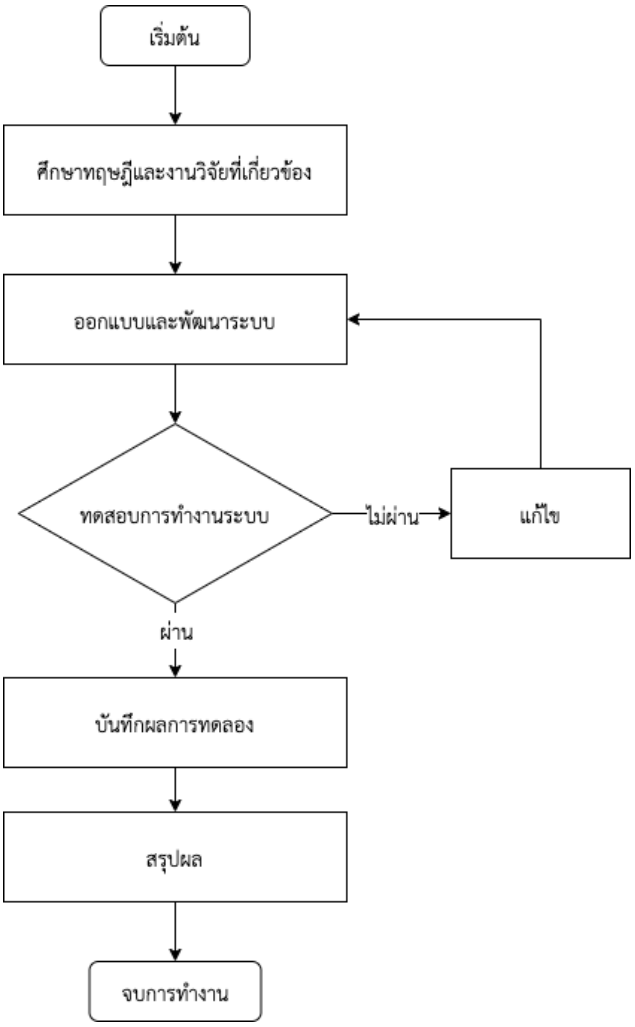
เมื่อพระราชบัญญัติ คุ้มครองข้อมูลส่วนบุคคล (PDPA) เริ่มประกาศใช้ องค์กรหลาย ๆ องค์กรต่างมองหาเทคโนโลยีที่ช่วยให้องค์กรของตนมีเอกสารที่รองรับต่อ PDPA เพื่อไม่ให้ผิดกฎหมาย แต่ในทางปฏิบัติแล้วองค์กรจะต้องเริ่มปรับเปลี่ยนจากกระบวนการจัดการ (Process) ข้อมูลส่วนบุคคลก่อน เพื่อให้กระบวนการนั้นคัดแยกข้อมูล จัดเก็บ ระบุกลุ่ม ทราบแหล่งที่มา ได้อย่างถูกต้อง และสามารถตามข้อมูลได้อย่างแม่นยำเมื่อเกิดปัญหารั่วไหล หลังจากเมื่อกระบวนการจัดการข้อมูลส่วนบุคคลได้ถูกสร้างสำเร็จแล้ว จึงค่อยนำเทคโนโลยีมาช่วยดูแลกระบวนการนั้นให้ทำงานได้ง่ายขึ้น รวดเร็วขึ้น และแม่นยำขึ้น



ภาพที่ 2.7 กระบวนการทำงานของการระบบจัดการข้อมูลให้สอดคล้องกับกฎหมายข้อมูลส่วนบุคคล

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้ได้นำเสนอเกี่ยวกับขั้นตอนและวิธีการดำเนินงานในด้านต่างๆ ทั้งด้านการวางแผนและออกแบบระบบเพื่อสร้างระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล ให้บรรลุจุดประสงค์ของงานวิจัยได้อย่างถูกต้องและสมบูรณ์ จึงได้มีการวางแผนงานการทำงานการออกแบบการดำเนินการสร้างเพื่อให้เสร็จตามระยะเวลาที่กำหนด ซึ่งมีขั้นตอนการทำงานดังภาพที่ 3.1



ภาพที่ 3.1 แผนผังการดำเนินงานวิจัย

จากภาพที่ 3.1 แสดงแผนผังการดำเนินงานของการออกแบบพัฒนาระบบเพื่อสร้างระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูลประกอบด้วยการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง โดย

ประกอบไปด้วยเนื้อหาดังนี้คือ แม่ข่าย, ลูกข่ายประเภทตรวจสอบข้อมูลจราจรคอมพิวเตอร์, ลูกข่ายประเภทส่งไฟล์, ลูกข่ายประเภทฐานข้อมูล และลูกข่ายประเภทข้อมูลจราจร ทดลองการทำงานของระบบและปรับปรุงแก้ไขระบบ โดยผู้วิจัยได้แบ่งขั้นตอนการดำเนินวิจัยได้ดังนั้น

3.1 การออกแบบและพัฒนากระบวนการทำงานของระบบในส่วนของภาพรวม

3.2 การออกแบบและพัฒนาระบบในส่วนของแม่ข่าย

3.3 การออกแบบและพัฒนาระบบในส่วนของลูกข่าย

3.4 การเชื่อมต่อและการทำงานระหว่างแม่ข่ายและลูกข่าย

3.5 การออกแบบและพัฒนาเว็บแอพพลิเคชันเพื่อจัดการแม่ข่ายลูกข่ายและตรวจสอบผลลัพธ์ของลูกข่าย

3.6 การตรวจสอบความครบถ้วนตามข้อตกลงของทางการนำไปใช้งานจริง

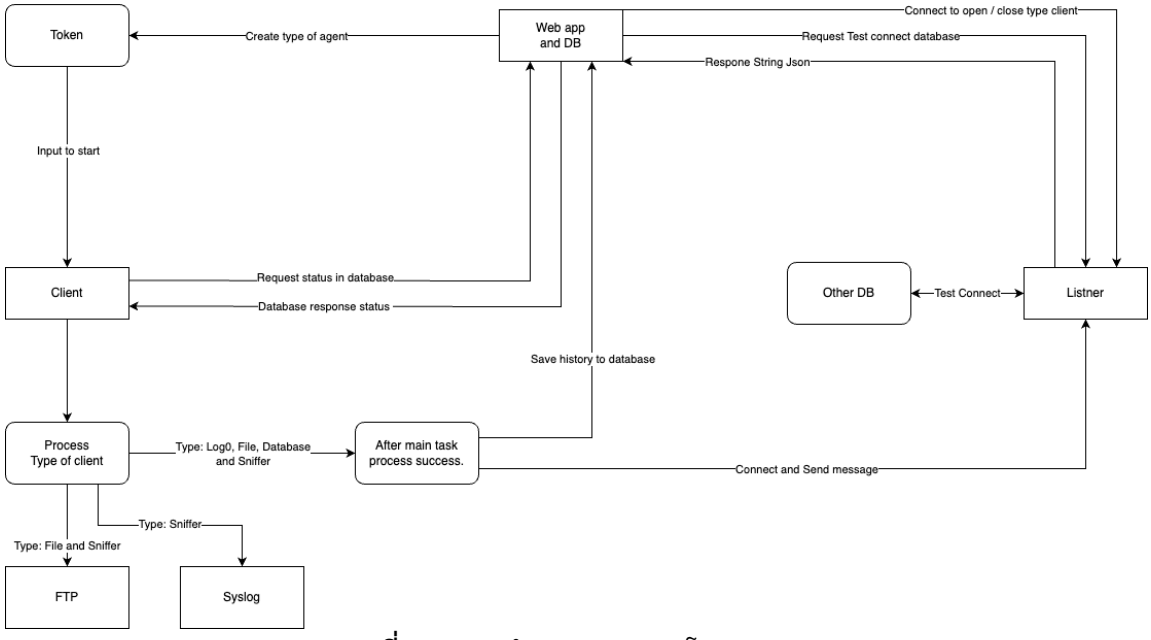
3.7 การตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ภายในเน็ตเวิร์คเดียวกันในแต่ละความเร็ว

3.8 การตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานทุกฟังก์ชันของแต่ละประเภท

3.9 การหาประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องสูงที่สุดในหน่วยของ ซีพียู, แรม, และการอ่านและการเขียนดิสก์

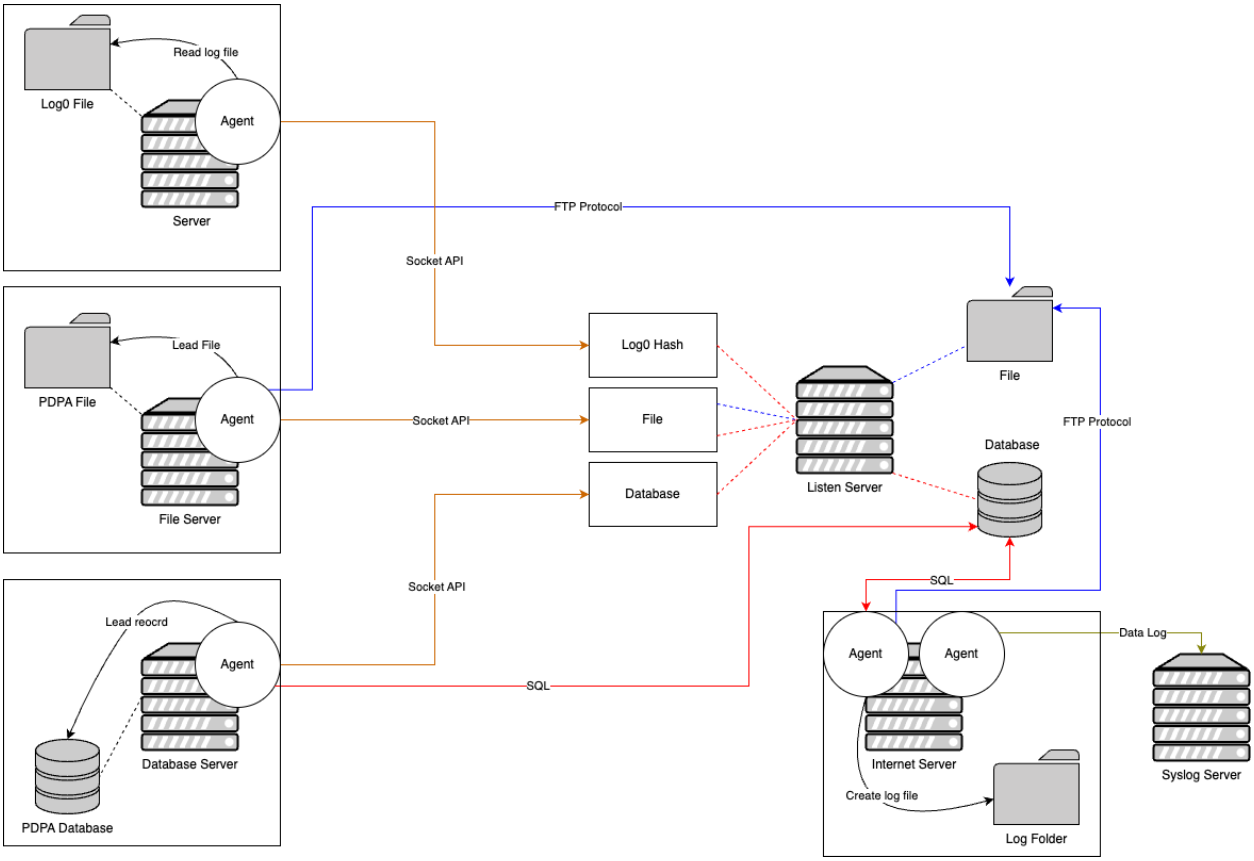
3.10 การทดลองรองรับลูกข่ายที่เชื่อมต่อแม่ข่ายและความเร็วในเสร็จสิ้นการทำงาน ต่อ 1 ครั้ง

3.1 การออกแบบและพัฒนากระบวนการทำงานของระบบในส่วนของภาพรวม



ภาพที่ 3.2 การทำงานของระบบโดยภาพรวม

จากภาพที่ 3.2 การทำงานของระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล ระบบแบ่งการทำงานหลักออกเป็น 2 ส่วนประกอบไปด้วย 1. ลูกข่าย, 2. แม่ข่าย เพิ่มเติมในส่วนของเว็บแอพพลิเคชัน, ฐานข้อมูลหลักของเว็บแอพพลิเคชันหรือจะเป็นฐานข้อมูลอื่นๆ, เซิร์ฟเวอร์ส่งข้อมูลผ่านโปรโตคอลเอฟทีพี, และจราจรเซิร์ฟเวอร์ เพื่อให้ครบองค์ประกอบของกระบวนการทำงานของงานวิจัย



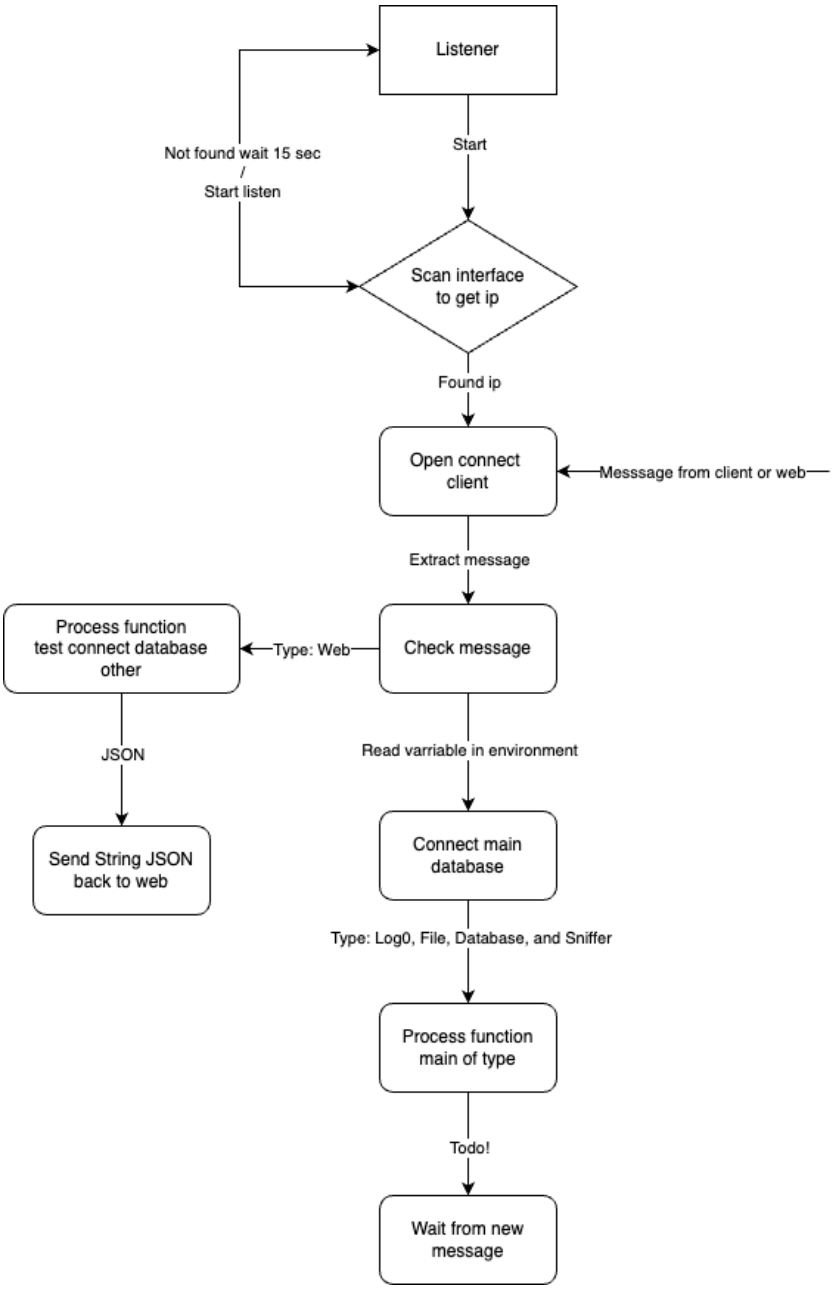
ภาพที่ 3.3 กระบวนการทำงานของระบบในส่วนภาพรวม

จากภาพที่ 3.3 กระบวนการทำงานของระบบจะถูกแบ่งการทำงานออกเป็น 3 รูปแบบหลัก คือ การติดต่อสื่อสารกันผ่าน Socket API, FTP Protocol, และ SQL โดยลูกข่ายประเภทตรวจสอบข้อมูลจราจรคอมพิวเตอร์, ลูกข่ายประเภทส่งไฟล์, และลูกข่ายประเภทฐานข้อมูล จะมีการนำข้อมูลเข้าผ่าน Socket API ต่อมาลูกข่ายประเภทส่งไฟล์ และลูกข่ายประเภทข้อมูลจราจรมี 2 รูปแบบคือ สร้างไฟล์ส่งไฟล์ และ ส่งข้อมูลจราจรไปยังเซิร์ฟเวอร์จราจร โดยรูปแบบแรก มีการนำข้อมูลเข้าผ่าน FTP Protocol ต่อมาลูกข่ายประเภทฐานข้อมูล และลูกข่ายประเภทข้อมูลจราจรสร้างไฟล์ส่งไฟล์ มีการนำข้อมูลเข้าผ่าน SQL และสุดท้ายลูกข่ายประเภทข้อมูลจราจรส่งข้อมูลจราจรไปยังเซิร์ฟเวอร์จราจร มีการส่งข้อมูลผ่านช่องทางบันทึกข้อมูลจราจรคอมพิวเตอร์

3.2 การออกแบบและพัฒนาระบบในส่วนของแม่ข่าย

การออกแบบและพัฒนาระบบในส่วนของแม่ข่ายนั้น จะแบ่งออกเป็นทั้งหมด 4 ส่วน คือ 1. ส่วนการทำงานภาพรวมของแม่ข่าย, 2. ส่วนการทำงานย่อยแม่ข่ายฟังก์ชันข้อมูลจราจร, 3. ส่วนของการทำงานย่อยแม่ข่ายฟังก์ชันไฟล์, 4. ส่วนของการทำงานย่อยแม่ข่ายฟังก์ชันฐานข้อมูล

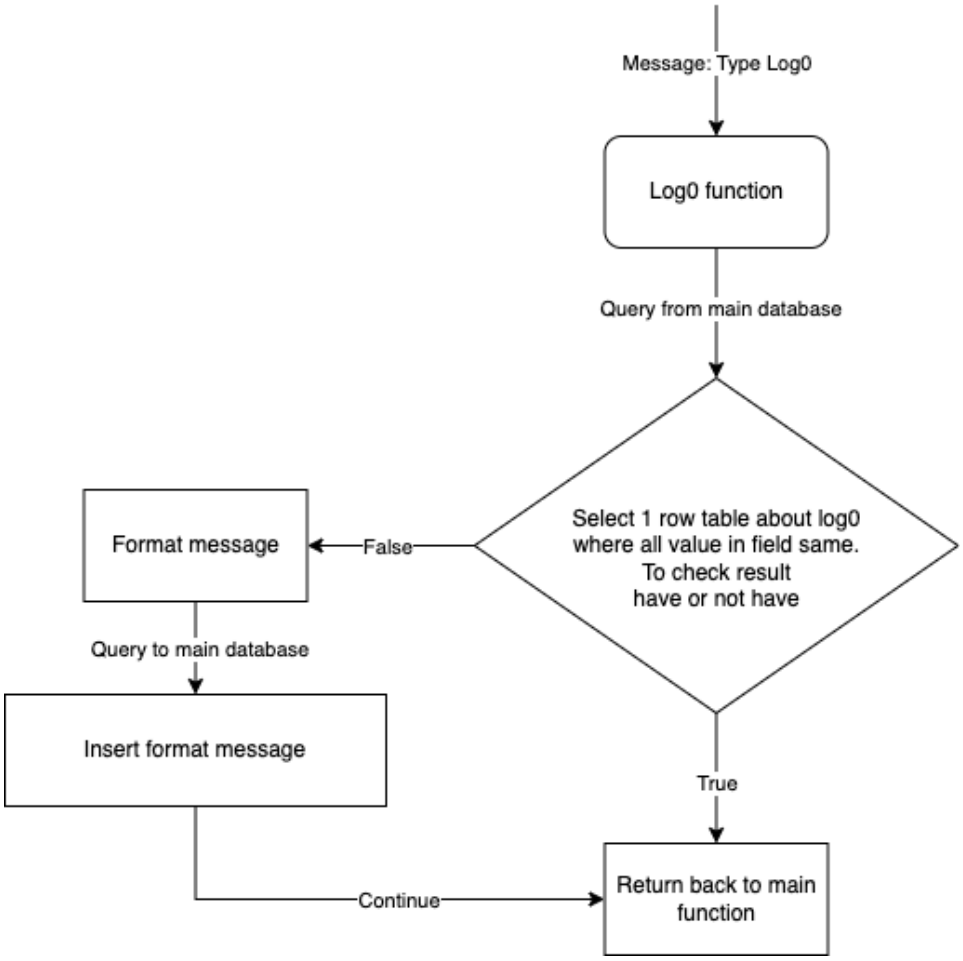
3.2.1 ส่วนการทำงานภาพรวมของแม่ข่าย



ภาพที่ 3.4 การทำงานของระบบภาพรวมโดยแม่ข่าย

จากภาพที่ 3.3 การทำงานของระบบภาพรวมโดยแม่ข่ายจะต้องค้นหาเน็ตเวิร์คอินเตอร์เฟสเพื่อทำการเปิดรับข้อมูลจากลูกข่าย โดยเมื่อค้นหาเจอและใช้งานเน็ตเวิร์คอินเตอร์เฟสนั้นเปิดช่องทางให้เน็ตเวิร์คโซเก็ตทำงานผ่านพอร์ตที่ได้กำหนดไว้ ก็สามารถรับข้อมูลจากลูกข่ายได้ เมื่อมีการรับข้อมูลลูกข่ายทางแม่ข่ายทำการแยกข้อความที่ได้รับจากลูกข่ายเพื่อนำข้อมูลชุดที่แยกนำไปใช้งานต่อ ขั้นตอนแรกนำชุดข้อมูลส่วนหนึ่งนำไปสอบว่าเป็นข้อความประเภทไหนเพื่อผ่านเข้าไปทำงานฟังก์ชันต่อ ถ้าเป็นประเภทพิเศษจากทางเว็บแอพพลิเคชัน จะส่งเข้าไปทำงานในฟังก์ชันทดสอบการเชื่อมต่อฐานข้อมูลที่ถูกกำหนดจากข้อความของประเภทพิเศษ และสอบถามไปยังฐานข้อมูลที่เลือกเพื่อส่งข้อมูลกลับคืนไปให้ทางเว็บแอพพลิเคชัน หรือเป็นประเภทปกติ แม่ข่ายจะเชื่อมต่อกับฐานข้อมูลหลัก จากนั้นจะส่งต่อไปทำงานฟังก์ชันนั้นๆ โดยจะอธิบายในหัวข้อต่อไป และเมื่อเสร็จการทำงานทั้งหมดแม่ข่ายก็จะรอรับข้อมูลลูกข่ายต่อไป

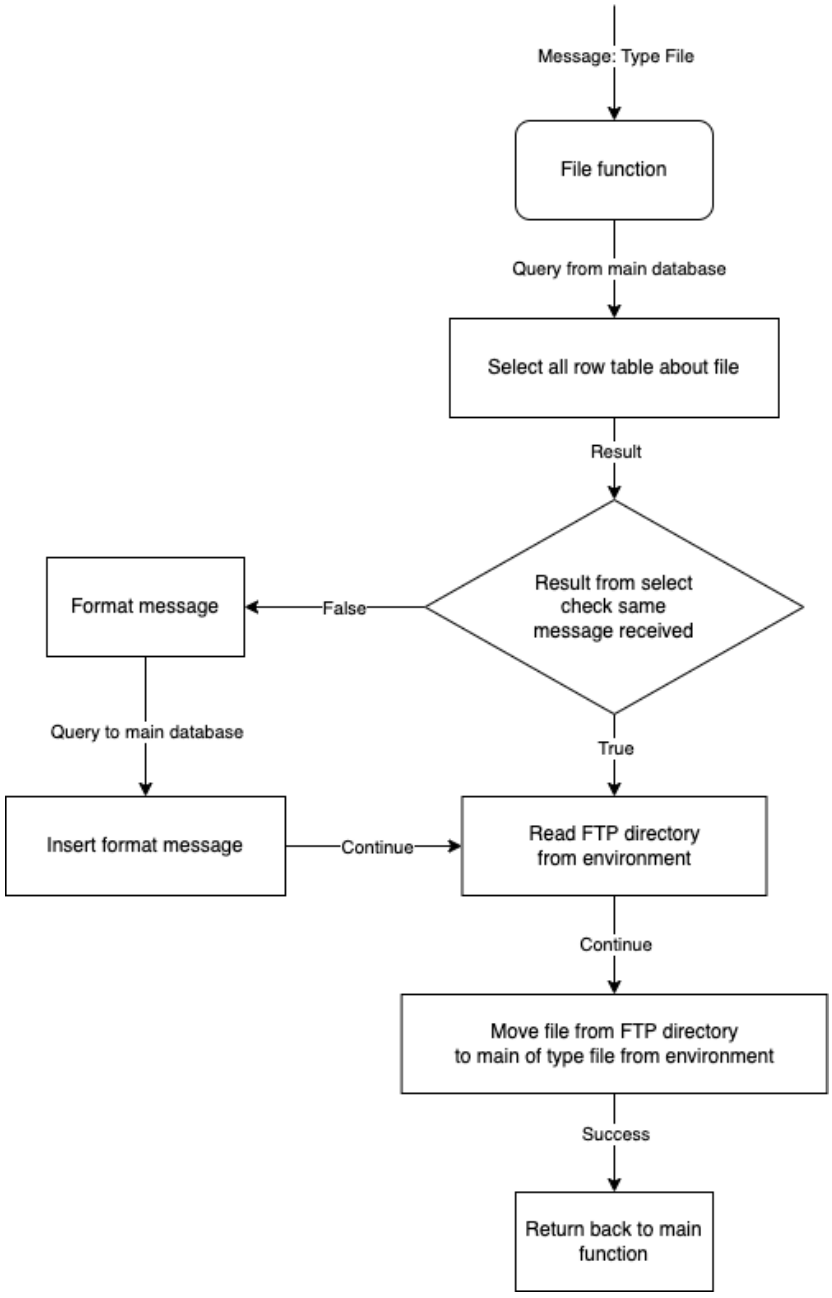
3.2.2 ส่วนการทำงานย่อยแม่ข่ายฟังก์ชันตรวจสอบข้อมูลจราจรคอมพิวเตอร์



ภาพที่ 3.5 การทำงานของระบบแม่ข่ายในส่วนฟังก์ชันตรวจสอบข้อมูลจราจร

จากภาพ 3.4 การทำงานของระบบแม่ข่ายในส่วนฟังก์ชันลอกแม่ข่ายจะทำสอบถามไปยังฐานข้อมูลหลักเพื่อดูผลลัพธ์ว่ามีข้อมูลในฐานข้อมูลที่ตรงกับข้อความที่ได้รับ โดยถ้ามีฟังก์ชันนี้ก็จะถูกปล่อยผ่าน แต่ถ้าไม่นั้นข้อความที่ได้รับจะถูกแปลงให้การเป็นรูปแบบลักษณะให้ตรงกับรูปแบบของฐานข้อมูล จากนั้นจะทำการเพิ่มข้อมูลลงไปเก็บยังฐานข้อมูลหลักก็เป็นอันเสร็จ

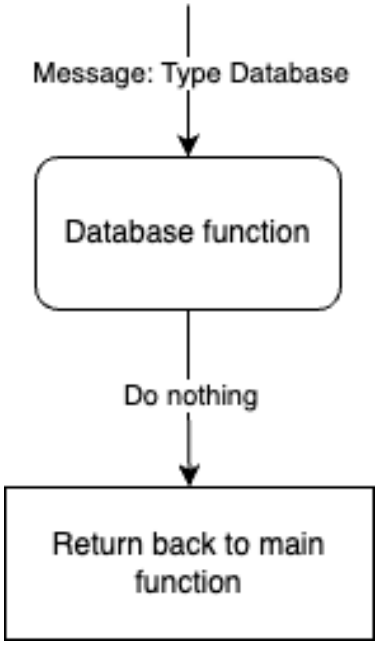
3.2.3 ส่วนของการทำงานย่อยแม่ข่ายฟังก์ชันไฟล์



ภาพที่ 3.6 การทำงานของระบบแม่ข่ายในส่วนส่งไฟล์

ภาพที่ 3.5 การทำงานของระบบแม่ข่ายในส่วนไฟล์ ขั้นแรกจะทำการสอบถามไปยังฐานข้อมูลหลักเพื่อรับข้อมูลทั้งหมดที่เกี่ยวข้องส่วนนี้และนำผลลัพธ์มาสอบว่าเหมือนกับข้อความที่ได้รับมาคล้ายกับการทำงานของแม่ข่ายในส่วนของลอก จากนั้นถ้าไม่มีจะทำการแปลงข้อความให้เป็นรูปแบบลักษณะให้ตรงกับรูปแบบของฐานข้อมูล เมื่อแปลงเสร็จก็ทำการเพิ่มข้อมูลลงไปเก็บไว้ยังฐานข้อมูลหลัก และกลับทำอีกส่วนคือ ไม่ว่าขั้นตอนแรกที่สอบนั้นจะจริงหรือไม่จริงส่วนต่อมานี้ก็จะทำเหมือนกัน คือการรับชื่อไฟล์ทั้งหมดที่อยู่ในโฟลเดอร์ของเอฟทีพี เพื่อดูผ่านไฟล์ที่ชื่อตรงกับข้อความที่แม่ข่ายได้รับมาจากนั้นย้ายไปยังโฟลเดอร์ที่แม่ข่ายได้กำหนดไว้เป็นอันเสร็จ

3.2.4 ส่วนของการทำงานย่อยแม่ข่ายฟังก์ชันฐานข้อมูล



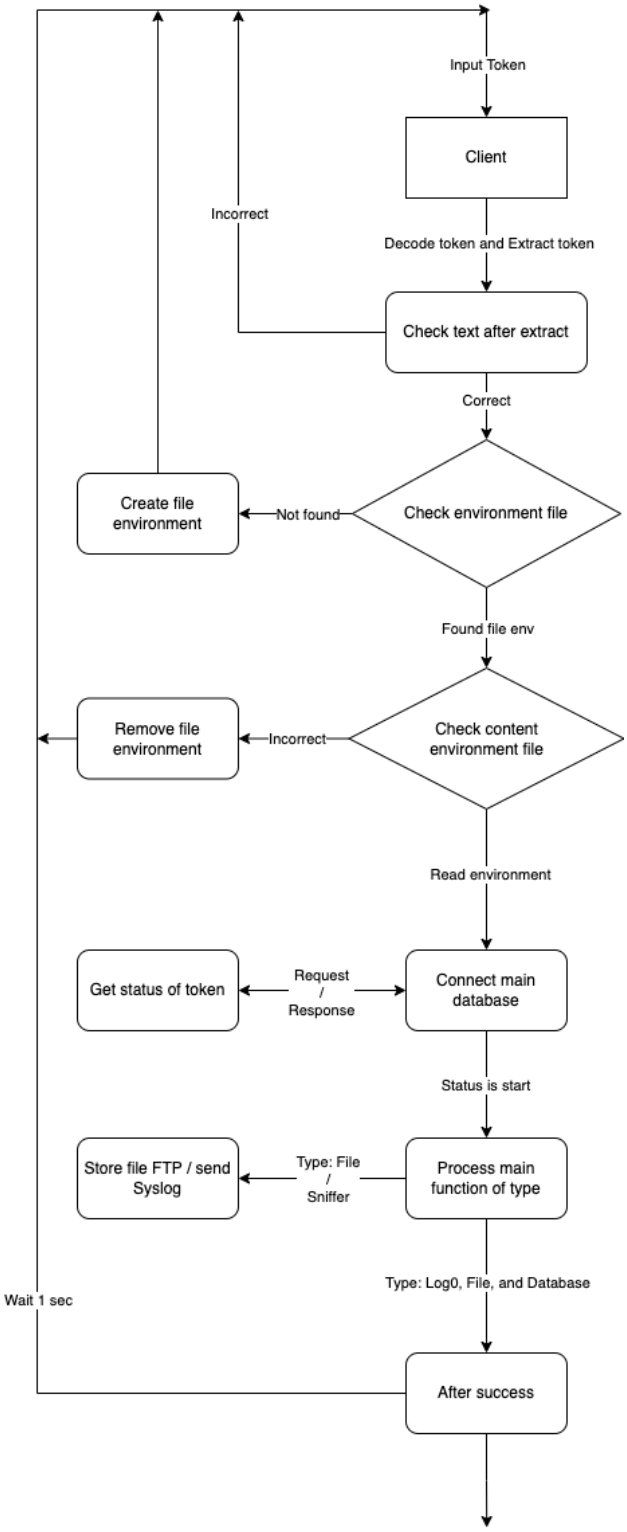
ภาพที่ 3.7 การทำงานของระบบแม่ข่ายในส่วนฐานข้อมูล

จากภาพที่ 3.6 การทำงานของระบบแม่ข่ายในส่วนฐานข้อมูลนั้น ยังไม่มีการทำงานอะไรมากเนื่องจากการทำงานส่วนนี้ไปตกอยู่กับลูกข่ายเป็นส่วนใหญ่

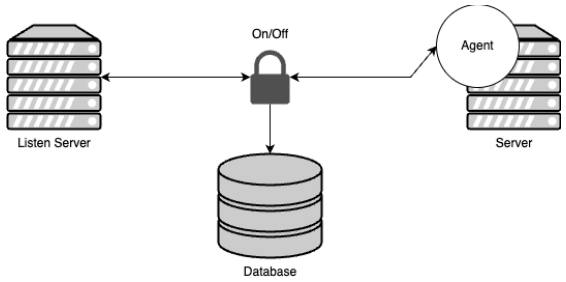
3.3 การออกแบบและพัฒนาระบบในส่วนของลูกข่าย

การออกแบบและพัฒนาระบบในส่วนของลูกข่ายนั้น จะแบ่งออกเป็นทั้งหมด 5 ส่วน คือ 1. ส่วนการทำงานภาพรวมของลูกข่าย, 2. ส่วนการทำงานย่อยลูกข่ายฟังก์ชันตรวจสอบข้อมูลจราจร, 3. ส่วนของการทำงานย่อยลูกข่ายฟังก์ชันส่งไฟล์, 4. ส่วนของการทำงานย่อยลูกข่ายฟังก์ชันฐานข้อมูล, 5. ส่วนของการทำงานย่อยลูกข่ายข้อมูลจราจรคอมพิวเตอร์

3.3.1 ส่วนการทำงานภาพรวมของลูกข่าย



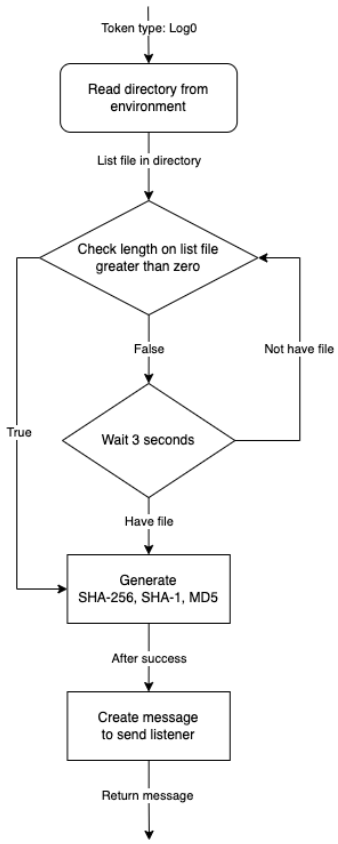
ภาพที่ 3.8 การทำงานของระบบภาพรวมโดยลูกข่าย



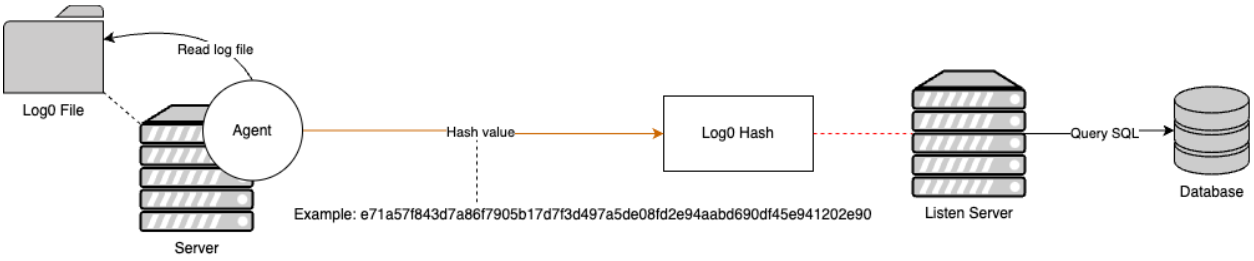
ภาพที่ 3.9 กระบวนการควบคุมการทำงานของระบบ

จากภาพที่ 3.7 การทำงานของระบบภาพรวมโดยลูกข่ายนั้น จะทำงานได้ต่อเมื่อมีการป้อนลักษณะเฉพาะหรืออีกเรียกชื่อว่า “โทเค็น” ระบบจะทำการตรวจสอบข้อมูล 3 อย่างด้วยกันคือ 1. ตรวจสอบความถูกต้องโทเค็น, 2. ตรวจสอบไฟล์ที่เป็นตัวแปรหลัก 3. ตรวจสอบเนื้อหาของไฟล์ที่เป็นตัวแปรหลัก และระบบตรวจสอบทั้ง 3 อย่างผ่าน ระบบทำขั้นตอนถัดไปคือการเชื่อมต่อฐานข้อมูลหลัก เพื่อดูผลลัพธ์ของโทเค็นที่ลูกข่ายได้รับว่า สถานะเป็นอะไร (เปิด/ปิด) และเมื่อสถานะเป็นเปิดส่งค่าไปยัง ดังภาพที่ 3.9 ขั้นตอนถัดไปในแต่ละฟังก์ชันของแต่ละประเภทดังต่อไปนี้

3.3.2 ส่วนการทำงานย่อยลูกข่ายฟังก์ชันตรวจสอบข้อมูลจราจร



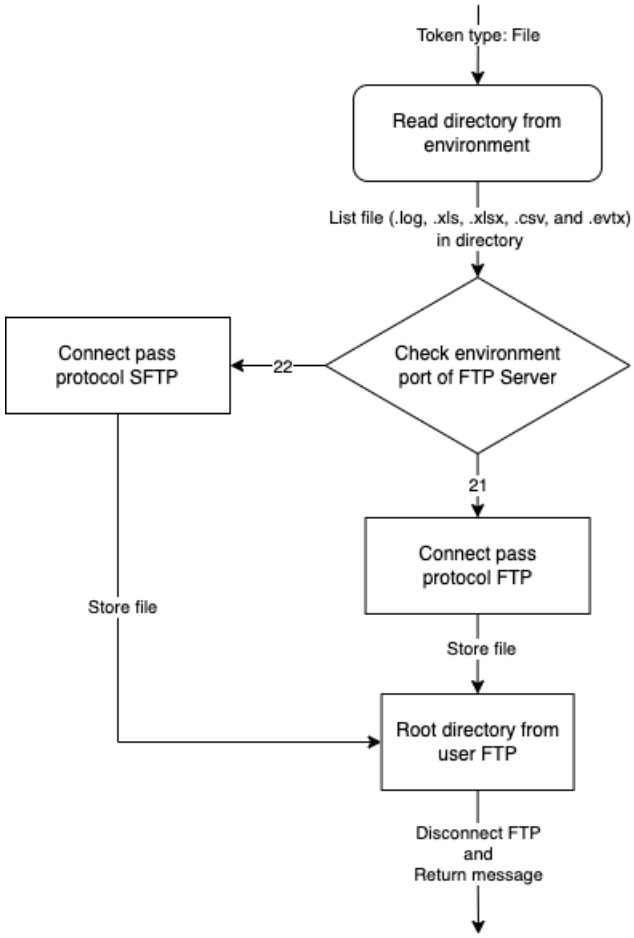
ภาพที่ 3.10 การทำงานของระบบลูกข่ายในส่วนตรวจสอบข้อมูลจราจร



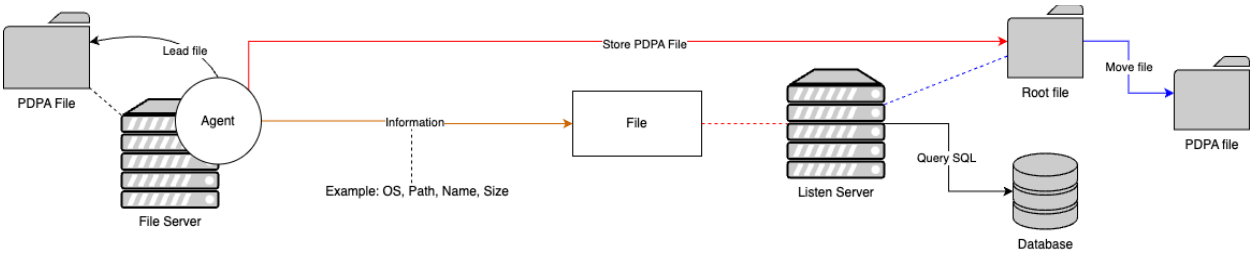
ภาพที่ 3.11 กระบวนการทำงานของตรวจสอบข้อมูลจราจรคอมพิวเตอร์

จากภาพที่ 3.10 และ 3.11 การทำงานของระบบลูกข่ายในส่วนข้อมูลจราจรนั้น เมื่อเข้ามาถึงฟังก์ชันทำการรับชื่อไฟล์ทั้งหมดของโฟลเดอร์ที่เป็นโฟลเดอร์เก็บไฟล์ข้อมูลจราจรที่เลือก จากนั้นตรวจสอบจำนวนไฟล์ทั้งหมดว่ามีหรือไม่ ถ้าไม่ทำการรอ 3 วินาทีก่อนจะเช็คอีกรอบหนึ่ง ถ้ามีทำการสร้างข้อมูลค่าแฮชขึ้นมา และอีกกรณีเมื่อมีไฟล์อยู่แล้ว จะทำการตรวจสอบจำนวนบรรทัดว่ามีค่าเพิ่มขึ้นหรือไม่ ถ้าไม่ก็ปล่อยผ่านไป แต่ถ้าจริงก็จะทำการสร้างข้อมูลค่าแฮชขึ้นมา และทำการจัดรูปแบบข้อความส่งไปให้กับแม่ข่าย

3.3.3 ส่วนของการทำงานย่อยลูกข่ายฟังก์ชันส่งไฟล์



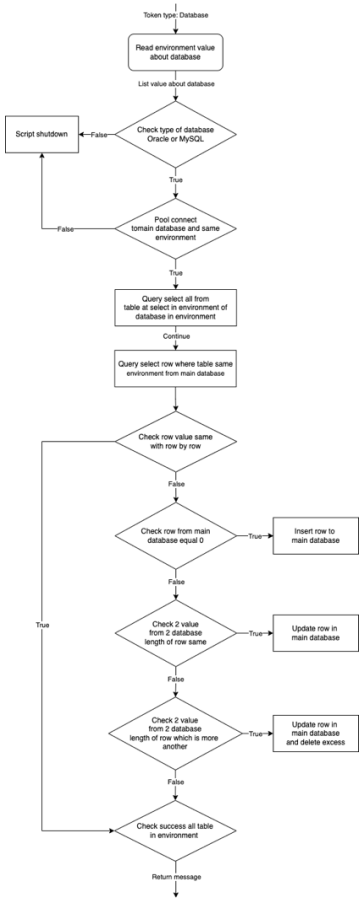
ภาพที่ 3.12 การทำงานของระบบลูกข่ายในส่วนส่งไฟล์



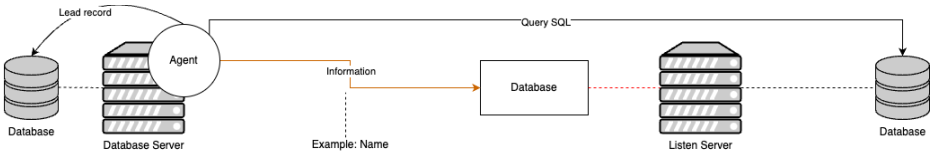
ภาพที่ 3.13 กระบวนการทำงานของระบบลูกข่ายในส่วนส่งไฟล์

จากภาพที่ 3.12 และ 3.13 การทำงานของระบบลูกข่ายในส่วนไฟล์นั้น เมื่อเข้ามาถึงฟังก์ชันทำการรับชื่อไฟล์ทั้งหมดของโฟลเดอร์ที่เป็นโฟลเดอร์เก็บไฟล์ข้อมูลที่เลือก เช่น ไฟล์จราจร, ไฟล์เอ็กเซล, ไฟล์ชุดข้อมูล (ซีเอสวี) จากนั้นตรวจสอบค่าตัวแปรที่เกี่ยวข้องกับโปรโตคอลเอฟทีพีเซิร์ฟเวอร์ โดยจะมี 2 เงื่อนไขคือ ถ้าพอร์ตของค่าตัวแปรที่เกี่ยวข้องเป็น 22 นั้นจะย้ายไปทำงานผ่านโปรโตคอลเอสเอฟทีพี จากนั้นไม่ว่าจะทำงานผ่านโปรโตคอลเอฟทีพี หรือเอสเอฟทีพีจะทำการบันทึกไฟล์ลงไปยังโฟลเดอร์หลักเซิร์ฟเวอร์ของโปรโตคอลนั้นๆ เมื่อบันทึกไฟล์ได้ทั้งหมดเสร็จจะทำการออกจากการเชื่อมต่อกับเซิร์ฟเวอร์ของโปรโตคอลนั้นๆ และส่งข้อความไปหายังแม่ข่าย

3.3.4 การทำงานของระบบลูกข่ายในส่วนฐานข้อมูล

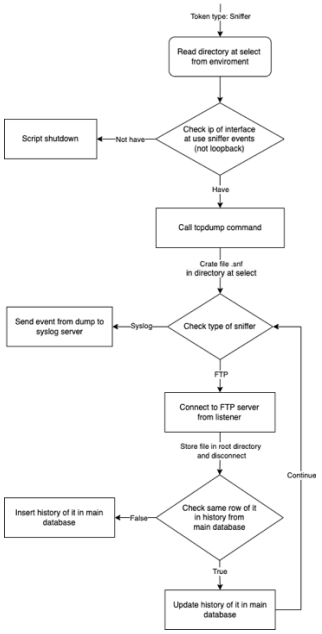


ภาพที่ 3.14 การทำงานของระบบลูกข่ายในส่วนฐานข้อมูล



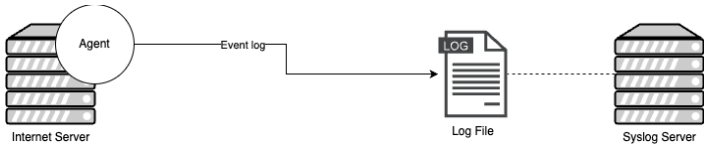
ภาพที่ 3.15 กระบวนการทำงานของระบบลูกข่ายในส่วนของฐานข้อมูล

จากภาพที่ 3.10 การทำงานของระบบลูกข่ายในส่วนฐานข้อมูล จะทำการตรวจสอบประเภทของฐานข้อมูลที่ลูกข่ายถูกกำหนดมาว่าเป็นอะไรโดยผู้วิจัยได้ทำรองรับไว้อยู่ 2 ประเภทคือ มายเอสคิวเอล (MySQL) และ ออราเคิ้ล (Oracle) เมื่อเป็น 2 ประเภทที่ระบบรองรับไว้ระบบจะทำการเชื่อมต่อฐานข้อมูลในรูปแบบพลู (Pool) เพื่อป้องกันไม่ให้มีการเชื่อมต่อจำนวนมากๆ แล้วทำการดึงข้อมูลฝั่งที่ลูกข่ายถูกกำหนดและดึงข้อมูลฝั่งของฐานข้อมูลหลัก จากนั้นทำการตรวจสอบว่ามีข้อมูลนั้นทั้งสองฝั่งมีเหมือนกันหรือไม่ ถ้าไม่ทำการเพิ่มลงไปยังฝั่งฐานข้อมูลหลัก ถ้ามีตรวจสอบว่ามีการเปลี่ยนแปลงหรือไม่ ถ้าจริงลูกข่ายจะทำการปรับปรุงข้อมูลนั้นฝั่งฐานข้อมูลหลัก แต่ถ้าไม่ก็จะตรวจสอบรอบสุดท้ายว่ามีข้อมูลเกินขึ้นมาจากฝั่งฐานข้อมูลหรือไม่ ถ้าจริงก็ทำการลบข้อมูลที่เกินมาจากนั้นก็ส่งข้อความไปยังแม่ข่าย กรณีฐานข้อมูลไม่เป็น 2 ประเภทที่ระบบรองรับไว้ระบบจะหยุดทำงานลง



ภาพที่ 3.16 การทำงานของระบบลูกข่ายในส่วนข้อมูลจราจรคอมพิวเตอร์ทั้ง 2 รูปแบบ

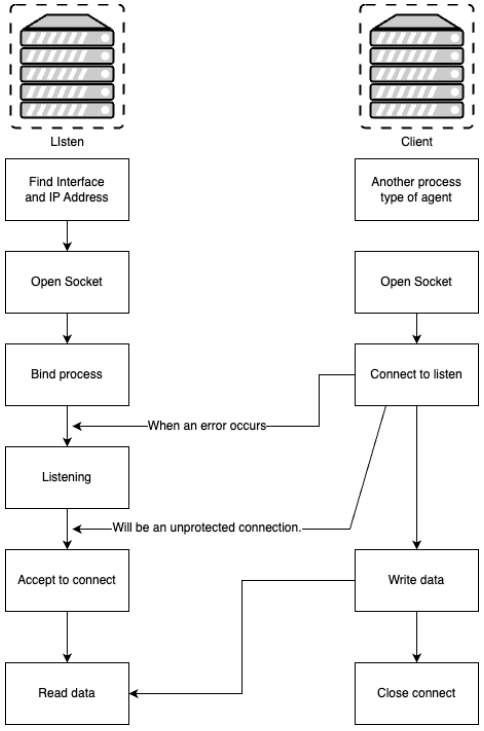
ภาพที่ 3.17 กระบวนการทำงานของระบบลูกข่ายในส่วนข้อมูลจราจรคอมพิวเตอร์รูปแบบสร้างไฟล์ส่งไฟล์



ภาพที่ 3.18 กระบวนการทำงานของระบบลูกข่ายในส่วนข้อมูลจราจรคอมพิวเตอร์รูปแบบส่งข้อมูลจราจรคอมพิวเตอร์

จากภาพที่ 3.16, 3.17, และ 3.18 การทำงานระบบลูกข่ายในส่วนตัววิเคราะห์โปรโตคอล โดยระบบนี้จะรองรับเฉพาะระบบปฏิบัติการยูนิกส์เท่านั้น ทำการค้นหาอินเตอร์เฟสเน็ตเวิร์คที่สามารถใช้งานอินเตอร์เน็ตได้ เมื่อเจอจะทำการเรียกใช้คำสั่งของระบบปฏิบัติการยูนิกส์คำสั่งการถ่ายโอนข้อมูลโปรโตคอลทีซีพี เพื่อเขียนไฟล์ที่เป็นไฟล์เฉพาะของระบบขึ้นมาโดยนำข้อมูลที่ได้รับจากคำสั่งการถ่ายโอนข้อมูลโปรโตคอลทีซีพี โดยจะเป็นถูกกำหนดการเขียนระยะเวลาหนึ่งก็จะสร้างไฟล์ขึ้นมาใหม่ จากนั้นส่งไฟล์ ไปยังเซิร์ฟเวอร์ผ่านโปรโตคอลเอฟทีพี ขั้นตอนสุดท้ายทำการรับข้อมูลประวัติการทำงานของลูกข่ายต่างๆในฐานข้อมูลหลักทำการตรวจสอบว่ามีข้อมูลที่เหมือนกันกับทางระบบลูกข่ายนี้หรือไม่ ถ้าจริงก็จะทำการอัพเดทข้อมูลนั้น แต่ถ้าไม่จริงก็จะทำการเพิ่มข้อมูลใหม่ลงไป

3.4 การเชื่อมต่อและการทำงานระหว่างแม่ข่ายและลูกข่าย

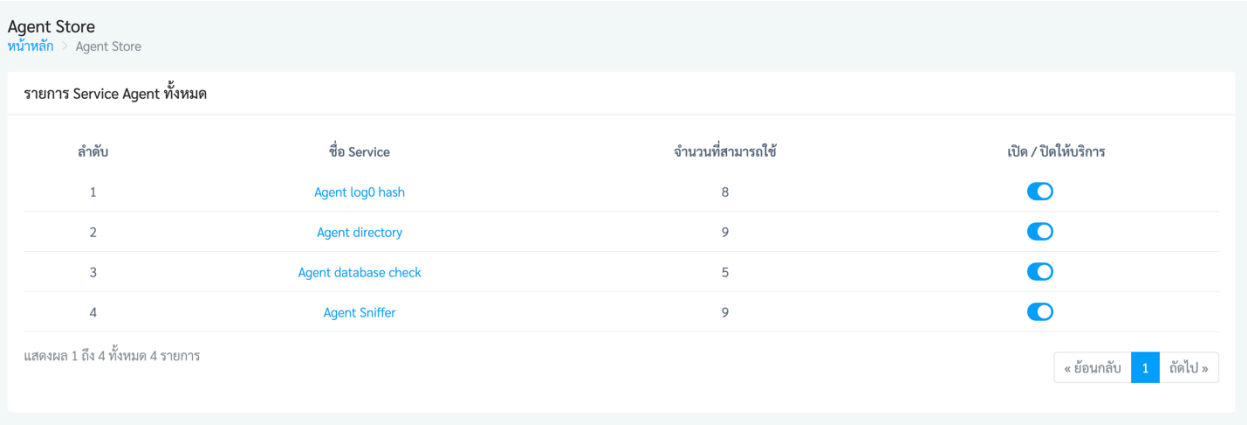


ภาพที่ 3.19 การเชื่อมต่อและการทำงานระหว่างแม่ข่ายและลูกข่าย

จากภาพที่ 3.19 การเชื่อมต่อและการทำงานระหว่างแม่ข่ายและลูกข่ายนั้นขั้นแรกจำเป็นต้องมีการเชื่อมต่อเข้ากับเน็ตเวิร์คในองค์กรนั้นๆ โดยแม่ข่ายจะค้นหาค่าของหมายเลขของเน็ตเวิร์ค (IP Address) โดยจะเลือกจากอันดับแรกที่ค้นหาเจอ เมื่อค้นหาเจอแม่ข่ายจะทำการเปิดรับข้อมูลผ่านโปรโตคอลโซเก็ต (Socket) ตามหมายเลขช่องทางคอมพิวเตอร์ (Port) ที่ได้กำหนดไว้ตามตั้งค่า ต่อมาแม่ข่ายจะเตรียมการรับข้อมูลจากลูกข่าย เมื่อไม่เกิดปัญหา แม่ข่ายพร้อมรับข้อมูลจากลูกข่าย แต่เมื่อเจอปัญหาจะแสดงความที่เกิดการผิดพลาดขึ้น เมื่อลูกข่ายได้ทำงานตามกระบวนการอะไรของแต่ละประเภทเสร็จสิ้น จะทำการเชื่อมต่อไปยังแม่ข่ายโดยผ่านโปรโตคอลโซเก็ต (Socket) ผ่านหมายเลขของเน็ตเวิร์ค (IP Address) และหมายเลขช่องทางคอมพิวเตอร์ (Port) ของแม่ข่ายที่อยู่ในการตั้งค่าของลูกข่าย และเมื่อส่งให้แม่ข่าย แม่ข่ายจะตรวจสอบว่าอนุญาติหรือไม่ เมื่ออนุญาติ ลูกข่ายจะเขียนข้อมูลส่งให้กับแม่ข่ายจนครบถ้วนตัวลูกข่ายจะทำการตัดการเชื่อมต่อเองทันที เพื่อไปทำงานตามกระบวนการเดิมต่างๆ และเมื่อเสร็จอีกครั้งก็จะเชื่อมต่อใหม่เพื่อส่งให้กับแม่ข่ายใหม่อีกรอบแบบนี้ไปเรื่อย ๆ เพิ่มเติมในส่วนของแม่ข่ายจะมีการทำงานแบบเป็นแบบต่อเนื่องกัน (Thread) เพื่อให้สามารถรองรับลูกข่ายได้จำนวนมาก ๆ พร้อมกัน

3.5 การออกแบบและพัฒนาเว็บแอพพลิเคชันเพื่อจัดการแม่ข่ายลูกข่ายและตรวจสอบผลลัพธ์ของลูกข่าย

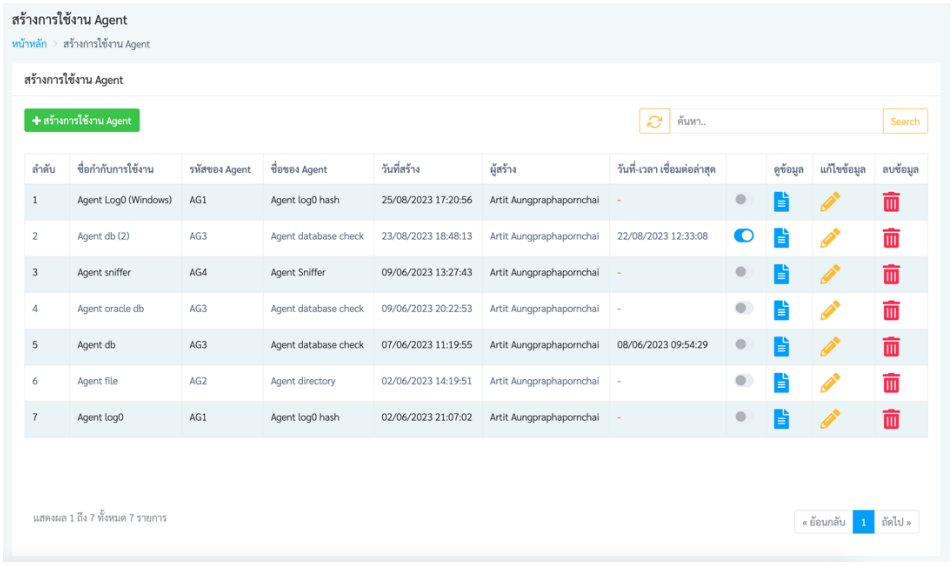
เนื่องจากการจัดการและการตรวจสอบผลลัพธ์ของลูกข่ายนั้นเป็นไปได้ยาว เมื่อต้องใช้งานผ่านการพิมพ์โดยรูปแบบการแสดงผลแบบโหมดข้อความ (Text mode) หรือ CLI (Command-line user interface) ทางผู้วิจัยจึงได้ออกแบบและพัฒนาเว็บไซต์ขึ้นมาเพื่อช่วยลดปัญหาการต้องแก้ไข หรือสร้างลูกข่ายขึ้นมาใช้งาน โดยจะมีหน้าต่างจัดการดังภาพที่ 3.20 – 3.27



ภาพที่ 3.20 ส่วนของแม่ข่ายในการรองรับและเปิดปิดรับลูกข่ายแต่ละประเภท

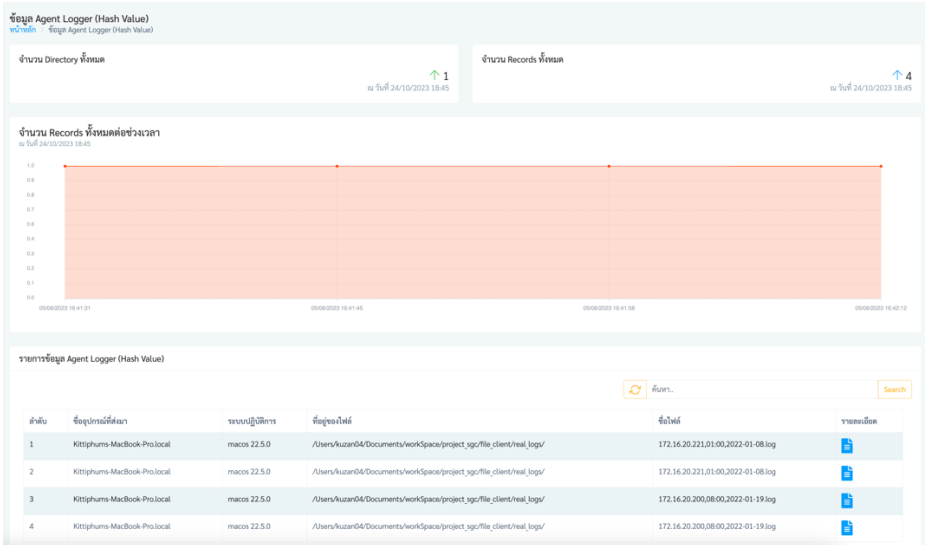
จากภาพที่ 3.20 ส่วนของแม่ข่ายในการรองรับและเปิดปิดรับลูกข่ายแต่ละประเภท เนื่องด้วยการสร้างลูกข่าย ผู้ให้บริการแม่ข่ายสามารถมีกำหนดการสร้างลูกข่ายแต่ละประเภทได้ตามต้องการ อีกส่วนหนึ่งคือการกำหนด

ว่าลูกข่ายแต่ละประเภทนั้นแม่ข่ายจะเปิดรับข้อมูลหรือไม่



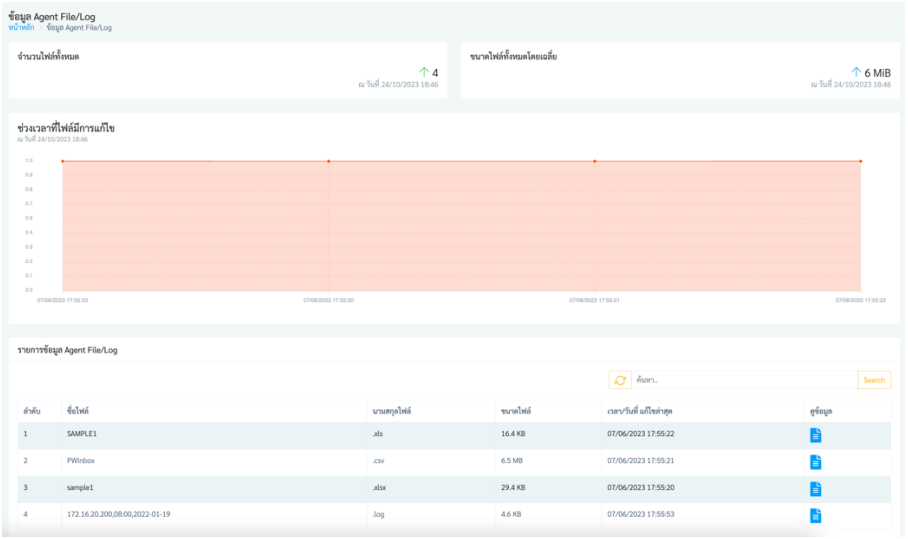
ภาพที่ 3.21 ส่วนของการจัดการลูกข่ายที่ถูกสร้างขึ้น

จากภาพที่ 3.21 ส่วนของการจัดการลูกข่ายที่ถูกสร้างขั้น การสร้าง, การแก้ไข, หรือการลบ จะอยู่ในส่วนนี้เป็นเหมือนกับส่วนรวมของลูกข่ายทั้งหมดทุกประเภทที่ถูกสร้างขึ้น เพื่อนำไปใช้งาน โดยสามารถดูสถานะการทำงานเปิดปิดได้ผ่านทางรูปที่เหมือนกับสวิตช์ดังภาพด้านบน รวมถึงมีการบอกสถานะเวลาที่ทำงานล่าสุดเมื่อไหร่ ยังรวมถึงผู้ใช้งานในเว็บแอพพลิเคชันนั้นใครเป็นผู้สร้างลูกข่ายนั้นขึ้นมา สร้างขึ้นมาเมื่อไหร่ ชื่อที่ตั้งให้กับลูกข่ายนั้น ๆ และรหัสของประเภทลูกข่าย



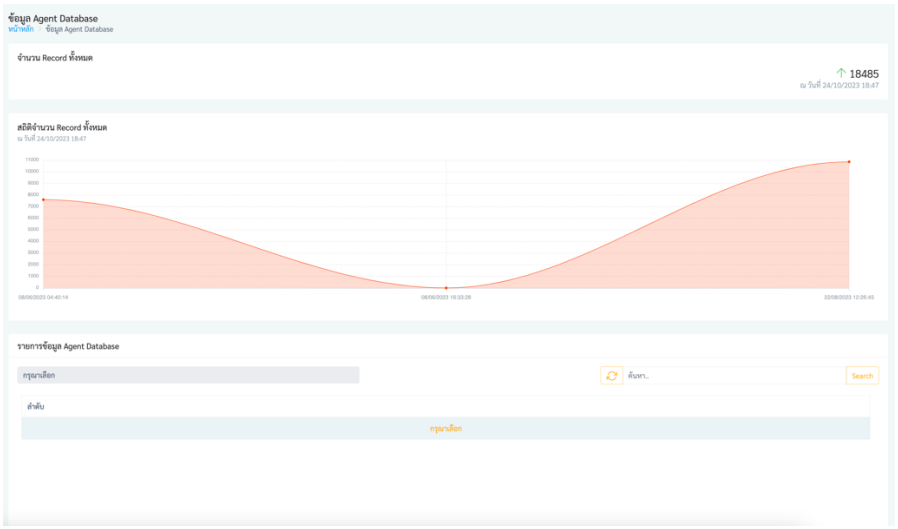
ภาพที่ 3.22 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทตรวจสอบข้อมูลจราจรคอมพิวเตอร์

จากภาพที่ 3.22 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทตรวจสอบข้อมูลจราจรคอมพิวเตอร์ จะเป็นการแสดงข้อมูลเพื่อนำข้อมูลไปตรวจสอบความถูกต้องของไฟล์ประเภทตรวจสอบข้อมูลจราจรคอมพิวเตอร์ที่ลูกข่ายได้ส่งมายังแม่ข่าย และบันทึกลงฐานข้อมูลหลัก โดยแสดงชื่อเครื่องที่ได้นำลูกข่ายไปติดตั้ง, ระบบปฏิบัติการอะไร, ที่อยู่ของไฟล์ สุดท้ายชื่อของไฟล์ข้อมูลจราจรข้อมูลนั้นที่ลูกข่ายได้ตรวจสอบ



ภาพที่ 3.23 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทไฟล์

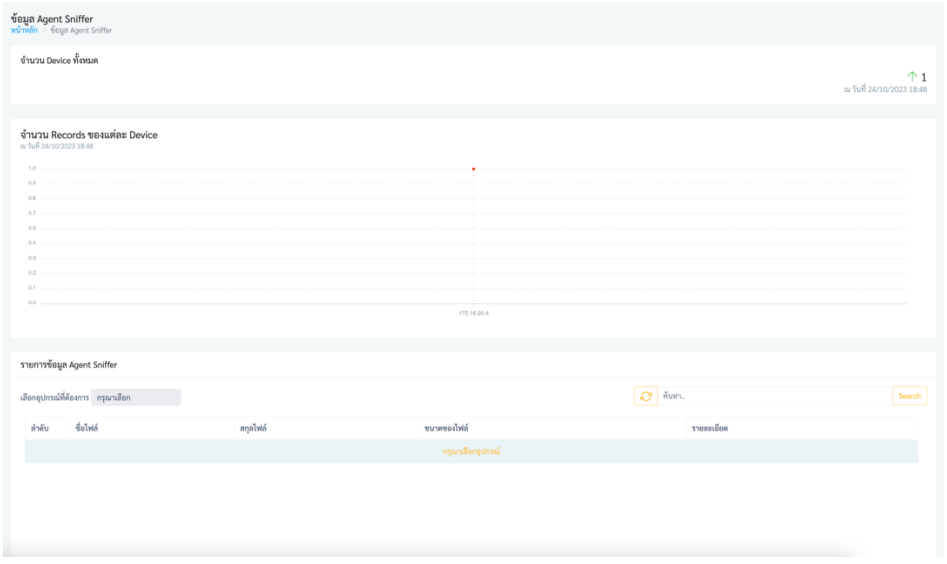
จากภาพที่ 3.23 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทไฟล์ จะเป็นการแสดงข้อมูลเพื่อบ่งบอกถึงไฟล์เอกสารหรือไฟล์ต่างๆ ที่ลูกข่ายประเภทไฟล์ ได้ส่งมาเก็บไว้ยังเซิร์ฟเวอร์กลาง โดยแสดงชื่อไฟล์, นามสกุลไฟล์, ขนาดของไฟล์, สุดท้ายจะเป็นเวลาที่ได้ทำการส่งไฟล์จากลูกข่ายมายังแม่ข่ายหรือเครื่องเซิร์ฟเวอร์หลัก



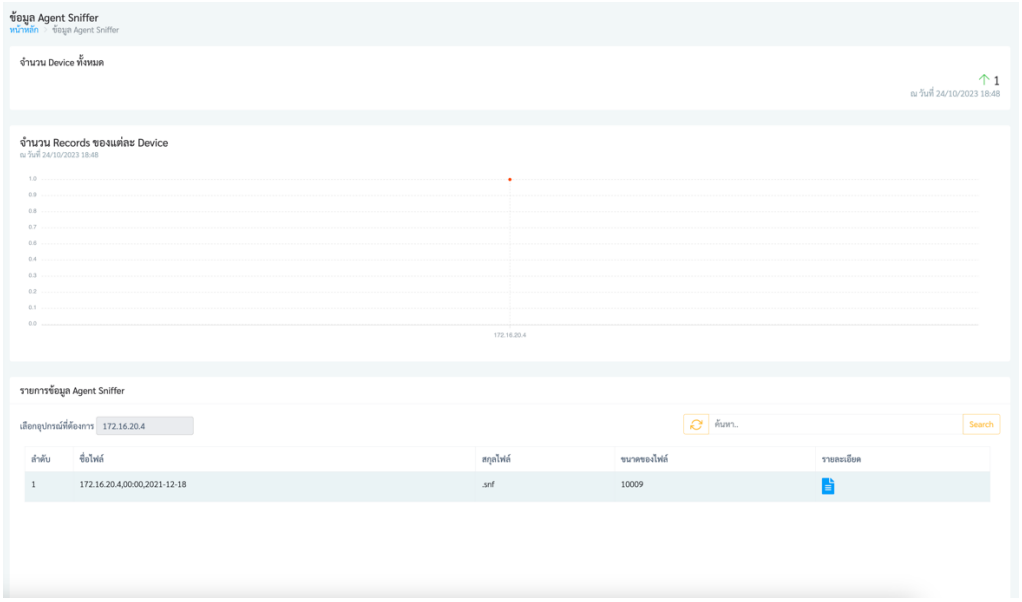
ภาพที่ 3.24 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทฐานข้อมูลส่วนแรก

ภาพที่ 3.25 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทฐานข้อมูลส่วนที่ 2

จากภาพที่ 3.24 และ 3.25 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทฐานข้อมูล จะเป็นการแสดงข้อมูลเพื่อสามารถนำไปตรวจสอบหรืออ้างอิงกับฐานข้อมูลที่ลูกข่ายประเภทฐานข้อมูลได้ติดตั้งอยู่ โดยส่วนแรก ภาพที่ 3.17 เมื่อเข้าไปยังส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทฐานข้อมูลครั้งแรก จะแสดงข้อมูลแบบเปล่า ๆ เนื่องจากผู้ใช้งานของเว็บแอพพลิเคชัน จำเป็นต้องเลือกดูฐานข้อมูลที่ต้องการเสียก่อน ดังภาพที่ 3.18 จะแสดงรายการทั้งหมดที่ลูกข่ายฐานข้อมูลได้ส่งมายังฐานข้อมูลหลัก โดยข้อมูลตารางหรือข้อมูลต่างๆ จะเป็นข้อมูลจากทางฝั่งลูกข่ายประเภทฐานข้อมูลส่งมาให้



ภาพที่ 3.26 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทตัววิเคราะห์โปรโตคอลรูปแบบสร้างไฟล์และส่งไฟล์ส่วนแรก



ภาพที่ 3.27 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทข้อมูลจราจรรูปแบบสร้างไฟล์และส่งไฟล์ส่วนที่ 2

จากภาพที่ 3.26 และ 3.27 ส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทข้อมูลจราจร จะเป็นการแสดงข้อมูลเพื่อสามารถนำวิเคราะห์ของการใช้งานเน็ตเวิร์คที่ลูกข่ายประเภทข้อมูลจราจร ได้ติดตั้งอยู่ โดยส่วนแรกภาพที่ 3.19 จะเหมือนกับส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทฐานข้อมูล เมื่อเข้าไปยังส่วนของการแสดงผลลัพธ์ของลูกข่ายประเภทข้อมูลจราจรครั้งแรก จะแสดงข้อมูลแบบเปล่า เนื่องจากผู้ใช้งานเว็บแอพพลิเคชัน จำเป็นจะต้องเลือกดูหมายเลข IP (IP Address) ของเครื่องที่ลูกข่ายประเภทข้อมูลจราจรได้ติดตั้งที่ต้องเสียก่อน ดังภาพที่ 3.20 จะแสดงรายการไฟล์ที่ลูกข่ายประเภทข้อมูลจราจรได้ส่งมาให้กับเซิร์ฟเวอร์หลัก โดยจะแสดงชื่อไฟล์, นามสกุลไฟล์, สุดท้ายขนาดของไฟล์

3.6 การตรวจสอบความครบถ้วนตามข้อตกลงของการนำไปใช้งานจริง

จากหัวข้อนี้ผู้วิจัยได้นำระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูลไปติดตั้งใช้งานจริงให้กับทางหน่วยงานหน่วยงานหนึ่งและทางหน่วยงานได้มีการกำหนดครอบเขตและผลลัพธ์ออกมาให้ครอบคลุมกับการใช้งานของหน่วยงานหน่วยงานนั้นได้ข้อสรุปดังนี้

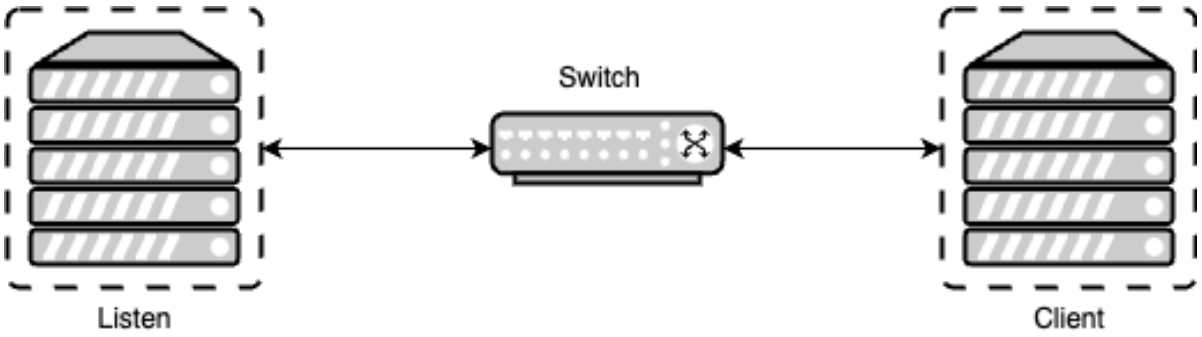
3.5.1 โปรแกรมที่ลงในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่าย เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้

3.5.2 อุปกรณ์ตัวแทนที่สามารถจำต้องได้ (Hardware Agent) นำติดตั้งเพื่อดัก (Tap) หรือเชื่อมต่อกับอุปกรณ์ สวิตช์เน็ตเวิร์ค (Switch) แบบสะท้อน (Mirror) เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการใช้เครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้

3.5.3 โปรแกรมที่อยู่ในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อเชื่อมต่อกับฐานข้อมูล (Database) ของโปรแกรมในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อทำการตรวจสอบข้อมูลส่วนบุคคลในฐานข้อมูล (Database) นั้น

3.7 การตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ภายในเน็ตเวิร์คเดียวกันในแต่ละความเร็ว

ขั้นตอนการตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์เดียวกันในแต่ละความเร็วของสายสัญญา ผู้วิจัยได้นำอุปกรณ์เครือข่ายไม่มีการกำหนดค่าไว้มาทดลอง โดยสภาพแวดล้อมของระบบที่จัดการกำหนดค่าอุปกรณ์เครือข่ายเป็นดังภาพที่ 3.28

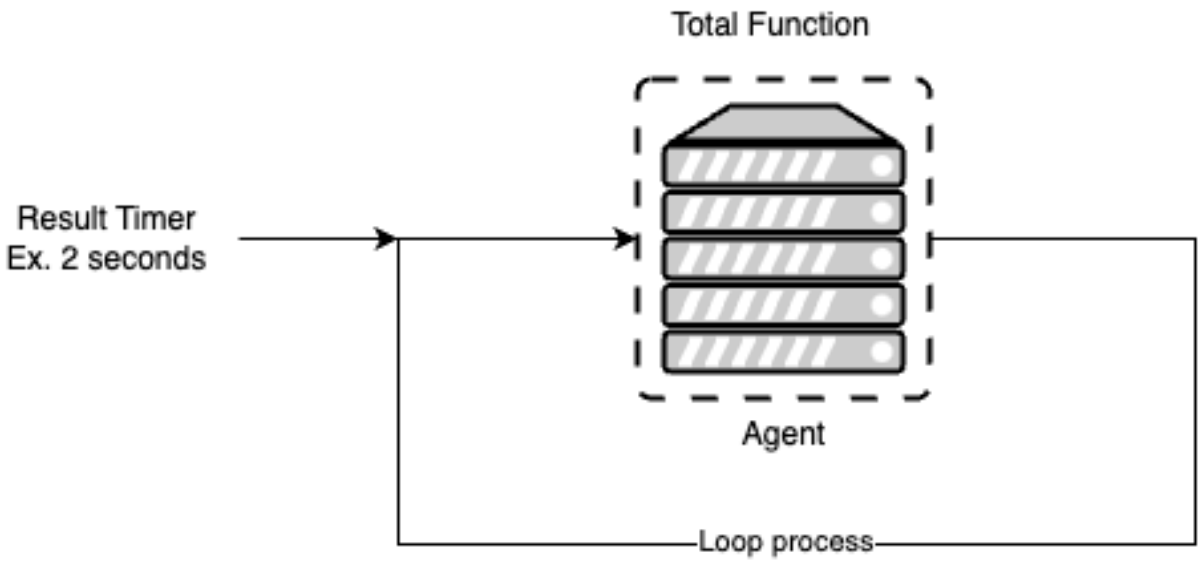


ภาพที่ 3.28 การเชื่อมต่อระหว่างระบบและอุปกรณ์เน็ตเวิร์คสวิตช์

จากโครงสร้างสภาพแวดล้อมของระบบจะเห็นได้ว่าทางระบบแม่ข่ายและลูกข่ายจะต้องส่งข้อมูลผ่านทางสายสัญญาที่เชื่อมต่อกับทางอุปกรณ์เน็ตเวิร์คสวิตช์โดยทางผู้วิจัยจะกำหนดให้กับทางพอร์ตการเชื่อมต่อของทั้ง 2 สายสัญญาระหว่าง 100Mbps และ 1000Mbps โดยจะดูข้อมูลของการรับส่งของจำนวนของค่าบัพเพอร์ (Buffer) ว่าจำนวนส่งและรับเท่านั้นหรือไม่

3.8 การตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานทุกฟังก์ชันของแต่ละประเภท

ในขั้นตอนนี้เป็นขั้นตอนสำหรับการตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานของแต่ละฟังก์ชันในแต่ละส่วนของระบบแม่ข่ายและลูกข่ายนั้นๆ ว่ามีจำนวนเท่าไหร่ และใช้งานเวลาในการประมวลผลจนเสร็จ แล้วจะเริ่มใหม่อีกครั้งหนึ่งดังภาพที่ 3.29



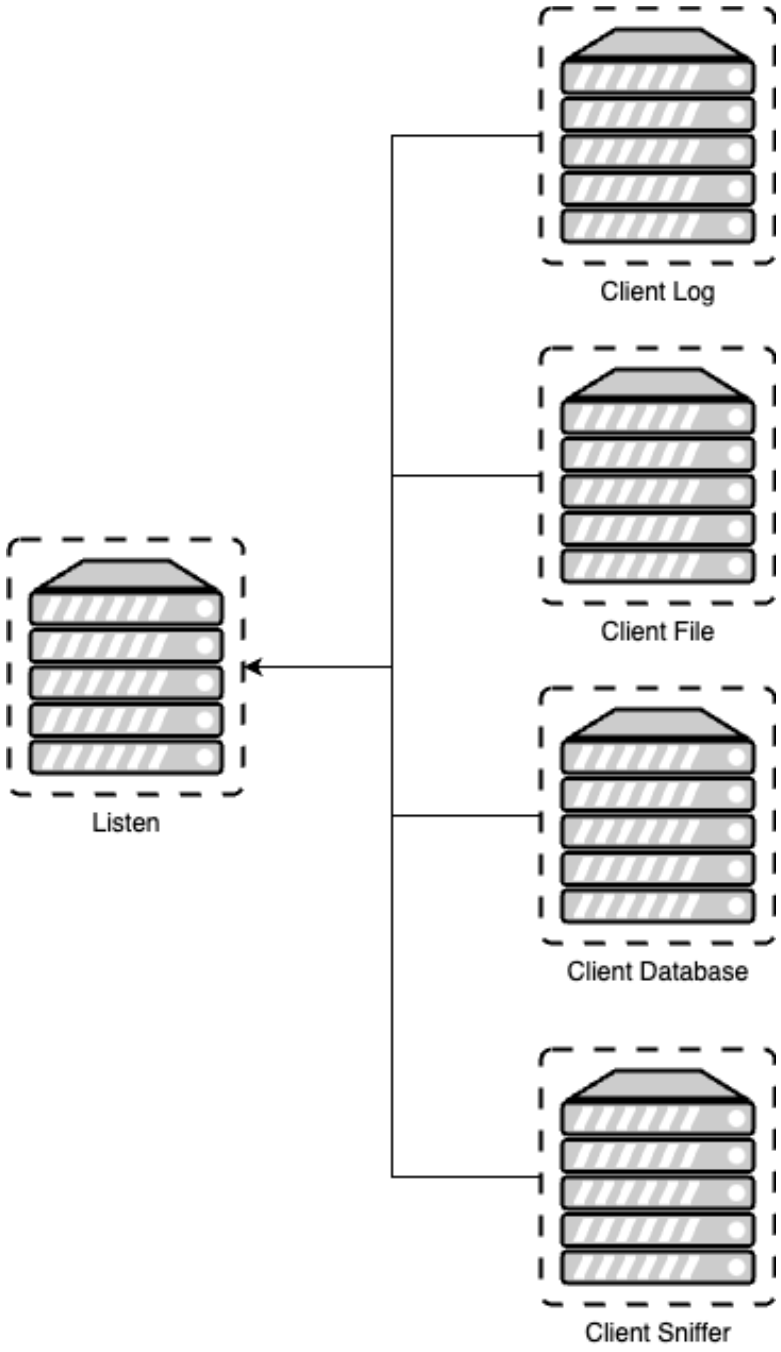
ภาพที่ 3.29 การประมวลและช่วงของการตรวจสอบดูผลลัพธ์

3.9 การหาประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องในหน่วยของ ซีพียู, แรม, และการอ่านและการเขียนดิสก์

ในขั้นตอนนี้เป็นขั้นตอนสำหรับการหาประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องในหน่วยของซีพียู, แรม, และการอ่านและการเขียนดิสก์ โดยผู้วิจัยจัดเตรียมลูกข่ายอย่างละ 1 แม่ข่าย อีก 1 รวมเป็น 6 โดยทางผู้วิจัยจะกำหนดทรัพยากรเครื่องที่ใช้งานแม่ข่าย ซีพียู (CPU) 2 หน่วยประมวลผล, แรม (RAM) 4 Gigabytes, จำนวนดิสก์ (Disk) 30 Gigabytes ส่วนลูกข่าย ซีพียู (CPU) 1 หน่วยประมวลผล, แรม (RAM) 2 จี Gigabytes, จำนวนดิสก์ (Disk) 25 Gigabytes โดยลูกข่ายจากทั้งหมด 5 ลูกข่าย ในการทดสอบผู้วิจัยได้เขียนฟังก์ชันในการตรวจสอบขึ้นมาเอง โดยอ่านค่าจาก System และคัดเลือกเฉพาะกระบวนการทำงานของระบบเท่านั้น

3.10 การทดลองรองรับลูกข่ายที่เชื่อมต่อแม่ข่ายและความเร็วในเสร็จสิ้นการทำงาน ต่อ 1 ครั้ง

ในขั้นตอนนี้เป็นขั้นตอนสำหรับการหาประสิทธิภาพที่แม่ข่ายสามารถรองรับจำนวนของลูกข่ายได้เท่าไหนในเวลาพร้อมกัน เพื่อตรวจสอบดูผลลัพธ์ของระบบแม่ข่ายที่ได้ออกแบบมาว่ามีสิทธิของการทำคิวและจำนวนโหลดที่เข้ามาเชื่อมต่อกับแม่ข่ายโดยจะนำลูกข่ายที่ได้จากข้อ 3.9 และสร้างเพิ่มขึ้น มาทำการต่อยอดและจะมีการเพิ่มจำนวนขึ้นไปเมื่อแม่ข่ายยังสามารถรองรับจาก 20 ลูกข่ายเพิ่มจำนวนขึ้น N + 1 เช่น ลูกข่ายมีอยู่แล้ว 20 รองรับได้ โดยจะนำลูกข่ายที่ 21 เข้ามาเชื่อมต่อต่อไปจนกว่าจะเจอค่าที่แม่ข่ายรับมาได้เจอหรือตกหล่น และตรวจสอบช่วงเวลาที่เริ่มทำงานของลูกข่ายว่าสามารถเชื่อมต่อกับแม่ข่ายว่าใช้ระยะเวลาเท่าไหร่ แล้วเสร็จสิ้นกระบวนต่อครั้งใช้เวลาเท่าไหร่ โดยขนาดข้อมูลที่แตกต่างกัน ดังภาพที่ 3.30



ภาพที่ 3.30 ตัวอย่างของการทดลองรับจำนวนของเครื่องลูกข่าย

บทที่ 4

ผลการทดลอง

การทดลองและข้อมูลที่กล่าวขึ้นในบทนี้เป็นส่วนที่เกี่ยวข้องกับระบบการตรวจสอบข้อมูลและนำเข้าข้อมูลระบบจัดการข้อมูลในรูปแบบที่ต้องการ โดยการนำระบบที่ได้พัฒนาขึ้นมาทดสอบหาค่าความถูกต้องและข้อผิดพลาดที่จะเกิดขึ้นของการตรวจสอบข้อมูลและนำเข้าข้อมูลจากระบบ ในงานวิจัยนี้มีการทดลองเพื่อเป็นทางเลือกให้กับการนำเข้าข้อมูลจราจรคอมพิวเตอร์, ไฟล์เอกสาร, และไฟล์เอกสารที่สำคัญ อยู่ 5 ผลการทดลอง โดยผู้วิจัยจะนิยามตัวย่อยของแต่ละประเภทเป็น

L คือ แม่ข่าย (Listen),

DS คือ ฐานข้อมูลหลัก (Database Server)

FS คือ เซิร์ฟเวอร์โปรโตคอล (FTP Server),

SS คือ เซิร์ฟเวอร์รับข้อมูลจราจรคอมพิวเตอร์ (Syslog Server),

L0 คือ ลูกข่ายตรวจสอบข้อมูลจราจรคอมพิวเตอร์ (Log0),

F คือ ลูกข่ายข้อมูลไฟล์เอกสารต่างๆ (File),

D คือ ลูกข่ายข้อมูลฐานข้อมูล (Database),

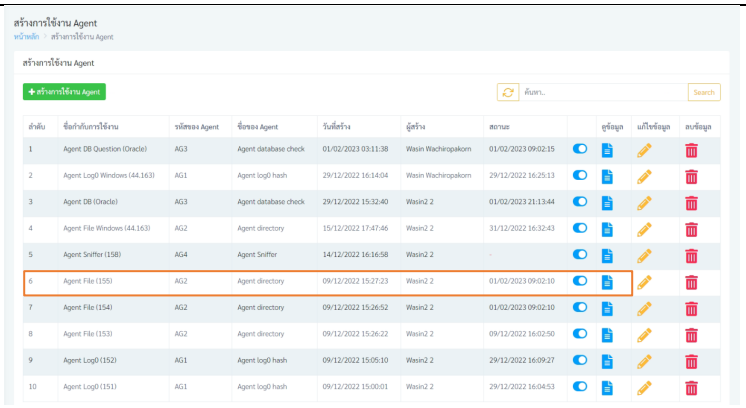
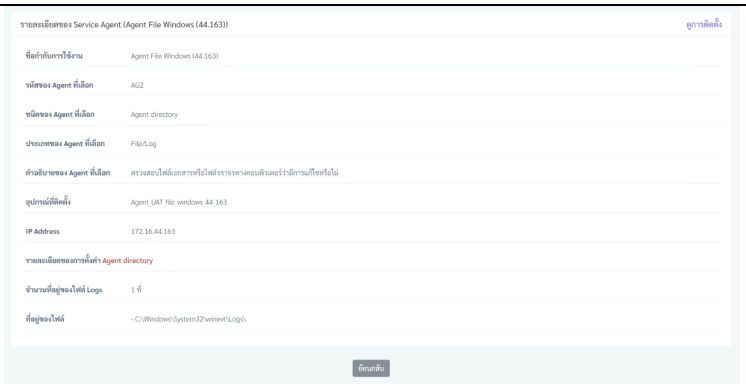
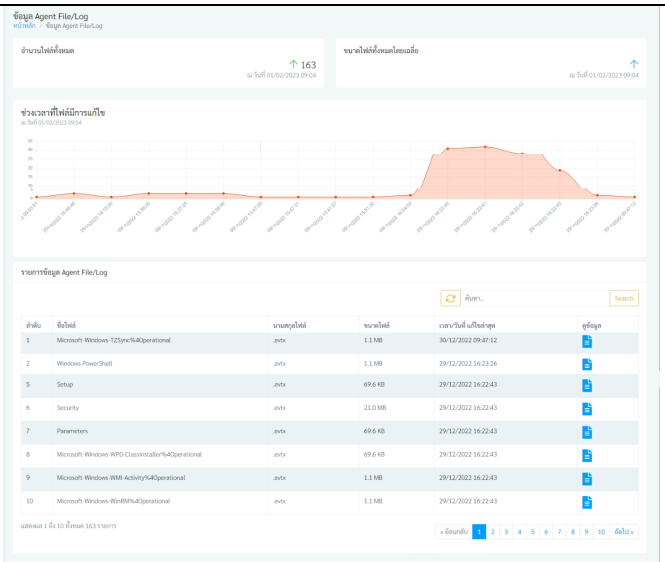
และ S คือ ลูกข่ายข้อมูลจราจร (Sniffer) จะมีรายละเอียดการทดลองดังนี้

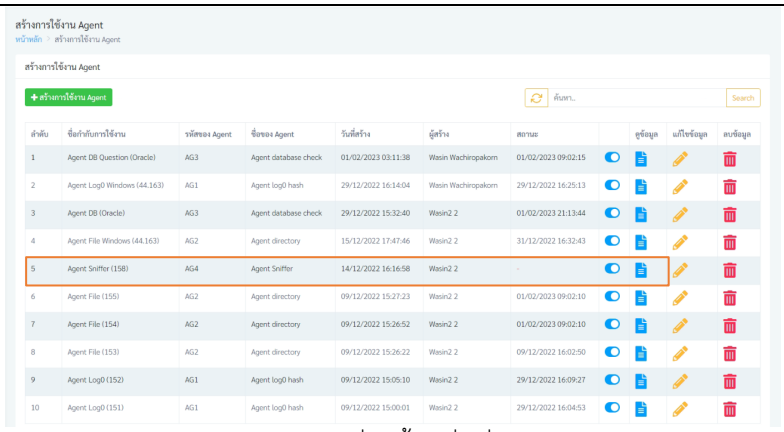
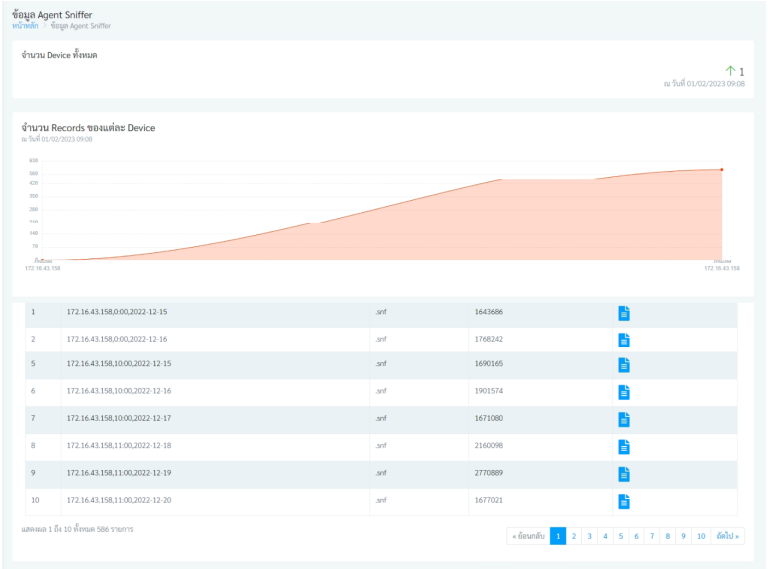
4.1 ผลการตรวจสอบความครบถ้วนตามข้อตกลงของการนำไปใช้งานจริง

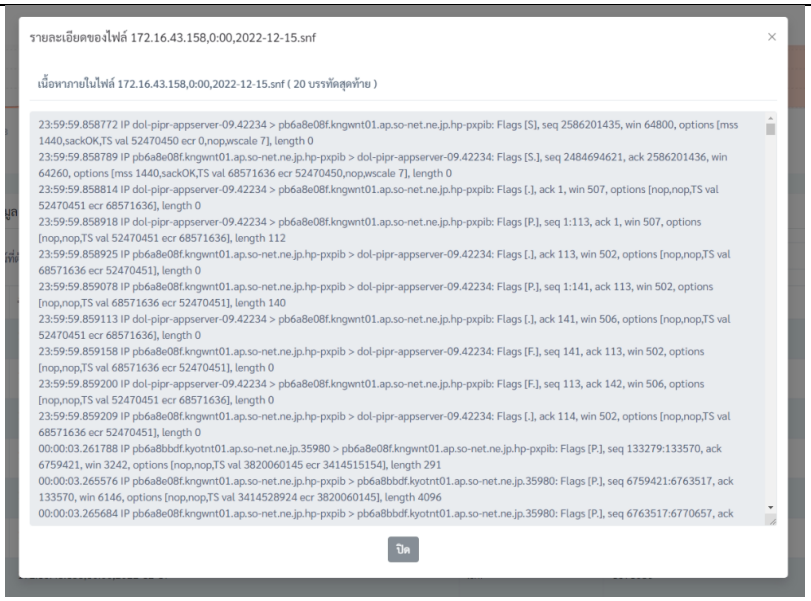
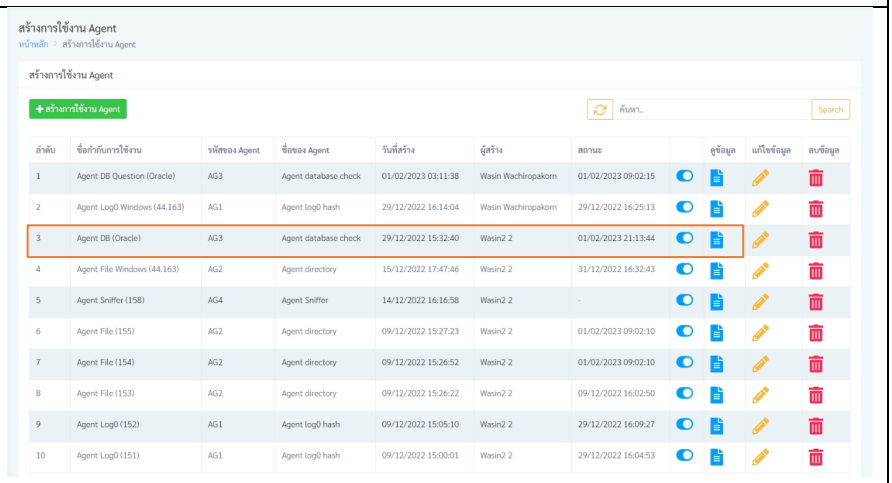
จากผลการทดลองตรวจสอบความครบถ้วนตามข้อตกลงของการนำไปใช้งานจริง โดยข้อมูลที่นำมาใช้งานต่อไปนี้เป็นข้อมูลจากทางเอกสารจริงของทางหน่วยงานหน่วยงานหนึ่งที่ผู้วิจัยได้นำระบบไปใช้งานจริง เพื่อสามารถยืนยันว่าผ่านและนำไปใช้งานได้จริงโดยไม่มีจุดตกหลุนบกพร่อง ตารางต่อไปนี้เป็นข้อมูลที่ได้จากการยืนยันการตรวจรับจากการตรวจสอบดังตาราง 4.1

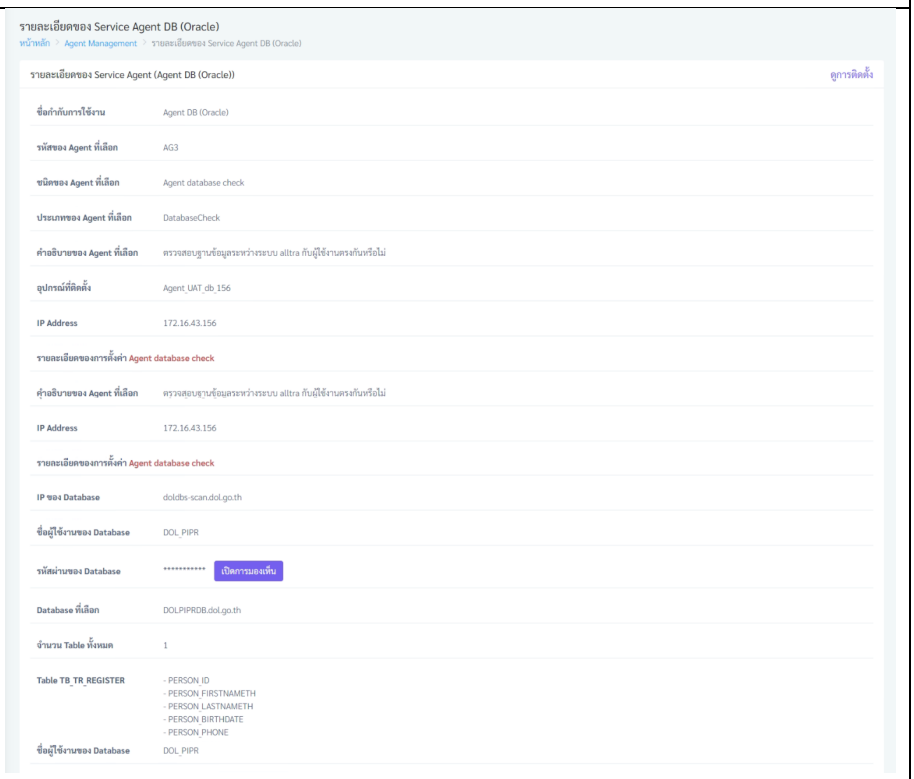
ตารางที่ 4.1 ข้อมูลการตรวจสอบความครบถ้วนตามข้อตกลงของการนำไปใช้งานจริงมีข้อกำหนดและรายละเอียดของการตรวจสอบระบบดังนี้

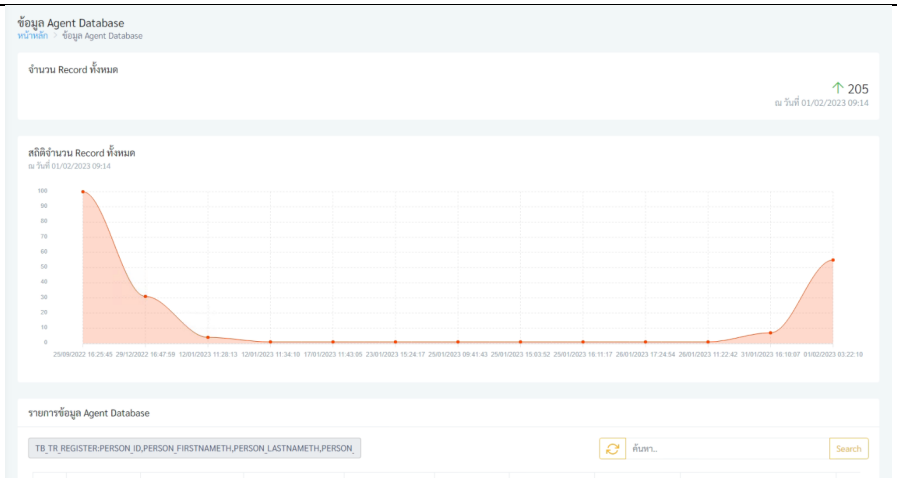
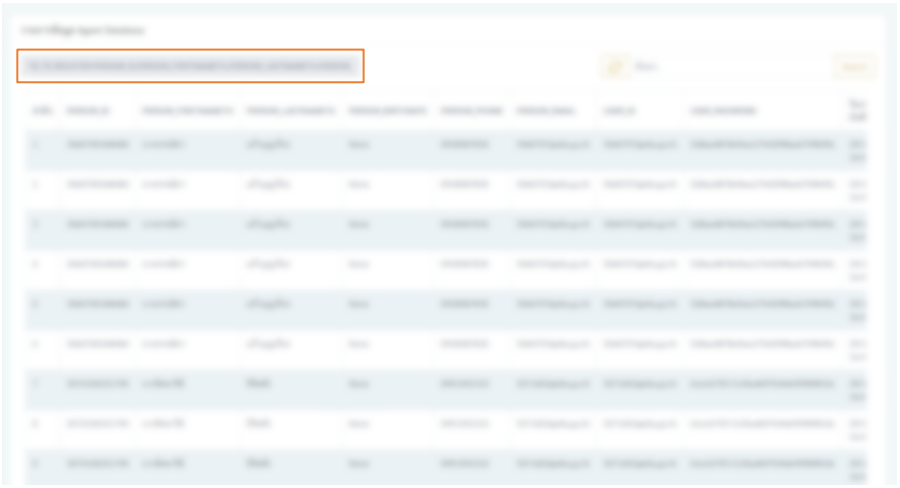
ข้อกำหนด	รายละเอียดตรวจสอบระบบ
1) โปรแกรมที่ลงในเครื่องคอมพิวเตอร์หรือเครื่องคอมพิวเตอร์แม่ข่าย เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้	ขั้นตอนการตรวจสอบ - เรียกใช้งานระบบ http://IP:Port/ ผ่าน Web Browser - เข้าสู่ระบบด้วยผู้ใช้ประเภท “ผู้ควบคุมข้อมูลส่วนบุคคล (Data Controller)” - เลือกเมนู “Data Storage > จัดการ Agent” - หน้าจอ “สร้างการใช้งาน Agent”

ข้อกำหนด	รายละเอียดตรวจสอบระบบ
1) โปรแกรมที่ลงในเครื่องคอมพิวเตอร์หรือเครื่องคอมพิวเตอร์แม่ข่าย เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้ (ต่อ)	 - ยกตัวอย่าง Agent File Windows (44.163) ที่ติดตั้งไว้ที่เครื่อง IP Address: XXX - ระบบจะนำส่งไฟล์ที่อยู่ใน Path ที่กำหนด นำมาเก็บไว้ในระบบ XXX PDPA
	 - เลือกเมนู “Data Storage > ข้อมูล Agent File/Log” - หน้าจอ “ข้อมูล Agent File/Log” - แสดงรายการไฟล์ที่จัดเก็บ
	

ข้อกำหนด	รายละเอียดตรวจสอบระบบ
2) อุปกรณ์ Hardware หรือ Agent โดยนำไป Tap หรือติดตั้ง หรือเชื่อมต่อกับ Switch แบบ Mirror เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการในเครื่องคอมพิวเตอร์หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้	ขั้นตอนการตรวจสอบ - เรียกใช้งานระบบ http://IP:Port/ ผ่าน Web Browser - เข้าสู่ระบบด้วยผู้ใช้ประเภท “ผู้ควบคุมข้อมูลส่วนบุคคล (Data Controller)” - เลือกเมนู “Data Storage > จัดการ Agent” - หน้าจอ “สร้างการใช้งาน Agent”
	 - ยกตัวอย่าง Agent Sniffer (158) ที่ติดตั้งไว้ที่เครื่อง IP Address: XXX - ระบบจะตรวจสอบข้อมูล Transaction และจัดส่งนำมาเก็บไว้ในระบบ XXX PDPA
	- เลือกเมนู “Data Storage > ข้อมูล Agent Sniffer” - หน้าจอ “ข้อมูล Agent Sniffer”  - เลือกอุปกรณ์ เพื่อแสดงรายการไฟล์ที่จัดเก็บ สามารถกดปุ่ม “ดูข้อมูล” เพื่อดูข้อมูลในไฟล์

ข้อกำหนด	รายละเอียดตรวจสอบระบบ
2) อุปกรณ์ Hardware หรือ Agent โดยนำไป Tap หรือติดตั้งหรือเชื่อมต่อกับ Switch แบบ Mirror เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้ (ต่อ)	
3) โปรแกรมที่อยู่ในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อเชื่อมกับ Database ของโปรแกรมในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อทำการตรวจสอบข้อมูลส่วนบุคคลใน Database นั้น	ขั้นตอนการตรวจสอบ - เรียกใช้งานระบบ http://IP:Port/ ผ่าน Web Browser - เข้าสู่ระบบด้วยผู้ใช้ประเภท “ผู้ควบคุมข้อมูลส่วนบุคคล (Data Controller)” - เลือกเมนู “Data Storage > จัดการ Agent” - หน้าจอ “สร้างการใช้งาน Agent”
	 - ยกตัวอย่าง Agent DB (Oracle) ที่ติดตั้งไว้ที่เครื่อง IP Address: XXX - ระบบจะนำส่งข้อมูลในฐานข้อมูล XXX_PIPR

ข้อกำหนด	รายละเอียดตรวจสอบระบบ
3) โปรแกรมที่อยู่ในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อเชื่อมกับ Database ของโปรแกรมในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อทำการตรวจสอบข้อมูลส่วนบุคคลใน Database นั้น (ต่อ)	ตาราง TB_TR_REGISTER - PERSON_ID - PERSON_FIRSTNAMETH - PERSON_LASTNAMETH - PERSON_BIRTHDATE - PERSON_PHONE - PERSON_EMAIL - USER_ID - USER_PASSWORD นำมาเก็บไว้ในระบบ XXX PDPA
	
	- เลือกเมนู “Data Storage > ข้อมูล Agent Database” - หน้าจอ “ข้อมูล Agent Database”

ข้อกำหนด	รายละเอียดตรวจสอบระบบ
3) โปรแกรมที่อยู่ในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อเชื่อมกับ Database ของโปรแกรมในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อทำการตรวจสอบข้อมูลส่วนบุคคลใน Database นั้น (ต่อ)	
	- เลือกรายการ TB_TR_REGISTER: เพื่อแสดงรายการข้อมูลที่จัดเก็บ 

จากตารางที่ 4.1 ข้อมูลการทดลองตรวจสอบความครบถ้วนตามข้อตกลงของการนำไปใช้งานจริง จะเป็นภาพที่แสดงอยู่เว็บแอพพลิเคชัน โดยแนวทางการนำไปใช้งานจริงจะแบบออกเป็นมี 3 หัวข้อหลัก คือ

4.1.1. โปรแกรมที่ลงในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่าย เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการในเครื่องคอมพิวเตอร์หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้ เป็นการตรวจสอบดูผลลัพธ์ของลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์นั้นได้ส่งข้อมูลมาจริงหรือไม่ และแสดงผลลัพธ์ได้ถูกต้องตามที่ออกแบบหรือไม่

4.1.2. อุปกรณ์ Hardware หรือ Agent โดยนำไป Tap หรือติดตั้ง หรือเชื่อมต่อกับ Switch แบบ Mirror เพื่อตรวจสอบข้อมูลส่วนบุคคลที่ต้องการในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายนั้นได้ เป็นการตรวจสอบดูผลลัพธ์ของลูกข่ายประเภทข้อมูลจราจร รูปแบบสร้างไฟล์และส่งไฟล์นั้นได้ส่งข้อมูลมาจริงหรือไม่ และแสดงผลลัพธ์ได้ถูกต้องตามที่ออกแบบหรือไม่

4.1.3. โปรแกรมที่อยู่ในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อเชื่อมกับ Database ของโปรแกรมในเครื่องคอมพิวเตอร์ หรือเครื่องคอมพิวเตอร์แม่ข่ายเพื่อทำการตรวจสอบข้อมูลส่วนบุคคลใน Database นั้น เป็นการตรวจสอบดูผลลัพธ์ของลูกข่ายประเภทฐานข้อมูลนั้นได้ส่งข้อมูลได้ส่งข้อมูลมาจริงหรือไม่ และแสดงผลลัพธ์ได้ถูกต้องตามที่ออกแบบหรือไม่

ดังนั้นเนื่องจากเว็บแอพพลิเคชันสามารถเห็นผลลัพธ์ได้อย่างชัดเจน เพราะระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูลจะทำงานอยู่เบื้องหลังของระบบจึงไม่อาจมองเห็นผลลัพธ์ได้ เป็นการตรวจสอบดูผลลัพธ์ของลูกข่ายประเภทข้อมูลจราจร

4.2 ผลการทดลองตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ภายในเน็ตเวิร์คเดียวกันในแต่ละความเร็ว

จากผลการทดลองตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ โดยข้อมูลที่นำมาใช้งานต่อไปนี้เป็นค่าของบัพเพอร์ (Buffer) ที่เป็นข้อมูลตัวอย่างคล้ายกับข้อมูลจริงที่ใช้งานบนระบบของข้อมูลไฟล์จราจรคอมพิวเตอร์, ข้อมูลไฟล์ประเภทซีเอสวี, และข้อมูลในฐานข้อมูล โดยทดสอบจำนวน 10 ครั้ง ทางผู้วิจัยได้ทำการทดลองส่งข้อมูลจากลูกข่ายไปยังแม่ข่าย เพื่อตรวจสอบดูผลลัพธ์ข้อมูลครบถ้วนหรือไม่ ตารางต่อไปนี้เป็นข้อมูลที่ได้จากการทดลองดังตารางที่ 4.2 ถึงตาราง 4.3

ตารางที่ 4.2 ข้อมูลทดลองตรวจสอบดูความถูกต้องและครบถ้วนข้อมูลของแม่ข่ายและลูกข่ายโดยความเร็วการส่งข้อมูล 100 Mbps, เซิร์ฟเวอร์เอฟทีพี, หรือ เซิร์ฟเวอร์และลูกข่ายประเภทที่เกี่ยวข้องต่างๆ ทดลองจำนวน 10 ครั้ง โดยกำหนดความเร็วการส่งข้อมูล 100Mbps ต่อ 1 ครั้ง

ประเภท	จำนวนครั้ง (จำนวนที่ได้รับ / จำนวนที่ส่ง)										เฉลี่ย
	1	2	3	4	5	6	7	8	9	10	
Listen	312	312	312	312	312	312	312	312	312	312	
Log0	312	312	312	312	312	312	312	312	312	312	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Listen	171	171	171	171	171	171	171	171	171	171	
File	171	171	171	171	171	171	171	171	171	171	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Listen	184	184	184	184	184	184	184	184	184	184	
Database	184	184	184	184	184	184	184	184	184	184	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Database	10874	10874	10874	10874	10874	10874	10874	10874	10874	10874	
Database	10874	10874	10874	10874	10874	10874	10874	10874	10874	10874	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
FTP Server	36311	36450	36532	36633	36710	36730	36809	36921	37023	37119	
Sniffer FTP	36311	36450	36532	36633	36710	36730	36809	36921	37023	37119	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Syslog	173	128	110	174	126	126	160	160	174	157	
Sniffer Log	173	128	110	174	126	126	160	160	174	157	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
เฉลี่ยรวมทั้งหมด (%)											**99.96**

จากตารางที่ 4.2 ข้อมูลการทดลองตรวจสอบดูความถูกต้องและครบถ้วนของข้อมูลในแต่ละครั้งนั้น แต่จะมีจุดสังเกตุในส่วนของประเภทตัววิเคราะห์ข้อมูล (Sniffer) และ เซิร์ฟเวอร์เอฟทีพีนั้นข้อมูลจะห่วงช่วงกันเป็นลักษณะฟันปลา ข้อมูลที่ได้จากการทดลองเป็นผลจากการเสร็จสิ้นกระบวนการนั้นๆ โดยผลลัพธ์เฉลี่ยทั้งหมดคิดเป็นเปอร์เซ็นต์ (%) คือ 100% จากนั้นตามโดยทดลองอีกครั้งในตารางที่ 4.3

ตารางที่ 4.3 ข้อมูลทดลองตรวจสอบดูความถูกต้องและครบถ้วนข้อมูลของแม่ข่ายและลูกข่ายโดยความเร็วการส่งข้อมูล 1000 Mbps, เซิร์ฟเวอร์เอฟทีพี, หรือ เซิร์ฟเวอร์และลูกข่ายประเภทที่เกี่ยวข้องต่างๆ ทดลองจำนวน 10 ครั้ง โดยกำหนดความเร็วการส่งข้อมูล 1000Mpbs ต่อ 1 ครั้ง

ประเภท	จำนวนครั้ง (จำนวนที่ได้รับ / จำนวนที่ส่ง)										เฉลี่ย
	1	2	3	4	5	6	7	8	9	10	
Listen	314	314	314	314	314	314	314	314	314	314	
Log0	314	314	314	314	314	314	314	314	314	314	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Listen	172	172	172	172	172	172	172	172	172	172	
File	172	172	172	172	172	172	172	172	172	172	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Listen	189	189	189	189	189	189	189	189	189	189	
Database	189	189	189	189	189	189	189	189	189	189	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Database	10872	10872	10872	10872	10872	10872	10872	10872	10872	10872	
Database	10872	10872	10872	10872	10872	10872	10872	10872	10872	10872	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
FTP Server	37233	37311	37459	37607	37698	37754	37862	37991	38033	38111	
Sniffer FTP	37233	37311	37459	37607	37698	37754	37862	37991	38033	38111	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
Syslog	131	133	126	126	110	174	126	126	173	126	
Sniffer Log	131	133	126	126	110	174	126	126	173	126	100
ครบถ้วน (%)	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	
เฉลี่ยรวมทั้งหมด (%)											**99.96**

จากตารางที่ 4.3 ข้อมูลการทดลองตรวจสอบดูความถูกต้องและครบถ้วนของข้อมูลในแต่ละครั้งนั้น แต่จะข้อมูลที่ได้จากการทดลองเป็นผลจากการเสร็จสิ้นกระบวนการนั้นๆ โดยผลลัพธ์เฉลี่ยทั้งหมดคิดเป็นเปอร์เซ็นต์ (%) คือ 100% เหมือนกับตารางที่ 4.2 ก่อนหน้า ดังนั้นสรุปแล้วนั้นในเมื่อความเร็วเพียง 100Mbps ก็ยังสามารถส่งข้อมูลครบถ้วนเมื่อขนาดนั้นที่มากกว่านั้นก็ไม่อาจจะทำให้เกินข้อมูลหายไปไหนได้นอกจากเน็ตเวิร์คภายในมีปัญหาเพียงเท่านั้น

4.3 ผลการทดลองการตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานทุกฟังก์ชันของแต่ละประเภท

จากผลการทดลองตรวจสอบดูผลลัพธ์ของการทำงานแต่ละฟังก์ชันของประเภทนั้นๆ รวมกันใช้เวลาไปเท่าไหร่ตอครั้งการทำงานได้ผลการทดลองหน่วยเป็นวินาที โดยทดลองจำนวน 10 ครั้ง ทางผู้วิจัยได้ทำการเขียน

ฟังก์ชันการตรวจเช็คเวลาขึ้นมาทดลองเมื่อลูกข่ายและแม่ข่ายส่งข้อมูลหากัน เพื่อตรวจสอบดูผลลัพธ์ว่าใช้เวลาไปเท่าไหร่ในแต่ละรอบการทำงาน ตารางต่อไปนี้เป็นข้อมูลที่ได้จากการทดลองดังตารางที่ 4.5

ตารางที่ 4.4 การทดลองตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานทุกฟังก์ชันของแต่ละประเภท ทดลองจำนวน 10 ครั้ง โดยกำหนดหน่วยของการนับเวลาคือมิลลิวินาที

ครั้ง	ประเภท			
	ตรวจสอบข้อมูลจราจรคอมพิวเตอร์	ส่งไฟล์	ฐานข้อมูล	ข้อมูลจราจรทั้ง 2 รูปแบบ
1	3.00	20,622.20	300.08	7,902.20
2	2.78	20,702.11	278.83	7,610.02
3	2.74	20,823.15	274.56	7,630.40
4	3.10	20,622.20	310.99	7,506.11
5	3.14	20,718.35	314.37	7,523.16
6	2.50	20,942.09	250.26	7,294.38
7	2.35	20,671.03	235.27	6,903.94
8	2.67	20,741.22	267.70	7,022.34
9	2.80	20,819.01	280.69	7,357.28
10	3.00	20,523.17	300.11	7,772.19
เฉลี่ย	2.81	20,718.45	281.29	7,452.20

จากตารางที่ 4.4 การทดลองตรวจสอบดูผลลัพธ์ของเวลาที่ใช่งานของแต่ละฟังก์ชันในแต่ละส่วนของระบบแม่ข่ายและข้อมูล จากที่ได้เห็นในตารางนั้น ประเภทแม่ข่าย และลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์นั้นมีการใช้เวลาในการทำงานน้อย ลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์ใช้เวลาในการทำงานเฉลี่ยคือ 2.81 มิลลิวินาที หรือ 0.0028 หน่วยของ วินาที ต่อมาลูกข่ายประเภทส่งไฟล์ค ที่เป็นแบบส่งไฟล์นั้นมีการใช้งานเวลาในการทำงานเฉลี่ยคือ 20,178.45 มิลลิวินาที ในส่วนของลูกข่ายประเภทส่งไฟล์ หรือก็คือ 20.7184 ในหน่วยของวินาที และจากการทดลองนั้นขนาดไฟล์ที่ส่งนั้นมีขนาดอยู่ที่ 579 MB ดังนั้นเมื่อขนาดไฟล์ที่มากก็จะส่งผลต่อเวลาในการทำงานของลูกข่ายประเภทส่งไฟล์ให้ใช้เวลานานหรือมากขึ้นตามไปด้วย ต่อมาลูกข่ายประเภทฐานข้อมูล มีการใช้เวลาในการทำงานน้อยเช่นกัน โดยใช้เวลาในการทำงานเฉลี่ยคือ 281.29 มิลลิวินาที หรือ 0.2812 ในหน่วยของ วินาที สุดท้ายเป็น ลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์ใช้เวลาการทำงานเฉลี่ยคือ 7,452.20 มิลลิวินาที หรือ 7.4522 ในหน่อยของ วินาที เนื่องจากการทำงานของลูกข่ายประเภทข้อมูลจราจรจะแบ่งออกเป็น 2 ประเภท

ย่อยอีกทีคือ แบบสร้างไฟล์และส่งไฟล์เก็บยังเซิร์ฟเวอร์เอฟทีพี (FTP) และแบบส่งข้อมูลไปให้กับทางเซิร์ฟเวอร์ที่เก็บข้อมูลจราจรคอมพิวเตอร์ (Syslog) โดยจริงๆแล้วประเภทแบบส่งข้อมูลไปให้กับทางเซิร์ฟเวอร์ที่เก็บข้อมูลจราจรคอมพิวเตอร์ (Syslog) น้อย โดยจากที่ผู้วิจัยได้ทดลองมาคืออยู่ที่ประมาณ 1 หรือน้อยกว่านั้นเท่านั้น แต่เนื่องจากอีกรูปแบบหนึ่งคือสร้างไฟล์และส่งไฟล์ไปเก็บยังเซิร์ฟเวอร์เอฟทีพี (FTP) นั้นต้องสร้างไฟล์, เขียนไฟล์, และส่งไฟล์ไปเก็บผลของการทำงานหลายอย่างพร้อมกันทำให้เกิดการใช้เวลาที่นานเฉลี่ยคือ 7,452.20 มิลลิวินาที หรือ 7.4522 ในหน่อยของ วินาที จุดสังเกตสุดท้ายคือในส่วนของลูกข่ายประเภทฐานข้อมูลที่มีค่าเวลาไม่ค่อยจะเท่ากันเลยแต่ก็ไม่ได้ใช้นานเนื่องจากใช้เวลาเฉลี่ยไป 281.29 มิลลิวินาที หรือ 0.2812 ในหน่วยของ วินาที ก็ยังไม่ถึงกับ 1 วินาทีเนื่องโดยที่มีการเหวี่ยงของเวลาเกิดขึ้นจากการนำข้อมูลของฐานข้อมูลที่ต้องการบันทึกลงไปยังฐานข้อมูลหลักต้องตรวจสอบและเตรียมข้อมูลให้พร้อมสำหรับการเพิ่มข้อมูลใหม่, แก้ไขข้อมูลใหม่, หรือลบข้อมูลที่เกินเพื่อให้ทั้งสองฝั่งมีข้อมูลที่ตรงกัน 100% ไม่ว่าจะมีกี่ข้อมูลโดยจากที่ทดลองสูงคือ 10,876 ข้อมูล

4.4 ผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องในหน่วยของ ซีพียู, แรม, และการอ่านและการเขียนดิสก์

จากผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องรวมกันใช้เวลาไปเท่าไหร่ต่อครั้งการทำงานได้ผลการทดลองหน่วยเปอร์เซ็นต์ (%) และหน่วยเอ็มบีหรือเมกะไบต์ (MB หรือ Megabytes) ผ่านการใช้ฟังก์ชันที่ได้เขียนขึ้นมาในการตรวจสอบ โดยทดลองจำนวน 10 ครั้ง ทางผู้วิจัยได้ทำการเขียนฟังก์ชันการตรวจเช็คทรัพยากรของระบบปฏิบัติการที่ทำงานนั้น หรือก็คือ “System” เพื่อเข้าตรวจสอบว่าเครื่องแม่ข่ายและเครื่องลูกข่ายนั้น ใช้งานทรัพยากรเท่าไหร่ต่อทรัพยากรที่กำหนดให้ ขึ้นมาทดลองเมื่อลูกข่ายและแม่ข่ายทำงานครั้งนั้นๆเสร็จเรียบร้อยแล้ว เพื่อตรวจสอบดูผลลัพธ์ว่าใช้ทรัพยากรของเครื่องมากน้อยเท่าไหร่ในแต่ละรอบการทำงาน โดยจะใช้กำหนดทรัพยากรของเครื่องคือ

ระบบปฏิบัติการ (OS) คือ “Oracle Linux x86 v.8u7”, ซีพียู (CPU) คือ 2 หน่วย (Core), แรม (RAM) คือ 2 Gigabytes, และจำนวนฮาร์ทดิสก์ (Disk) คือ 35 Gigabytes ทั้งหมด 7 เครื่องทดสอบบนเครื่องเซิร์ฟเวอร์ของทางห้องปฏิบัติการ ตารางต่อไปนี้เป็นข้อมูลที่ได้จากการทดลองดังตารางที่ 4.5 ถึง 4.11

ตารางที่ 4.5 การทดลองตรวจสอบดูผลลัพธ์ทรัพยากรที่ใช้งานของระบบแม่ข่ายทั้งหมด ทดลองจำนวน 10 ครั้ง โดยกำหนดหน่วยของซีพียู (CPU) เป็นเปอร์เซ็นต์, ส่วนแรม (RAM) เป็นเอ็มบี หรือเมกะไบต์ (MB หรือ Megabytes), และส่วนสุดท้ายคือเรื่องของการอ่านการเขียนดิสก์ (Disk) เป็นเป็นเอ็มบีต่อวินาที หรือเมกะไบต์ ต่อวินาที (MB/s หรือ Megabytes per second)

ครั้ง	ซีพียู (%)	แรม (MB)	การอ่านดิสก์ (MB/s)	การเขียนดิสก์ (MB/s)
1	5.40	0.25	691.25	134.20
2	5.90	8.81	691.25	134.20
3	5.90	8.81	691.25	134.20
4	3.10	8.81	691.25	134.20
5	3.00	1.38	691.25	134.20
6	3.10	8.81	673.96	134.20
7	2.80	6.59	673.96	134.20
8	3.00	67.09	690.35	134.20
9	3.00	0.26	651.88	134.20
10	1.60	12.56	651.88	134.20
เฉลี่ย	3.70	12.33	679.83	134.20

จากตาราง 4.5 จากผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องส่วนของแม่ข่ายทั้งหมดใช้งานในการทำงานได้ผลการทดลองหน่วยเปอร์เซ็นต์และหน่วยเอ็มบีหรือเมกะไบต์ (MB หรือ Megabytes) เฉลี่ยทั้งหมดคือ ซีพียู (CPU) ได้ 3.70% แรม (RAM) ได้ 12.33 Megabytes ถือว่าน้อยสำหรับการทำงานระบบบริการแต่ก็อาจจะมีช่วงขึ้นสูงเช่น 67.09% เป็นครั้งคร่าวแต่ก็ไม่ถึงขั้นใช้งานทรัพยากรหลัก จีบี หรือจิกะไบต์ (GB หรือ Gigabytes) ส่วนสุดท้ายการอ่านดิสก์ได้ 679.83 Megabytes per second การเขียนดิสก์ได้ 134.20 Megabytes per second ก็ถือว่าเป็นค่ากลางของการทำงานระบบบริการ

ตารางที่ 4.6 การทดลองตรวจสอบดูผลลัพธ์ทรัพยากรที่ใช้งานของระบบลูกข่ายประเภทตรวจสอบข้อมูลจราจรคอมพิวเตอร์ ทดลองจำนวน 10 ครั้ง โดยกำหนดหน่วยของซีพียู (CPU) เป็นเปอร์เซ็นต์, ส่วนแรม (RAM) เป็นเอ็มบี หรือเมกะไบต์ (MB หรือ Megabytes), และส่วนสุดท้ายคือเรื่องของการอ่านการเขียนดิสก์ (Disk) เป็นเป็นเอ็มบีต่อวินาที หรือเมกะไบต์ ต่อวินาที (MB/s หรือ Megabytes per second)

ครั้ง	ซีพียู (%)	แรม (MB)	การอ่านดิสก์ (MB/s)	การเขียนดิสก์ (MB/s)
1	3.20	1.80	1,600.59	148.00
2	3.60	1.80	1,600.59	148.01
3	3.80	1.80	1,600.59	148.02
4	3.70	1.80	1,600.53	148.00
5	4.40	1.80	1,600.53	147.99
6	3.70	1.80	1,594.81	148.01
7	3.50	1.80	1,598.47	148.01
8	3.20	4.19	1,600.53	148.01
9	3.60	1.80	1,600.59	148.00
10	3.70	13.76	1,600.59	147.99
เฉลี่ย	3.60	3.23	1599.78	148.00

จากตารางที่ 4.6 จากผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องส่วนของลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์ในการทำงานได้ผลการทดลองหน่วยเปอร์เซ็นต์และหน่วยเอ็มบีหรือเมกะไบต์ (MB หรือ Megabytes) เฉลี่ยทั้งหมดคือ ซีพียู (CPU) ได้ 3.60% แรม (RAM) ได้ 3.20 Megabytes ก็ยังถือว่าน้อยสำหรับการทำงานระบบบริการ ส่วนสุดท้ายการอ่านดิสก์ได้ 1599.78 Megabytes per second การเขียนดิสก์ได้ 148.00 Megabytes per second ก็ยังถือว่าเป็นค่าที่น้อยสำหรับการทำงานระบบบริหารแต่การอ่านอาจจะมีช่วงขึ้นสูงเช่น 1,600.59 ถึงขั้นใช้งานทรัพยากรหลัก จีบี หรือ จิกะไบต์ (GB หรือ Gigabytes) เป็นครั้งคร่าวในกรณีอ่านไฟล์ใหม่หรือไฟล์ขนาดไม่เท่ากับไฟล์เดิม

ตารางที่ 4.7 การทดลองตรวจสอบดูผลลัพธ์ทรัพยากรที่ใช้งานของระบบลูกข่ายประเภทส่งไฟล์ ทดลองจำนวน 10 ครั้ง โดยกำหนดหน่วยของซีพียู (CPU) เป็นเปอร์เซ็นต์, ส่วนแรม (RAM) เป็นเอ็มบี หรือเมกะไบต์ (MB หรือ Megabytes), และส่วนสุดท้ายคือเรื่องของการอ่านการเขียนดิสก์ (Disk) เป็นเป็นเอ็มบีต่อวินาที หรือเมกะไบต์ ต่อวินาที (MB/s หรือ Megabytes per second)

ครั้ง	ซีพียู (%)	แรม (MB)	การอ่านดิสก์ (MB/s)	การเขียนดิสก์ (MB/s)
1	5.00	0.00	853.00	158.65
2	1.10	0.00	1,234.66	158.64
3	3.70	0.71	1,355.74	158.65
4	3.80	0.25	1,400.32	158.66
5	3.30	0.00	1651.69	158.67
6	3.30	0.00	1,827.99	158.67
7	3.20	0.00	1651.69	158.66
8	3.50	0.00	1,400.32	158.65
9	3.30	0.00	1,355.74	158.64
10	3.10	0.00	1,234.66	158.65
เฉลี่ย	3.30	0.10	1396.58	158.65

จากตารางที่ 4.7 จากผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องส่วนของลูกข่ายประเภทส่งไฟล์ในการทำงานได้ผลการทดลองหน่วยเปอร์เซ็นต์และหน่วยเอ็มบีหรือเมกะไบต์ (MB หรือ Megabytes) เฉลี่ยทั้งหมดคือ ซีพียู (CPU) ได้ 3.30% แรม (RAM) ได้ 0.10 Megabytes per second ก็ยังถือว่าน้อยสำหรับการทำงานระบบบริการส่งไฟล์ ส่วนสุดท้ายการอ่านดิสก์ได้ 1396.58 Megabytes การเขียนดิสก์ได้ 158.65 Megabytes per second ก็ยังถือว่าเป็นค่ากลางปกติสำหรับการทำงานระบบบริการ แต่การอ่านอาจจะมีช่วงขึ้นสูงเช่น 1,400.32 ถึงขั้นใช้งานทรัพยากรหลัก จีบี หรือ จิกะไบต์ (GB หรือ Gigabytes) เป็นครั้งคร่าวในกรณีอ่านไฟล์ใหม่ ๆ หรือไฟล์ขนาดไม่เท่ากับไฟล์เดิม

ตารางที่ 4.8 การทดลองตรวจสอบดูผลลัพธ์ทรัพยากรที่ใช้งานของระบบลูกข่ายประเภทฐานข้อมูล ทดลองจำนวน 10 ครั้ง โดยกำหนดหน่วยของซีพียู (CPU) เป็นเปอร์เซ็นต์, ส่วนแรม (RAM) เป็นเอ็มบี หรือเมกะไบต์ (MB หรือ Megabytes), และส่วนสุดท้ายคือเรื่องของการอ่านการเขียนดิสก์ (Disk) เป็นเป็นเอ็มบีต่อวินาที หรือเมกะไบต์ ต่อวินาที (MB/s หรือ Megabytes per second)

ครั้ง	ซีพียู (%)	แรม (MB)	การอ่านดิสก์ (MB/s)	การเขียนดิสก์ (MB/s)
1	4.00	0.52	1167.84	1573.02
2	6.80	0.53	1167.84	1573.02
3	6.40	1.99	1167.84	1573.02
4	4.10	1.93	1167.84	1573.03
5	3.70	0.54	1167.84	1573.03
6	3.40	1.98	1167.84	1573.04
7	7.00	0.54	1167.84	1573.04
8	7.20	2.09	1167.84	1573.04
9	7.60	0.18	1167.84	1573.09
10	7.10	0.26	1167.84	1573.09
เฉลี่ย	5.70	1.06	1167.84	1573.04

จากตารางที่ 4.8 จากผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องส่วนของลูกข่ายประเภทฐานข้อมูลในการทำงานได้ผลการทดลองหน่วยเปอร์เซ็นและหน่วยเอ็มบีหรือเมกะไบต์ (MB หรือ Megabytes) เฉลี่ยทั้งหมดคือ ซีพียู (CPU) ได้ 5.70% แรม (RAM) ได้ 1.06 Megabytes ถือว่าน้อยสำหรับการทำงานระบบบริการ ส่วนสุดท้ายการอ่านดิสก์ได้ 1167.84 Megabytes per second การเขียนดิสก์ได้ 1573.04 Megabytes per second ถือว่าเป็นการใช้งานเป็นค่าที่สูงเนื่องจากใช้งานทรัพยากรขึ้นถึงหลัก จีบี หรือ จิกะไบต์ (GB หรือ Gigabytes) คือ 1.167 / 1.573 Gigabytes เลยที่เดียวเนื่องจากอาจจะอยู่ที่จำนวนขนาดข้อมูลในฐานข้อมูลนั้นๆที่เลือกตารางนั้น ๆ ด้วยเช่นกัน ทางการทดสอบจะใช้งานข้อมูลฐานข้อมูลที่ 10,876 จำนวนข้อมูล

ตารางที่ 4.9 การทดลองตรวจสอบดูผลลัพธ์ทรัพยากรที่ใช้งานของระบบลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์รูปแบบสร้างไฟล์และส่งไฟล์ ทดลองจำนวน 10 ครั้ง โดยกำหนดหน่วยของซีพียู (CPU) เป็นเปอร์เซ็นต์, ส่วนแรม (RAM) เป็นเอ็มบี หรือเมกะไบต์ (MB หรือ Megabytes), และส่วนสุดท้ายคือเรื่องของการอ่านการเขียนดิสก์ (Disk) เป็นเป็นเป็นเอ็มบีต่อวินาที หรือเมกะไบต์ ต่อวินาที (MB/s หรือ Megabytes per second)

ครั้ง	ซีพียู (%)	แรม (MB)	การอ่านดิสก์ (MB/s)	การเขียนดิสก์ (MB/s)
1	4.00	0.60	240.21	18.62
2	5.20	0.34	240.21	18.62
3	5.00	0.09	240.21	18.63
4	4.80	0.32	240.21	18.63
5	6.30	0.02	240.21	18.64
6	4.70	0.32	240.21	18.64
7	6.30	0.50	240.21	18.64
8	4.70	0.32	240.21	18.64
9	5.10	0.02	240.22	18.67
10	5.30	0.02	240.22	18.68
เฉลี่ย	5.10	0.26	240.21	18.64

จากตารางที่ 4.9 จากผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องส่วนของลูกข่ายประเภทข้อมูลจราจร แบบส่งไฟล์ในการทำงานได้ผลการทดลองหน่วยเปอร์เซ็นและหน่วยเอ็มบีหรือเมกะไบต์ (MB หรือ Megabytes) เฉลี่ยทั้งหมดคือ ซีพียู (CPU) ได้ 5.10% แรม (RAM) ได้ 0.26 Megabytes ถือว่าน้อยสำหรับการทำงานระบบบริการสร้างไฟล์และส่งไฟล์ ส่วนสุดท้ายการอ่านดิสก์ได้ 240.21 Megabytes per second การเขียนดิสก์ได้ 18.64 Megabytes per second ถือว่าเป็นการใช้งานเป็นค่าที่น้อยมากสำหรับการใช้ดิสก์จากประเภทอื่นทั้งหมด

ตารางที่ 4.10 การทดลองตรวจสอบดูผลลัพธ์ทรัพยากรที่ใช้งานของระบบลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์รูปแบบส่งไปเซิร์ฟเวอร์จราจร ทดลองจำนวน 10 ครั้ง โดยกำหนดหน่วยของซีพียู (CPU) เป็นเปอร์เซ็นต์, ส่วนแรม (RAM) เป็นเอ็มบี หรือเมกะไบต์ (MB หรือ Megabytes), และส่วนสุดท้ายคือเรื่องของการอ่านการเขียนดิสก์ (Disk) เป็นเป็นเอ็มบีต่อวินาที หรือเมกะไบต์ ต่อวินาที (MB/s หรือ Megabytes per second)

ครั้ง	ซีพียู (%)	แรม (MB)	การอ่านดิสก์ (MB/s)	การเขียนดิสก์ (MB/s)
1	3.70	0.26	596.96	161.61
2	3.20	0.00	596.96	161.61
3	5.30	0.24	596.96	161.61
4	3.00	0.00	596.96	161.61
5	3.00	0.00	596.96	161.61
6	3.30	0.18	596.96	161.61
7	6.10	0.26	596.96	161.61
8	3.20	0.00	596.96	161.61
9	5.70	0.00	596.96	161.61
10	5.70	0.00	596.96	161.61
เฉลี่ย	4.20	0.10	596.96	161.61

จากตารางที่ 4.10 จากผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องส่วนของลูกข่ายประเภทข้อมูลจราจรแบบสร้างไฟล์ส่งไปยังเซิร์ฟเวอร์ข้อมูลจราจรคอมพิวเตอร์ในการทำงานได้ผลการทดลองหน่วยเปอร์เซ็นและหน่วยเอ็มบีหรือเมกะไบต์ (MB หรือ Megabytes) เฉลี่ยทั้งหมดคือ ซีพียู (CPU) ได้ 4.20% แรม (RAM) 0.10 Megabytes ถือว่าน้อยมาก ๆ สำหรับการทำงานระบบบริการและประเภทอื่นทั้งหมด ส่วนสุดท้ายการอ่านดิสก์ได้ 596.96 Megabytes per second การเขียนดิสก์ได้ 161.64 Megabytes per second ถือว่าเป็นการใช้งานเป็นค่าที่น้อยเป็นอันเสร็จสิ้นของการทดลองประสิทธิภาพทั้งหมดของทุกประเภท

4.5 ผลการทดลองรองรับจำนวนโหลดของเครื่องลูกข่ายและความเร็วในเสร็จสิ้นการทำงานต่อ 1 ครั้ง

จากผลการทดลองประสิทธิภาพของระบบในการใช้งานการเชื่อมต่อกันระหว่างลูกข่ายและแม่ข่ายได้ผลการทดลองสามารถเชื่อมต่อได้หรือไม่และช่วงเวลาที่เริ่มทำงานลูกข่ายที่ส่งค่ามาให้ยังแม่ข่าย โดยทดลองจำนวน 20 ลูกข่าย โดยจะใช้กำหนดทรัพยากรของเครื่องลูกข่ายคือ ระบบปฏิบัติการ (OS) คือ “Oracle Linux x86 v.8u7”, ซีพียู (CPU) คือ 1 หน่วย (Core), แรม (RAM) คือ 2 Gigabytes, และจำนวนฮาร์ทดิสก์ (Disk) คือ 25 Gigabytes ทำเพื่อตรวจสอบดูผลลัพธ์ว่ารองรับลูกข่ายได้มากน้อยเท่าไหร่และถ้ามากน้อยเวลาที่ลูกข่ายจะเชื่อมต่อได้กับแม่ข่ายใช้เวลาเท่าไหร่ครั้งแรกและสิ้นสุดการทำงานในครั้งแรกเท่านั้น ตารางต่อไปนี้เป็นข้อมูลที่ได้จากการทดลองดังตารางที่ 4.11 ถึง 4.17

ตารางที่ 4.11 การทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์เริ่มงานทำงานครั้งแรกและสิ้นสุดครั้งนั้นเท่าไหร่ โดยจะทดลองจำนวน 3 ลูกข่ายที่ขนาดข้อความเท่ากันคือประมาณ 300 bytes ผลลัพธ์ที่ได้คือสามารถเชื่อมต่อผ่านหรือไม่รวมถึงใช้เวลาในการเชื่อมต่อกับแม่ข่ายเร็วหรือนานเท่าไหร่

ครั้งที่	ประเภท	เวลาเริ่มทำงาน	เวลาที่หยุดเชื่อมต่อ	เวลาที่ใช้เชื่อมต่อ (วินาที)	ค่าเฉลี่ยอัตราความเร็ว
1	Log0	6/8/2023 14:34	6/8/2023 14:36	3.79	79 bytes ต่อวินาที
2	Log0	6/8/2023 15:13	6/8/2023 15:16	3.49	85 bytes ต่อวินาที
3	Log0	6/8/2023 16:09	6/8/2023 16:13	4.03	74 bytes ต่อวินาที

จากตางรางที่ 4.11 จากผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย 3 ลูกข่ายแรกก็ประสบผลสำเร็จไม่เกิดปัญหาใด ๆ ด้วยจากที่เวลาที่เริ่มทำงานกับหลังจากเพิ่มประวัติบันทึกลงไปยังเว็บแอพพลิเคชัน ในส่วนของประเภทข้อมูลจราจรคอมพิวเตอร์โดยมีค่าข้อความส่งไปให้กับทางแม่ข่ายอยู่ที่ 300 ไบต์ (Bytes) ได้ค่าเฉลี่ยในการบันทึกได้ 3.77 วินาที

ตารางที่ 4.12 การทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์เริ่มงานทำงานครั้งแรกและสิ้นสุดครั้งนั้นเท่าไหร่ โดยจะทดลองจำนวน 3 ลูกข่ายถัดมาที่ขนาดข้อมูลเท่ากันคือประมาณ 280 bytes แต่แตกต่างจาก 3 ลูกข่ายก่อนหน้า ผลลัพธ์ที่ได้คือสามารถเชื่อมต่อผ่านหรือไม่รวมถึงใช้เวลาในการเชื่อมต่อกับแม่ข่ายเร็วหรือนานเท่าไหร่

ครั้งที่	ประเภท	เวลาเริ่มทำงาน	เวลาที่หยุดเชื่อมต่อ	เวลาที่ใช้เชื่อมต่อ (วินาที)	ค่าเฉลี่ยอัตราความเร็ว
4	Log0	9/8/2023 11:24	9/8/2023 11:27	3.86	72 bytes ต่อวินาที
5	Log0	9/8/2023 11:37	9/8/2023 11:40	3.71	75 bytes ต่อวินาที
6	Log0	9/8/2023 11:45	9/8/2023 11:48	3.26	85 bytes ต่อวินาที

จากตางรางที่ 4.12 จากผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย 6 ลูกข่ายก็ประสบผลสำเร็จไม่เกิดปัญหาใด ๆ ด้วยจากที่เวลาที่เริ่มทำงานกับหลังจากเพิ่มประวัติบันทึกลงไปยังเว็บแอพพลิเคชัน ในส่วนของประเภทข้อมูลจราจรคอมพิวเตอร์โดยมีค่าข้อความส่งไปให้กับทางแม่ข่ายอยู่ที่ 280 ไบต์ (Bytes) ได้ค่าเฉลี่ยในการบันทึกได้ 3.61 วินาที

ตารางที่ 4.13 การทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่ายประเภทส่งไฟล์ เริ่มงานทำงานครั้งแรกและสิ้นสุดครั้งนั้นเท่าไหร่ โดยจะทดลองจำนวน 3 ลูกข่ายถัดมาที่ขนาดข้อมูลเท่ากันคือ ไฟล์ประมาณ 140 Megabyte แต่แตกต่างจาก 6 ลูกข่ายก่อนหน้า ผลลัพธ์ที่ได้คือสามารถเชื่อมต่อผ่านหรือไม่รวมถึงใช้เวลาในการเชื่อมต่อกับแม่ข่ายเร็วหรือนานเท่าไหร่

ครั้งที่	ประเภท	เวลาเริ่มทำงาน	เวลาที่หยุดเชื่อมต่อ	เวลาที่ใช้เชื่อมต่อ (วินาที)	ค่าเฉลี่ยอัตราความเร็ว
7	File	6/8/2023 16:22	6/8/2023 16:32	10.35	13 Megabytes ต่อวินาที
8	File	6/8/2023 17:18	6/8/2023 17:27	9.13	15 Megabytes ต่อวินาที
9	File	6/8/2023 17:39	6/8/2023 17:53	14.42	9 Megabytes ต่อวินาที

จากตางรางที่ 4.13 จากผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย 9 ลูกข่ายก็ประสบผลสำเร็จไม่เกิดปัญหาใด ๆ ด้วยจากที่เวลาที่เริ่มทำงานกับหลังจากเพิ่มประวัติบันทึกลงไปยังเว็บแอพพลิเคชัน ในส่วนของประเภทส่งไฟล์โดยมีค่าขนาดไฟล์ที่ส่งไปให้กับทางแม่ข่ายอยู่ที่ 140 Megabytes ได้ค่าเฉลี่ยในการบันทึกได้ 11.3 วินาที

ตารางที่ 4.14 การทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่ายประเภทส่งไฟล์ เริ่มงานทำงานครั้งแรกและสิ้นสุดครั้งนั้นเท่าไหร่ โดยจะทดลองจำนวน 3 ลูกข่ายถัดมาที่ขนาดข้อมูลโดยจะทดลองจำนวน 3 ลูกข่ายถัดมาที่ขนาดข้อมูลเท่ากันคือ ไฟล์ประมาณ 260 Megabyte แต่แตกต่างจาก 9 ลูกข่ายก่อนหน้า ผลลัพธ์ที่ได้คือสามารถเชื่อมต่อผ่านหรือไม่รวมถึงใช้เวลาในการเชื่อมต่อกับแม่ข่ายเร็วหรือนานเท่าไหร่

ครั้งที่	ประเภท	เวลาเริ่มทำงาน	เวลาที่หยุดเชื่อมต่อ	เวลาที่ใช้เชื่อมต่อ (วินาที)	ค่าเฉลี่ยอัตราความเร็ว
10	File	9/8/2023 11:57	9/8/2023 12:14	17.67	14 Megabytes ต่อวินาที
11	File	9/8/2023 12:00	9/8/2023 12:18	18.87	13 Megabytes ต่อวินาที
12	File	9/8/2023 12:03	9/8/2023 12:20	17.63	14 Megabytes ต่อวินาที

จากตางรางที่ 4.14 จากผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย 12 ลูกข่ายก็ประสบผลสำเร็จไม่เกิดปัญหาใด ๆ ด้วยจากที่เวลาที่เริ่มทำงานกับหลังจากเพิ่มประวัติบันทึกลงไปยังเว็บแอพพลิเคชัน ในส่วนของประเภทส่งไฟล์โดยมีค่าขนาดไฟล์ที่ส่งไปให้กับทางแม่ข่ายอยู่ที่ 260 Megabytes ได้ค่าเฉลี่ยในการบันทึกได้ 18.06 วินาที

ตารางที่ 4.15 การทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่ายประเภทฐานข้อมูล เริ่มงานทำงานครั้งแรกและสิ้นสุดครั้งนั้นเท่าไหร่ โดยจะทดลองจำนวน 3 ลูกข่ายถัดมาที่ขนาดข้อมูลไม่เท่ากันคือ MySQL จำนวนข้อมูล 10,872, Oracle Database จำนวนข้อมูล 7,611 กับ 7,862 และแตกต่างจาก 12 ลูกข่ายก่อนหน้า ผลลัพธ์ที่ได้คือสามารถเชื่อมต่อผ่านหรือไม่รวมถึงใช้เวลาในการเชื่อมต่อกับแม่ข่ายเร็วหรือนานเท่าไหร่

ครั้งที่	ประเภท	เวลาเริ่มทำงาน	เวลาที่หยุดเชื่อมต่อ	เวลาที่ใช้เชื่อมต่อ (วินาที)	ค่าเฉลี่ยอัตราความเร็ว
13	Database	6/8/2023 20:28	6/8/2023 20:41	13.27	819 ข้อมูล ต่อ วินาที
14	Database	6/8/2023 20:29	6/8/2023 20:43	14.45	526 ข้อมูล ต่อ วินาที
15	Database	7/8/2023 17:39	7/8/2023 17:52	13.34	589 ข้อมูล ต่อ วินาที

จากตางรางที่ 4.15 จากผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย 15 ลูกข่ายก็ประสบผลสำเร็จไม่เกิดปัญหาใด ๆ ด้วยจากที่เวลาที่เริ่มทำงานกับหลังจากเพิ่มประวัติบันทึกลงไปยังเว็บแอพพลิเคชัน ในส่วนของประเภทฐานข้อมูลโดยมีจำนวนข้อมูลที่บันทึกให้กับทางฐานข้อมูลจากฝั่งของแม่ข่ายอยู่ที่ 10,872, 7,611, และ 7862 ข้อมูล ได้ค่าเฉลี่ยในการบันทึกได้ 13.68 วินาที

ตารางที่ 4.16 การทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่ายประเภทฐานข้อมูล เริ่มงานทำงานครั้งแรกและสิ้นสุดครั้งนั้นเท่าไหร่ โดยจะทดลองจำนวน 3 ลูกข่ายถัดมาที่ขนาดข้อมูลไม่เท่ากันคือ MySQL จำนวนข้อมูล 10,872, Oracle Database จำนวนข้อมูล 7,611 กับ 7,862 แต่เท่ากับลูกข่ายก่อนหน้า 3 ลูกข่าย และยังแตกต่างจาก 15 ลูกข่ายก่อนหน้า ผลลัพธ์ที่ได้คือสามารถเชื่อมต่อผ่านหรือไม่รวมถึงใช้เวลาในการเชื่อมต่อกับแม่ข่ายเร็วหรือนานเท่าไหร่

ครั้งที่	ประเภท	เวลาเริ่มทำงาน	เวลาที่หยุดเชื่อมต่อ	เวลาที่ใช้เชื่อมต่อ (วินาที)	ค่าเฉลี่ยอัตราความเร็ว
16	Database	9/8/2023 16:01	9/8/2023 16:19	11.18	819 ข้อมูล ต่อ วินาที
17	Database	9/8/2023 16:02	9/8/2023 16:27	24.57	526 ข้อมูล ต่อ วินาที
18	Database	9/8/2023 16:07	9/8/2023 16:17	10.33	589 ข้อมูล ต่อ วินาที

จากตางรางที่ 4.16 จากผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย 18 ลูกข่ายก็ประสบผลสำเร็จไม่เกิดปัญหาใด ๆ ด้วยจากที่เวลาที่เริ่มทำงานกับหลังจากเพิ่มประวัติบันทึกลงไปยังเว็บแอพพลิเคชัน ในส่วนของประเภทฐานข้อมูลคอมพิวเตอร์โดยมีจำนวนข้อมูลที่บันทึกให้กับทางฐานข้อมูลจากฝั่งของแม่ข่ายอยู่ที่ 10,872, 7,611, และ 7862 ข้อมูล ได้ค่าเฉลี่ยในการบันทึกได้ 15.36 วินาที

ตารางที่ 4.17 การทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่ายประเภทข้อมูลจราจร เริ่มงานทำงานครั้งแรกและสิ้นสุดครั้งนั้นเท่าไหร่ โดยจะทดลองจำนวน 2 ลูกข่ายถัดมาที่ขนาดข้อความเท่ากันคือ 180 Megabytes แต่แตกต่างจาก 18 ลูกข่ายก่อนหน้า ผลลัพธ์ที่ได้คือสามารถเชื่อมต่อผ่านหรือไม่รวมถึงใช้เวลาในการเชื่อมต่อกับแม่ข่ายเร็วหรือนานเท่าไหร่

ครั้งที่	ประเภท	เวลาเริ่มทำงาน	เวลาที่หยุดเชื่อมต่อ	เวลาที่ใช้เชื่อมต่อ (วินาที)	ค่าเฉลี่ยอัตราความเร็ว
19	Sniffer	7/8/2023 17:39	7/8/2023 18:04	24.98	7 Megabytes ต่อ วินาที
20	Sniffer	9/8/2023 16:33	9/8/2023 17:00	26.58	6 Megabytes ต่อ วินาที

จากตางรางที่ 4.17 จากผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย 20 ลูกข่ายก็ประสบผลสำเร็จไม่เกิดปัญหาใด ๆ ด้วยจากที่เวลาที่เริ่มทำงานกับหลังจากเพิ่มประวัติบันทึกลงไปยังเว็บแอพพลิเคชัน ในส่วนของประเภทข้อมูลจราจร โดยมีสร้างไฟล์และส่งไฟล์ให้กับทางเครื่องแม่ข่ายอยู่ที่ 180 Megabytes ได้ค่าเฉลี่ยในการบันทึกได้ 25.78 วินาที

4.6 สรุปผลการทดลอง และปัญหาที่เกิดขึ้น

ในหัวข้อนี้จะกล่าวถึงการทดลองการหาประสิทธิภาพของการทำงานของระบบ การเปรียบเทียบความแตกต่างในการทำงานของระบบและปัญหาที่เกิดขึ้น ซึ่งผลการทดลองที่ 1 ผลการทดลองตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ภายในเน็ตเวิร์คเดียวกันในแต่ละความเร็วของสายสัญญา ได้ผลลัพธ์ออกมาคือครบถ้วน เว้นแต่ลูกข่ายประเภทตัววิเคราะห์ข้อมูลแบบสร้างไฟล์และส่งไฟล์จะนำหน้าแม่ข่ายอยู่ 1 ก้าวเสมอ ผลการทดลองที่ 2 ผลการทดลองตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานของแต่ละฟังก์ชันในแต่ละส่วนของระบบแม่ข่ายและลูกข่าย ได้ผลลัพธ์ออกใช้เวลาน้อยสำหรับระบบที่ให้เป็นการให้บริการเว้นแต่ รูปแบบส่งไฟล์จะไวหรือนานขึ้นอยู่กับขนาดของไฟล์นั้นๆ ผลการทดลองที่ 3 ผลการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องในหน่วยของ ซีพียู, แรม, และการอ่านและการเขียนดิสก์ ผลลัพธ์ออกมาใช้น้อยกว่าที่คาดคิดไว้แต่

เรื่องของการใช้ทรัพยากรการอ่านการเขียนดิสก์ก็คือว่ากลาง ๆ ไม่มากไปน้อยไป ผลการทดลองที่ 4 ผลการทดลองโดยการรับจำนวนโหลดของเครื่องลูกข่าย สามารถรองรับได้ ผลลัพธ์ออกมาถือว่าสามารถใช้งานได้เป็นกลาง เนื่องจากไม่สามารถทดลองจำนวนลูกข่ายได้มากกว่านั้นเพราะเซิร์ฟเวอร์ทรัพยากรได้เกือบ 100% และถ้าเกิดถึง 100% อาจจะทำให้อันตรายเซิร์ฟเวอร์ที่ใช้ทดสอบ และนอกจากหัวข้อนี้ได้กล่าวถึงปัญหาที่เกิดขึ้นดังตาราง 4.14

ตารางที่ 4.18 สรุปปัญหาที่เกิดขึ้นกรณีทดลองที่ได้ทดลองมาจากทั้งหมด 4 หัวข้อทดลอง โดยจะแบ่งเป็นประเภท ๆ ดังนี้

ประเภท	ปัญหาที่เกิดขึ้น
Listen	เมื่อมีการใช้งานฟังก์ชันทดสอบของทรัพยากรกับแม่ข่ายและลูกข่ายจำนวน อาจทำให้เกิดการเชื่อมต่อกับฐานข้อมูล (Pool time out) ได้เป็นระยะ
File	ขนาดของไฟล์ที่ส่งนั้น (memory allocation of ...) เกินการที่มีอยู่ของทรัพยากร แรม (RAM) ดังนั้นถ้าไฟล์ขนาดใหญ่เกินกว่าจำนวน RAM อาจจะทำให้ระบบลูกข่ายประเภทนี้หยุดทำงานไป
Sniffer FTP	เมื่อมีลูกข่ายทำงานอยู่เยอะอาจจะรอคิวในการส่งไฟล์นาน หรือ อาจจะหลุด connect ไป แต่ยังสามารถสร้างไฟล์ของ sniffer เองได้ (ยังเป็นปัญหาที่ยังไม่แน่ใจ)
ทั้งแม่ข่ายและลูกข่ายทุกประเภท	เมื่อทำงานได้ถึง 15 หรือ 16 ตัว ทำงานไปได้สัก 2-3 ชั่วโมง จะเกิด PoolTimeOut หรือไม่ก็ ConnectTime Out และไม่แน่ใจ อาจเพราะใช้งาน script ตั้งแต่ตัวที่ 11 - 16 คือทำงานบนเครื่อง เดียวกัน และทรัพยากรที่ให้ไปไม่พอใช้งาน แต่สามารถกลับรีการทำงานใหม่อีกครั้งได้ โดยทรัพยกรที่ใช้คือ 2 CPUs, 4 RAM, 100GB Disk และดูเหมือนว่าจะพอใช้งานกับ script 2 - 3 ลูกข่ายต่อ 1 เครื่อง เนื่องจาก script ไม่ได้ใช้ทรัพยกรมาก แต่ library หรือ package ที่ script ที่ดึงมาใช้งานใช้งานมากสมควร เนื่องจาก ไม่ได้เตรียม script ในการ restart บริการทำงานใหม่อีกครั้ง เมื่อ script หยุดรัน โดยเปลี่ยนการทดสอบเป็น 2 CPUs, 4 RAM, 100GB Disk เป็น 2 เครื่อง

จากตารางที่ 4.18 ก็จะแสดงถึงปัญหาที่เกิดขึ้นมาในการทำงานแม่ข่ายและลูกข่ายแต่ก็ไม่ได้เป็นปัญหาใหญ่ที่จะให้เกิดแต่อย่างใด เพียงแต่เป็นปัญหาที่จุกจิกละต้องใช้ผู้ดูแลระบบเข้าจัดการเพื่อให้ระบบกลับมาทำงานปกติอีกครั้งหรือจะปล่อยผ่านไป

บทที่ 5
สรุปผลการวิจัย

ระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล ได้นำปัญหาและช่องว่างต่างๆ ของการทำระบบให้ตรงกับกฎหมายข้อมูลจราจรคอมพิวเตอร์และกฎหมายข้อมูลส่วนบุคคล ทั้งเรื่องปัญหาระบบหลักไม่สามารถเข้าถึงหรือรองรับได้ทางระบบจะช่วยเข้ามาเป็นอีกหนึ่งตัวเลือกหรือหนึ่งส่วนเสริม ที่สามารถให้ลดภาระของผู้ดูแลระบบหรือผู้ใช้งานระบบ ซึ่งสามารถสรุปผลการทดลองและอภิปรายผลการทดลองได้ดังนี้

5.1 สรุปและอภิปรายผล

จากการทดลองงานวิจัยเรื่อง ระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล สามารถสรุปและอธิปรายผลการทดลองทั้งหมดได้ดังนี้

5.1.1 การตรวจสอบความครบถ้วนตามข้อตกลงของทางการนำไปใช้ในการตรวจสอบผลลัพธ์ของลูกข่ายประเภทครวจสอบข้อมูลจราจรคอมพิวเตอร์, ฐานข้อมูล, ข้อมูลจราจรว่าแสดงผลถูกต้องและครบถ้วนตามข้อกำหนดหรือข้อตกลงของการนำไปใช้งานจริงให้กับทางหน่วยงานหน่วยงานหนึ่ง โดยผลลัพธ์ของการตรวจการสอบนั้นจะคล้ายกับการทำแสดงตัวอย่างของการผลลัพธ์ที่มองเห็นได้แน่ชัดจึงถูกนำเสนอโดยเว็บแอพพลิเคชัน ในส่วนแสดงผลข้อมูลที่เกี่ยวข้องกับระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล เนื่องจากระบบที่ผู้วิจัยได้พัฒนาไม่ได้มีการแสดงผลให้เห็นได้อย่างชัดเจน และได้ทำขึ้นเพื่อทำงานอยู่ด้านหลังระบบอีกต่อหนึ่งนั้นเอง

5.1.2 การทดลองตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ภายในเน็ตเวิร์คเดียวกันในแต่ละความเร็วของสายสัญญา นั้นเตรียมทดสอบในรูปแบบของผ่านเน็ตเวิร์คสวิตช์โดยเกณฑ์คือ ความเร็วหน่วย Mbps (Megabyte per seconds) ผลสรุปคือได้ครบจำนวนของข้อความที่ลูกข่ายส่งและแม่ข่ายรับเนื่องจากสามารถส่งและรับครบในความเร็วที่ต่ำที่สุดคือ 100 Mbps (Megabyte per seconds) ตารางที่ 5.1

ตารางที่ 5.1 ผลสรุปจากการทดลองตรวจสอบดูผลลัพธ์ของการส่งข้อมูลผ่านอุปกรณ์เน็ตเวิร์คสวิตช์ดังนี้

ลูกข่ายประเภท	ความครบถ้วน (%)	
	100 Mbps	1000 Mbps
ตรวจสอบข้อมูลจราจรคอมพิวเตอร์	100	100
สำรองไฟล์	100	100
ฐานข้อมูล	100	100
ข้อมูลจราจร (สร้างไฟล์ส่งไฟล์)	100	100
ข้อมูลจราจร (ส่งไปยังเซิร์ฟเวอร์จราจรคอมพิวเตอร์)	100	100

ดังนั้นจากตาราง 5.1 เมื่อนำค่าความครบถ้วนมาเฉลี่ยรวมได้ 100% ไม่ว่าความเร็วจะเร็วกว่าที่ผู้วิจัยได้กำหนดไว้ต่ำก็จะไม่มีผล

5.1.3 การทดลองตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานของแต่ละฟังก์ชันในแต่ละส่วนของระบบแม่ข่ายและลูกข่าย นั้นเตรียมทดสอบผ่านฟังก์ชันที่เขียนขึ้นมาในโค้ดของระบบโดยเกณฑ์คือ หน่วยเป็นวินาทีและทศนิยม 4 หลักและตรวจสอบผลลัพธ์ด้วยการดูเวลาในการทำงานต่อหนึ่งครั้ง ผลสรุปดังตารางที่ 5.2

ตารางที่ 5.2 ผลสรุปจาการทดลองตรวจสอบดูผลลัพธ์ของเวลาที่ใช้งานของแต่ละฟังก์ชันในแต่ละส่วนดังนี้

ประเภท	ค่าเฉลี่ย (มิลลิวินาที)
ตรวจสอบข้อมูลจราจรคอมพิวเตอร์	2.81
ส่งไฟล์	20,718.45
ฐานข้อมูล	281.29
ข้อมูลจราจร 2 รูปแบบ	7,452.20

จากตารางที่ 5.2 ลูกข่ายประเภทตรวจสอบข้อมูลจราจรคอมพิวเตอร์ ได้เฉลี่ย 2.81 มิลลิวินาที หรือ 0.0028 วินาที, ลูกข่ายประเภทส่งไฟล์ ได้เฉลี่ย 20,718.45 มิลลิวินาที หรือ 20.7184 วินาที ลูกข่ายประเทฐานข้อมูล ได้ 281.29 มิลลิวินาที หรือ 0.2812 วินาที, และสุดท้ายลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์ทั้ง 2 รูปแบบคือ รูปแบบสร้างไฟล์ส่งไฟล์ และรูปแบบส่งข้อความไปยังเซิร์ฟเวอร์ข้อมูลจราจรคอมพิวเตอร์ ได้เฉลี่ย 7,452.20 มิลลิวินาที หรือ 7.4522 วินาที เมื่อนำค่าเฉลี่ยของแต่ละประเภทรวมกันละหาค่าเฉลี่ยใหม่จะได้ 7113.68 มิลลิวินาที หรือ 7.1136 วินาที เพิ่มเติมจากแม่ข่ายและลูกข่ายทั้งหมดที่ได้ทดลองเวลาที่ใช้น้อยที่สุดคือ แม่ข่ายทุกประเภทและลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์ รองลงมาคือลูกข่ายประเภทฐานข้อมูล และกลางคือลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์ทั้ง 2 รูปแบบ สุดท้ายคือลูกข่ายประเภทส่งไฟล์ใช้เวลานานที่สุด เนื่องจากปัจจัยอยู่ที่ขนาดของไฟล์ที่ส่งไปยังเซิร์ฟเวอร์เอฟทีพี (FTP Server) โดยผู้วิจัยได้ใช้งานไฟล์ขนาด 579 Megabytes

5.1.4 การทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องในหน่วยของ ซีพียู, แรม, และการอ่านการเขียนดิสก์ นั้นเตรียมการทดสอบผ่านฟังก์ชันที่เขียนขึ้นมาในโค้ดของระบบเช่นกันโดยเกณฑ์คือ หน่วยเป็นเปอร์เซ็นต์ และหน่วยเอ็มบี หรือ เมกะไบต์ (MB หรือ Megabytes) ผลสรุปดังตารางที่ 5.3

ตารางที่ 5.3 ผลสรุปจาการทดลองประสิทธิภาพของระบบในการใช้งานทรัพยากรเครื่องในหน่วยของ ซีพียู, แรม. และการอ่านการเขียนดิสก์ดังนี้

ประเภท	ค่าเฉลี่ย			
	ซีพียู (CPU) (%)	แรม (RAM) (MB)	การอ่าน (MB/s)	การเขียนดิสก์ (MB/s)
แม่ข่าย	3.1	12.33	679.83	134.20
ลูกข่ายตรวจสอบข้อมูลจราจรคอมพิวเตอร์	3.6	3.23	1599.78	148.00
ลูกข่ายส่งไฟล์	2.9	0.10	1396.58	158.65
ลูกข่ายฐานข้อมูล	5.7	1.06	1167.84	1573.04
ลูกข่ายข้อมูลจราจร (สร้างไฟล์ส่งไฟล์)	5.1	0.26	240.21	18.64
ลูกข่ายข้อมูลจราจร (ส่งไปเซิร์ฟเวอร์จราจรคอมพิวเตอร์)	4.2	0.10	596.96	161.61

จากตารางที่ 5.3 แม่ข่ายใช้งานทรัพยากร ซีพียู (CPU) เฉลี่ยได้ 3.1%, แรม (RAM) เฉลี่ยได้ 12.33 Megabytes และการอ่านของดิสก์ได้เฉลี่ย 679.83 Megabytes per second การเขียนของดิสก์ 134.20 Megabytes per second, ต่อมาลูกข่ายประเภทข้อมูลจราจรคอมพิวเตอร์ใช้งานทรัพยากร ซีพียู (CPU) เฉลี่ยได้ 3.6%, แรม (RAM) เฉลี่ยได้ 3.23 Megabytes, และการอ่านดิสก์ของดิสก์ได้เฉลี่ย 1599.78 Megabytes per second การเขียนของดิสก์ 148.00 Megabytes per second ต่อมาลูกข่ายประเภทส่งไฟล์ใช้งานทรัพยากร ซีพียู (CPU) เฉลี่ยได้ 2.9%, แรม (RAM) เฉลี่ยได้ 0.10 Megabytes และการอ่านของดิสก์ได้เฉลี่ย 1396.58 Megabytes per second การเขียนของดิสก์ 158.65 Megabytes per second ต่อมาลูกข่ายประเภทฐานข้อมูลใช้ทรัพยากร ซีพียู (CPU) เฉลี่ยได้ 5.7%, แรม (RAM) 0.90 Megabytes per second และการอ่านของดิสก์ได้เฉลี่ย 1,167.84 Megabytes การเขียนของดิสก์ 1573.04 Megabytes per second, ต่อมาลูกข่ายประเภทตัววิเคราะห์โปรโตคอลรูปแบบสร้างไฟล์และส่งไฟล์ใช้งานทรัพยากร ซีพียู (CPU) ได้เฉลี่ย 5.1%, แรม (RAM) 0.26 Megabytes และการอ่านของดิสก์ได้เฉลี่ย 240.21 Megabytes per second การเขียนของดิสก์ 18.64 Megabytes per second, สุดท้ายลูกข่ายประเภทตัววิเคราะห์โปรโตคอลรูปแบบส่งข้อความไปยังเซิร์ฟเวอร์ข้อมูลจราจรคอมพิวเตอร์ใช้งานทรัพยากร ซีพียู (CPU) 4.2%, แรม (RAM) 0.10 Megabytes และการอ่านของดิสก์ได้เฉลี่ย 596.96 Megabytes per second การเขียนของดิสก์ 161.61 Megabytes per second จากที่ได้อธิบายมาทั้งหมดได้ค่าเฉลี่ยของทรัพยากรน้อยสำหรับการทำงานระบบบริการแต่ค่าของการอ่านการเขียนของดิสก์เป็นค่ากลางไม่มากเกินไปและไม่น้อยเกินไป

5.1.5 ผลการทดลองรองรับจำนวนโหลดของเครื่องลูกข่ายและความเร็วในเสร็จสิ้นการทำงานต่อ 1 ครั้ง นั้นเตรียมทดสอบผ่านการใช้นาฬิกานับเวลาในการเชื่อมต่อของเครื่องลูกข่ายและใช้เวลาในการเชื่อมต่อกับเครื่องแม่ข่ายเท่าไหร่ โดยเกณฑ์คือ หน่วยเป็นวินาทีและมีเตรียมการทดสอบลูกข่ายจำนวน 20 ลูกข่ายบนเซิร์ฟเวอร์ของห้องปฏิบัติการโดยจะใช้กำหนดทรัพยากรของเครื่องลูกข่ายคือ ระบบปฏิบัติการ (OS) คือ “Oracle Linux x86

v.8u7”, ซีพียู (CPU) คือ 1 หน่วย (Core), แรม (RAM) คือ 2 Gigabytes, และจำนวนฮาร์ทดิสก์ (Disk) คือ 35 Gigabytes แบ่งออกเป็นอีก 2 เครื่องโดยที่เครื่องแรกทำงาน 6 เครื่องประเภทข้อมูลจราจรคอมพิวเตอร์ ต่อด้วยลูกข่ายประเภทส่งไฟล์ 6 เครื่อง และลูกข่ายประเภทข้อมูลฐานข้อมูล 6 เครื่อง สุดท้ายตัววิเคราะห์โปรโตคอล 2 เครื่อง รวมทั้งหมดเป็น 20 เครื่องลูกข่าย ผลสรุปดังตารางที่ 5.4

ประเภท	**ค่าเฉลี่ย (วินาที)**
ลูกข่ายตรวจสอบข้อมูลจราจรคอมพิวเตอร์	3.69
ลูกข่ายสำรองไฟล์	14.69
ลูกข่ายฐานข้อมูล	14.52
ลูกข่ายข้อมูลจราจรทั้ง 2 รูปแบบ	25.78

จากตารางที่ 5.4 ผลลัพธ์ค่าเฉลี่ยของเครื่องลูกข่ายทุกประเภทนั้นผลเฉลี่ย คือ 12.44 วินาที จากทั้งหมด 20 ลูกข่าย โดยจากการทดลอง เมื่อตรวจสอบผลลัพธ์ขณะระบบได้เริ่มทำงานนั้น ช่วงแรกไม่มีเจอปัญหาอะไร สามารถเชื่อมต่อได้ปกติ แต่เมื่อเวลาผ่านไป 3 ถึง 4 ชั่วโมงลูกข่ายที่ 19 คือลูกข่ายประเภทตัววิเคราะห์โปรโตคอลรูปแบบสร้างไฟล์ส่งไฟล์หยุดทำงานไปแต่ยังสามารถกลับไปเรียกทำงานใหม่อีกครั้งได้ไม่มีปัญหาอะไรแต่เมื่อเวลาผ่านไปอีกสักระยะก็จะหยุดไปอีกครั้ง ต่อมาเพิ่มลูกข่ายไปอีก 1 ลูกข่ายก็ยังสามารถเชื่อมต่อได้ปกติเป็น 20 ลูกข่ายจากทั้งหมดสรุปเชื่อมต่อหรือรองรับได้ทั้งหมด 20 แต่ก็จะมีจุดสังเกตเมื่อมีการทำงานลูกข่ายใน 1 เครื่องจำนวนหลาย ๆ ลูกข่ายอาจทำให้เกิดปัญหาดังรูปภาพด้านล่าง 5.1 - 5.3 ขึ้นได้

```
thread 'main' panicked at 'Failed to connect to server: Os { code: 110, kind: TimedOut,
 message: "Connection timed out" }', src/module/handles.rs:224:109
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
^Cmake: *** [Makefile:23: run] Interrupt
```

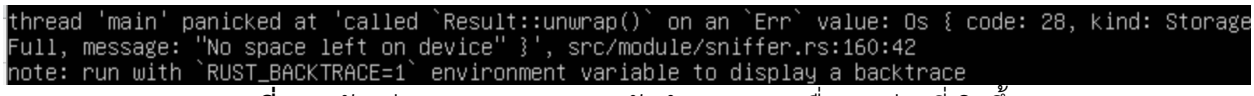
ภาพที่ 5.1 ตัวอย่างของผลการทดลองรับจำนวนของเครื่องลูกข่ายที่เกิดปัญหาขึ้น 1

จากภาพที่ 5.1 กรณีนี้เกิดขึ้นได้จากทรัพยากรเครื่องที่ใช้งานไม่เพียงพอต่อการทำงานจำนวนหลาย ๆ ลูกข่ายตัวระบบปฏิบัติการเกิดการค้างขึ้นในช่วงที่ลูกข่ายพยายามเชื่อมต่อกับแม่ข่าย

```
thread 'main' panicked at 'called `Result::unwrap()` on an `Err` value: PoolTimedOut',
src/module/log0.rs:64:20
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
^Cmake: *** [Makefile:23: run] Interrupt
```

ภาพที่ 5.2 ตัวอย่างของผลการทดลองรับจำนวนของเครื่องลูกข่ายที่เกิดปัญหาขึ้น 2

จากภาพที่ 5.2 กรณีต่อมาเกิดขึ้นคล้ายครึงกับกรณีแรกคือทำให้การระบบลูกข่ายบนตัวเครื่องเชื่อมต่อกับฐานข้อมูลหลักไม่ได้จึงหยุดทำงานไป



ภาพที่ 5.3 ตัวอย่างของผลการทดลองรับจำนวนของเครื่องลูกข่ายที่เกิดขึ้น 3

จากภาพที่ 5.3 กรณีนี้จะเป็นจำนวนความจุของเครื่องไม่เหลือจึงหยุดทำงานไปเนื่องจากทำงานจำนวนลูกข่ายเยอะเช่นกัน

โดยผลการทดลองทั้ง 4 การทดลองสามารถบอกได้ว่าระบบมีความสามารถในการจัดการเรื่องของการตรวจสอบข้อมูลและนำเข้าข้อมูลที่ได้มาไปตรวจสอบต่ออีกทอดหนึ่งให้กับระบบหลังบ้านส่วนอื่นเพิ่มเติมของเว็บแอพพลิเคชัน เพื่อให้มีความน่าเชื่อถือมากขึ้นในการยืนยันให้กับกฎหมายข้อมูลจราจรคอมพิวเตอร์และกฎหมายข้อมูลส่วนบุคคล

5.2 ข้อเสนอแนะในการพัฒนา

ระบบจัดการข้อมูลในรูปแบบที่ต้องการตรวจสอบข้อมูลและนำเข้าข้อมูล สามารถเป็นแนวทางประยุกต์และพัฒนาระบบให้สามารถนำไปใช้ประโยชน์ได้หลายอย่างให้มีประสิทธิภาพมากยิ่งขึ้นโดยมีข้อเสนอแนะในการทำวิจัยครั้งต่อไปนี้

5.2.1. เพิ่มความสามารถลูกข่ายในการทำให้ทำงานระบบใหม่อีกครั้งได้เมื่อเกิดมีการหยุดการทำงานไป

5.2.2. เพิ่มการเข้าถึงโปรโตคอลความปลอดภัยอย่างเช่น TLS เพื่อป้องกันการเข้าถึงข้อมูลระหว่างส่งข้อมูลให้กับแม่ข่ายและลูกข่าย

5.2.3. ปรับปรุงหรือปรับแก้ไขปัญหาที่เกิดขึ้นจากทดลอง เพื่อให้ได้ผลลัพธ์ออกมามีประสิทธิภาพดีขึ้น

บรรณานุกรม

Softnix. (2561). Softnix Logger ตัวจัดเก็บ Log ตามพรบ. คอมพิวเตอร์ 2560

แหล่งที่มา https://www.techtalkthai.com/softnix-logger-comply-with-thai-computer-law-2560/

ราชกิจจานุเบกษา. (2564). พระราชบัญญัติคุ้มครองขข้อมูลจราจรคอมพิวเตอร์

แหล่งที่มา https://www.ratchakitcha.soc.go.th/DATA/PDF/2564/E/188/T_0009.PDF

ราชกิจจานุเบกษา. (2562). พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. ๒๕๖๒

แหล่งที่มา https://www.ratchakitcha.soc.go.th/DATA/PDF/2562/A/069/T_0052.pdf

Wikipedia Encyclopedia. (2023). Network socket

แหล่งที่มา https://en.wikipedia.org/wiki/Network_socket

Javatpoint. (2011). Java Socket Programming

แหล่งที่มา https://www.javatpoint.com/socket-programming

Geeksforgeeks Akshat Sinha. (2023). Socket Programming in C/C++

แหล่งที่มา https://www.geeksforgeeks.org/socket-programming-cc/

Cryptographic hash function

แหล่งที่มา https://en.wikipedia.org/wiki/Cryptographic_hash_function

freeCodeCamp Jeff M Lowery. (2020). MD5 vs SHA-1 vs SHA-2 - Which is the Most Secure Encryption Hash and How to Check Them

แหล่งที่มา https://www.freecodecamp.org/news/md5-vs-sha-1-vs-sha-2-which-is-the-most-secure-encryption-hash-and-how-to-check-them/

Wikipedia Encyclopedia. (2023). Sniffer (protocol analyzer)

แหล่งที่มา https://en.wikipedia.org/wiki/Sniffer_(protocol_analyzer)

DTC Internetworking Co.,Ltd. (2022). สรุป PDPA คืออะไร ต้องทำอะไรบ้าง - รับปรึกษาทำ PDPA ครบวงจร - dtci

แหล่งที่มา https://www.dtci.co.th/pdpa

ภาคผนวก

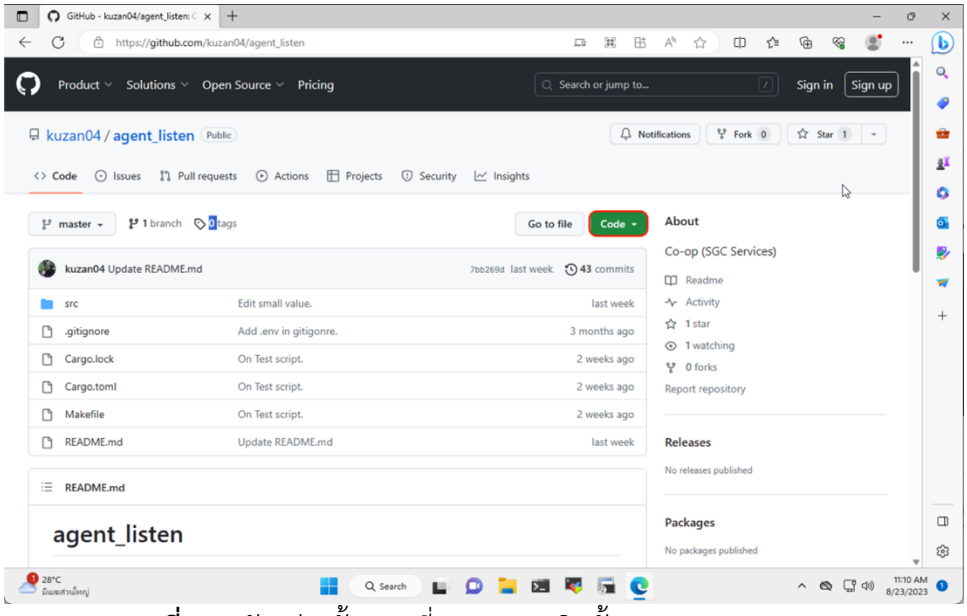
ภาคผนวก ก
คู่มือการใช้งานระบบตรวจสอบและการนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคล

คู่มือการติดตั้ง Agent listen & client เฉพาะ Windows

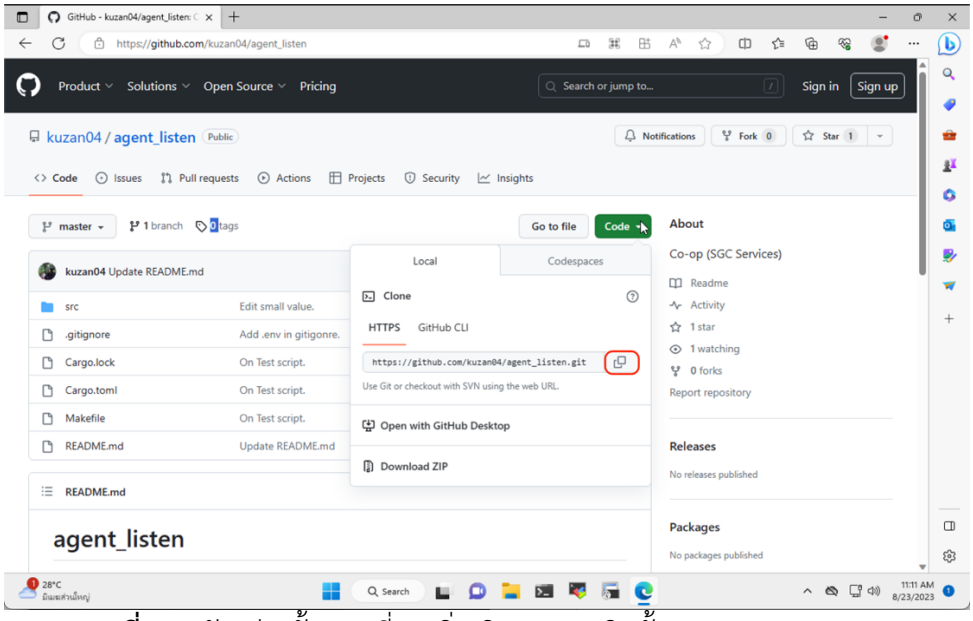
การติดตั้ง Script agent listen

เนื่องจากตอนนี้ยังไม่รองรับ Windows อย่างเต็มรูปแบบ อาจจะมีปัญหาเกิดขึ้นได้เสมอ

1) เปิดเว็บเบราว์เซอร์ชนิดใดก็ได้จากนั้นเข้าไปยังเว็บ “https://github.com/kuzan04/agent_listen” เมื่อเข้าไป คลิกปุ่มสีเขียว “Code” จากนั้น คลิก “สี่เหลี่ยมซ้อนกัน” เพื่อทำการคัดวองข้อความด้านหน้านั้น

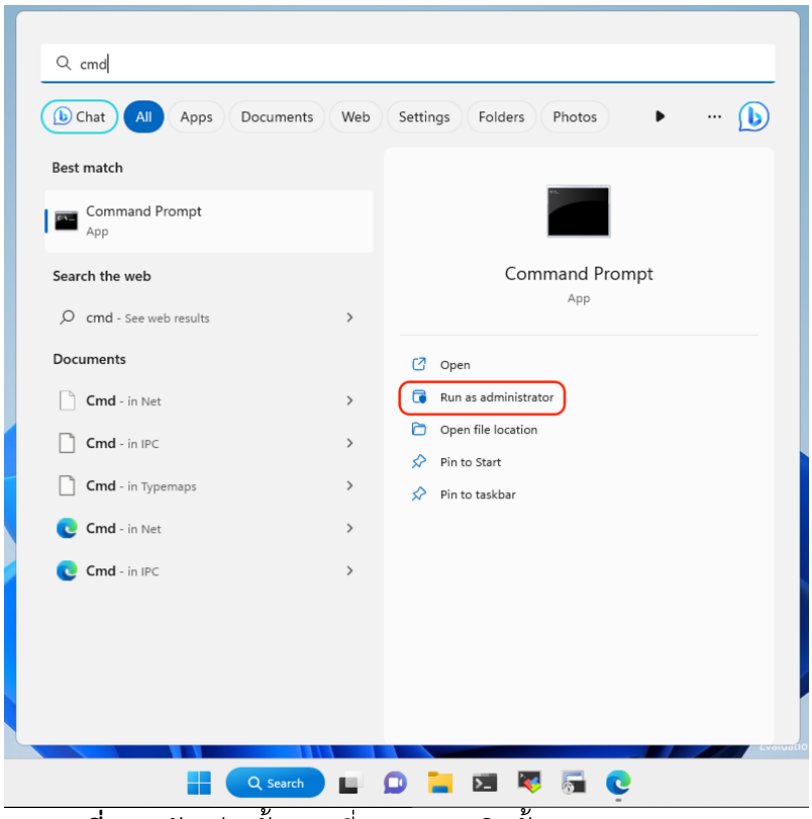


ภาพที่ ก.1 ตัวอย่างขั้นตอนที่ 1 ของการติดตั้ง Script agent listen



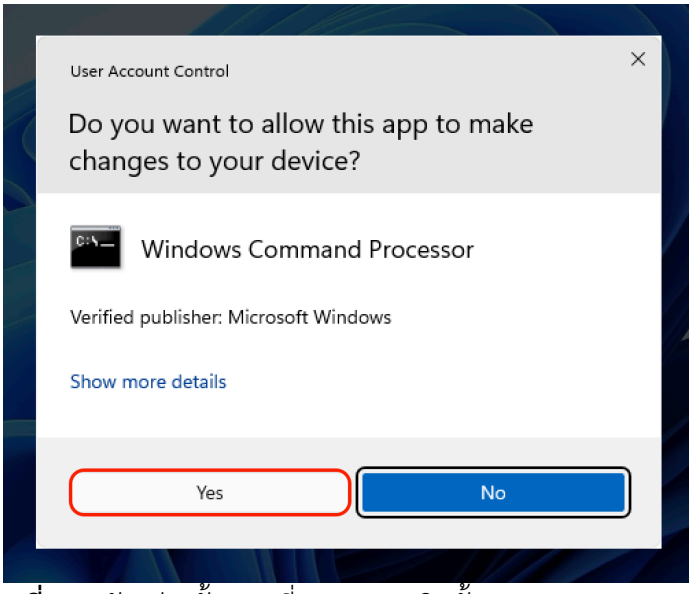
ภาพที่ ก.2 ตัวอย่างขั้นตอนที่ 1 เพิ่มเติมของการติดตั้ง Script agent listen

2) ทำการ คลิก “Search” หรือรูป “แว่นขยาย” จากนั้นพิมพ์ “CMD” เลือก “Command Prompt” โดยให้ คลิก “Run as administrator”



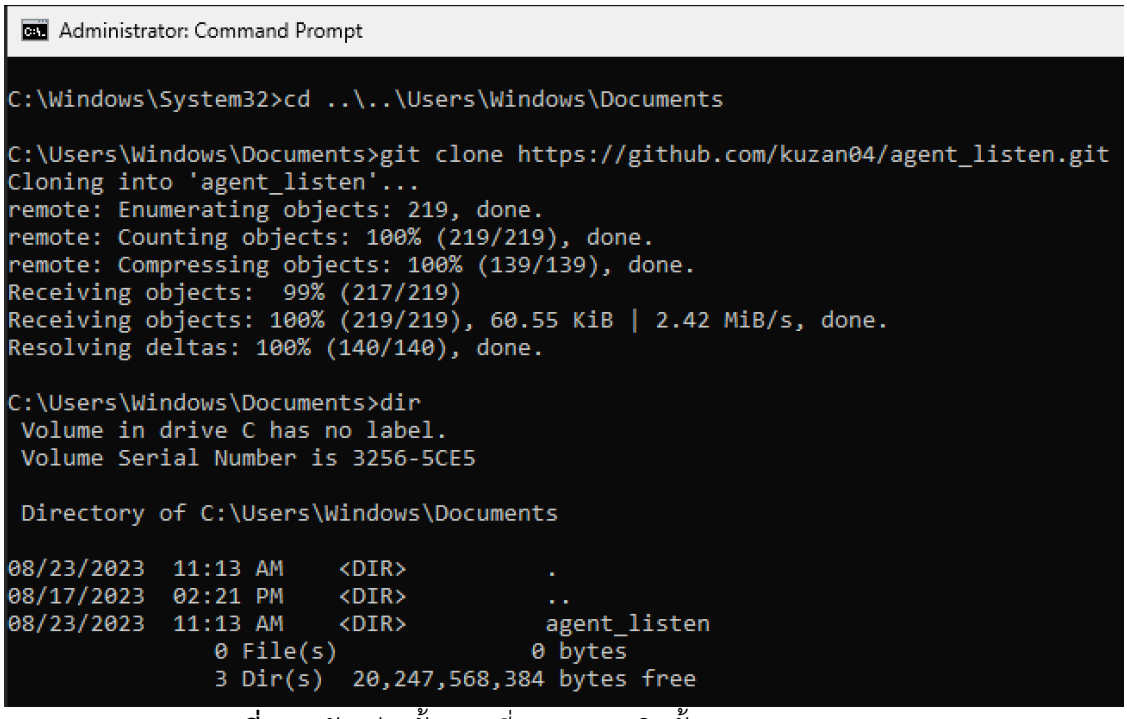
ภาพที่ ก.3 ตัวอย่างขั้นตอนที่ 2 ของการติดตั้ง Script agent listen

3) ในบางครั้งอาจจะมีหน้าต่าง “User Account Control” ให้ทำการ คลิก “Yes”



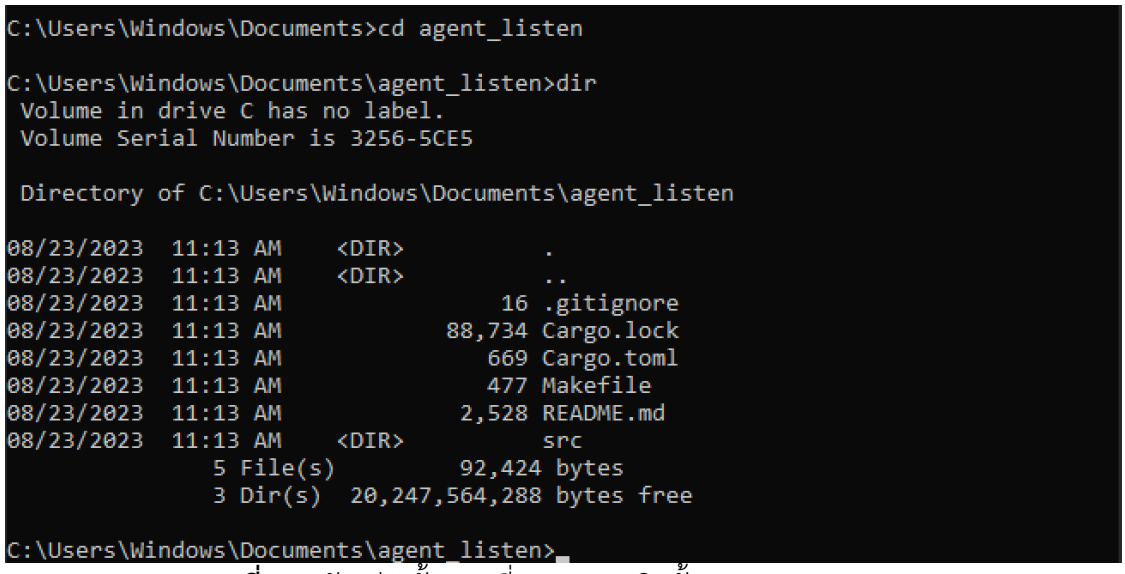
ภาพที่ ก.4 ตัวอย่างขั้นตอนที่ 3 ของการติดตั้ง Script agent listen

4) จากนั้นทำการย้ายที่อยู่ของโฟลเดอร์ “cd ..\..\Users\##User##\Documents” จากนั้นเมื่อย้ายแล้ว พิมพ์ “git clone และข้อความที่ได้จากคัดลอกมาวาง” กดปุ่ม “Enter” ต่อมา พิมพ์ “dir” เพื่อแสดงรายการโฟลเดอร์และไฟล์มีอยู่ขณะตรงนี้



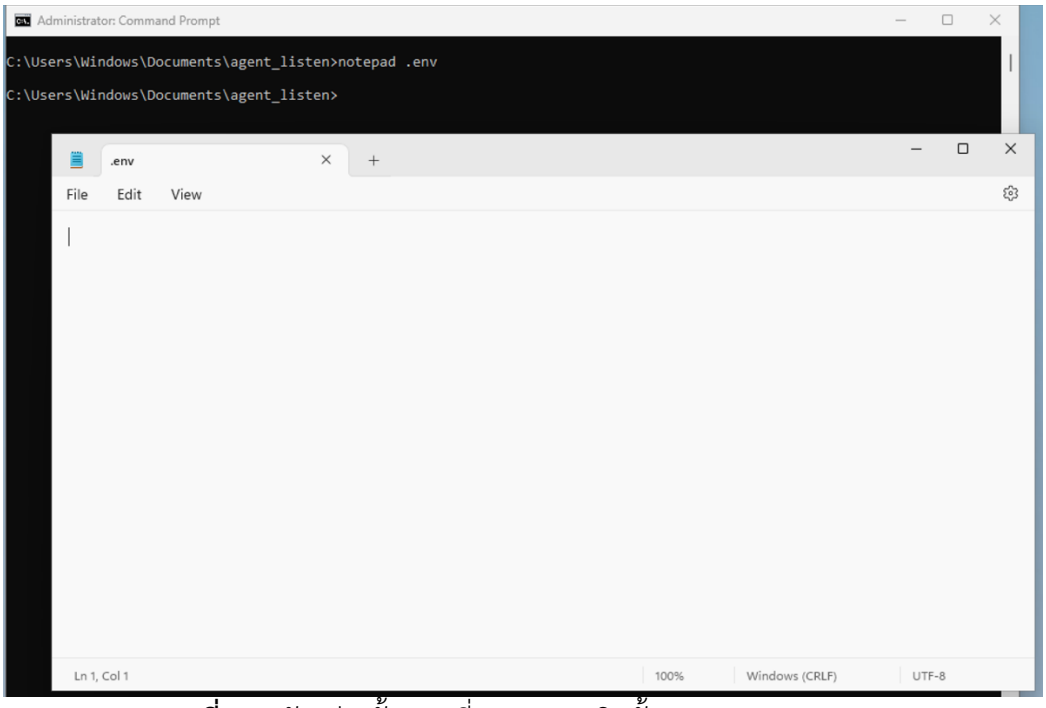
ภาพที่ ก.5 ตัวอย่างขั้นตอนที่ 4 ของการติดตั้ง Script agent listen

5) พิมพ์ “cd agent_listen” เพื่อเข้าไปยังโฟลเดอร์นั้น จากนั้นเพื่อทำการตรวจสอบรายการไฟล์ พิมพ์ “dir”



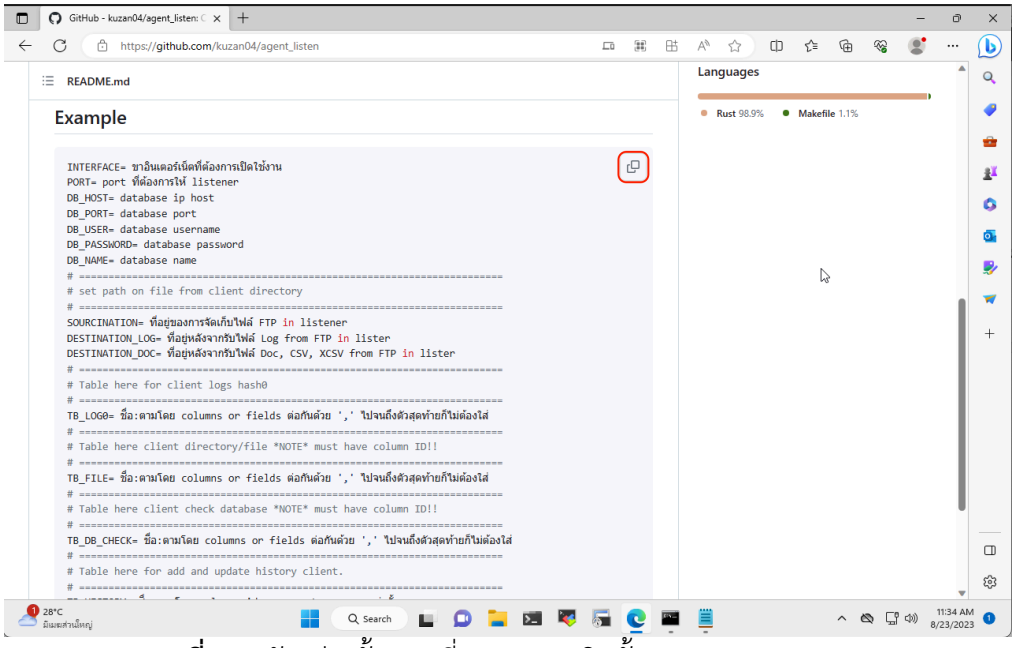
ภาพที่ ก.6 ตัวอย่างขั้นตอนที่ 5 ของการติดตั้ง Script agent listen

6) พิมพ์ “notepad .env” หลังจากนั้นอาจจะมีข้อมูลขึ้นมายืนยันการสร้างไฟล์ ให้ทำการคลิก “Yes” จะได้หน้าต่าง notepad เปล่าๆขึ้นมา



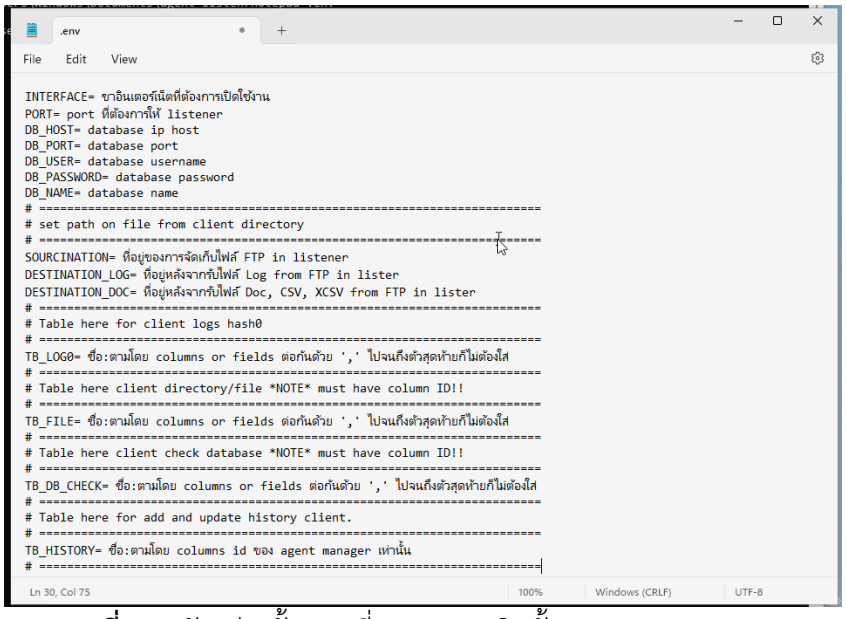
ภาพที่ ก.7 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent listen

7) กลับไปยังเว็บเบราว์เซอร์เพื่อทำการเลื่อนแถบด้านล่างตรงกลาง มุมบนขวาจะมี รูป “สี่เหลี่ยมซ้อนกัน” คลิกเพื่อคัดลอก



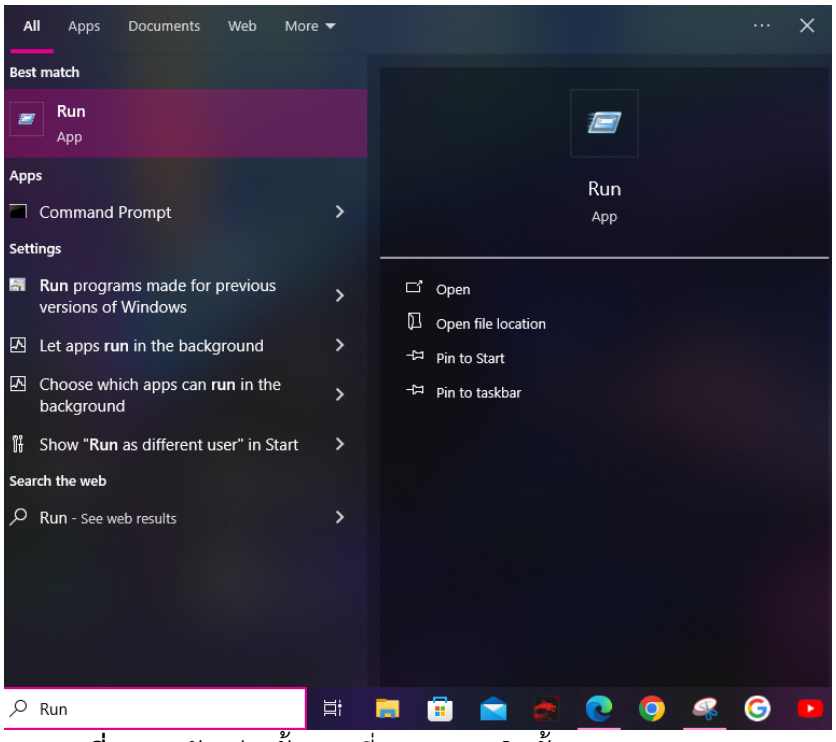
ภาพที่ ก.8 ตัวอย่างขั้นตอนที่ 7 ของการติดตั้ง Script agent listen

8) นำมาวางและทำการแก้ไขข้อมูลด้านหลัง “=” ให้ครบทั้งหมด



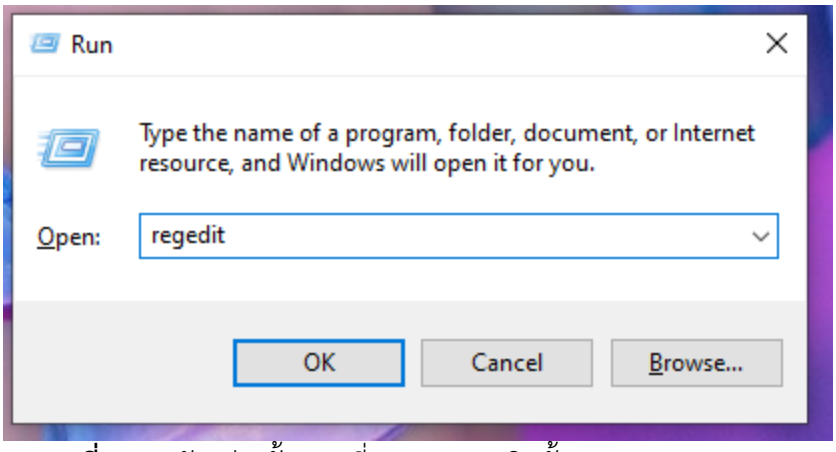
ภาพที่ ก.9 ตัวอย่างขั้นตอนที่ 8 ของการติดตั้ง Script agent listen

9) ทำการกดปุ่ม “Win + R” หรือ คลิก “Search” หรือรูป “แว่นขยาย” จากนั้นพิมพ์ “run” เลือก “Run” คลิกเปิดขึ้นมา



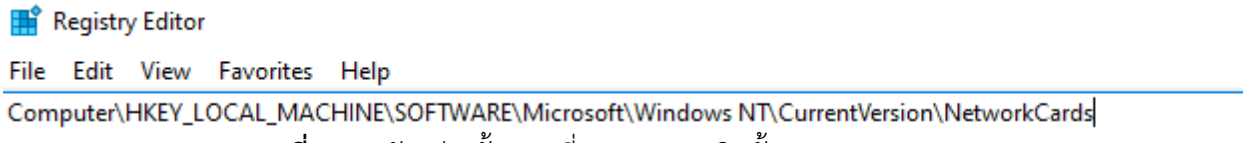
ภาพที่ ก.10 ตัวอย่างขั้นตอนที่ 9 ของการติดตั้ง Script agent listen

10) จากนั้นพิมพ์ หรือคัดลอก วาง “regedit” คลิก “OK”



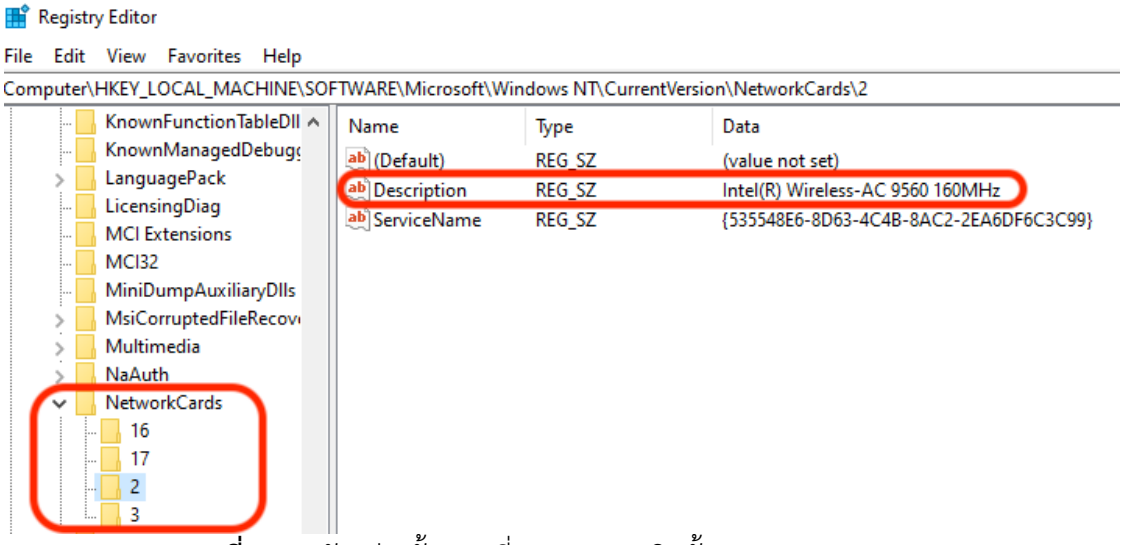
ภาพที่ ก.11 ตัวอย่างขั้นตอนที่ 10 ของการติดตั้ง Script agent listen

11) ให้ทำการพิมพ์ หรือคัดลอก วาง “Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVesion\NetworkCards” เพื่อหา “ServiceName” เพื่อนำไปใช้งาน



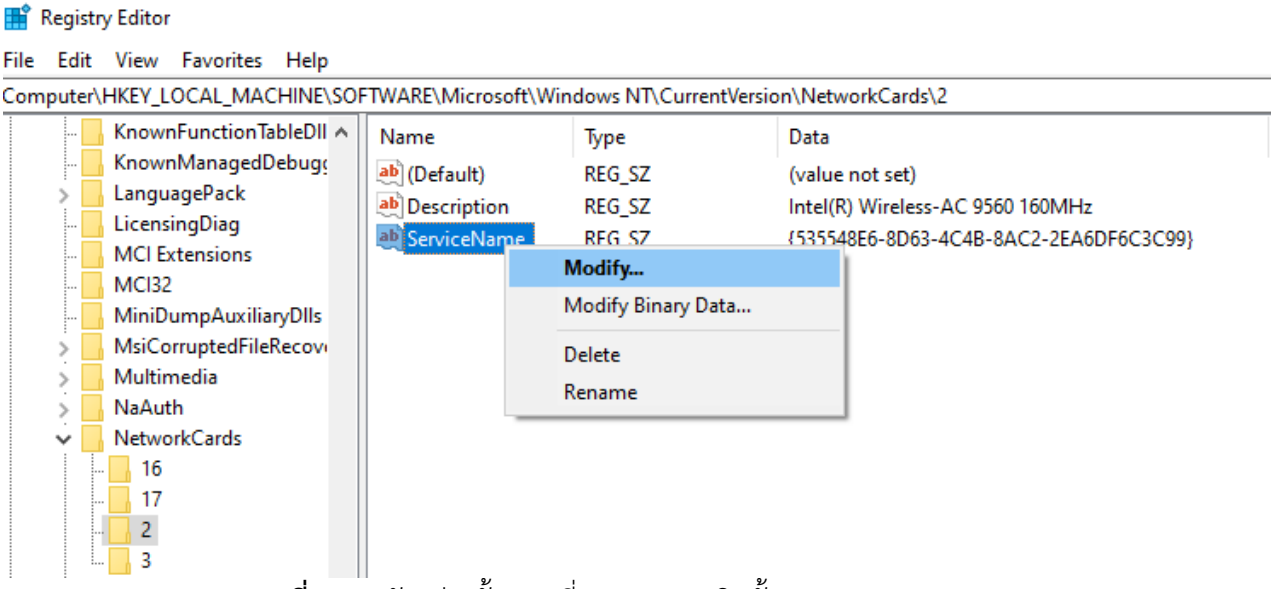
ภาพที่ ก.12 ตัวอย่างขั้นตอนที่ 11 ของการติดตั้ง Script agent listen

12) จากนั้นแถบเมนูซ้ายมือจะมีหมายเลขต่างๆ ให้เข้าไปดูด้านใน และสังเกตดูว่า “Description” อะไรที่ต้องการใช้งานให้กับ script agent listen จากตัวอย่างจะเป็น “Inter(R) Wireless-AC 9560 160MHZ”



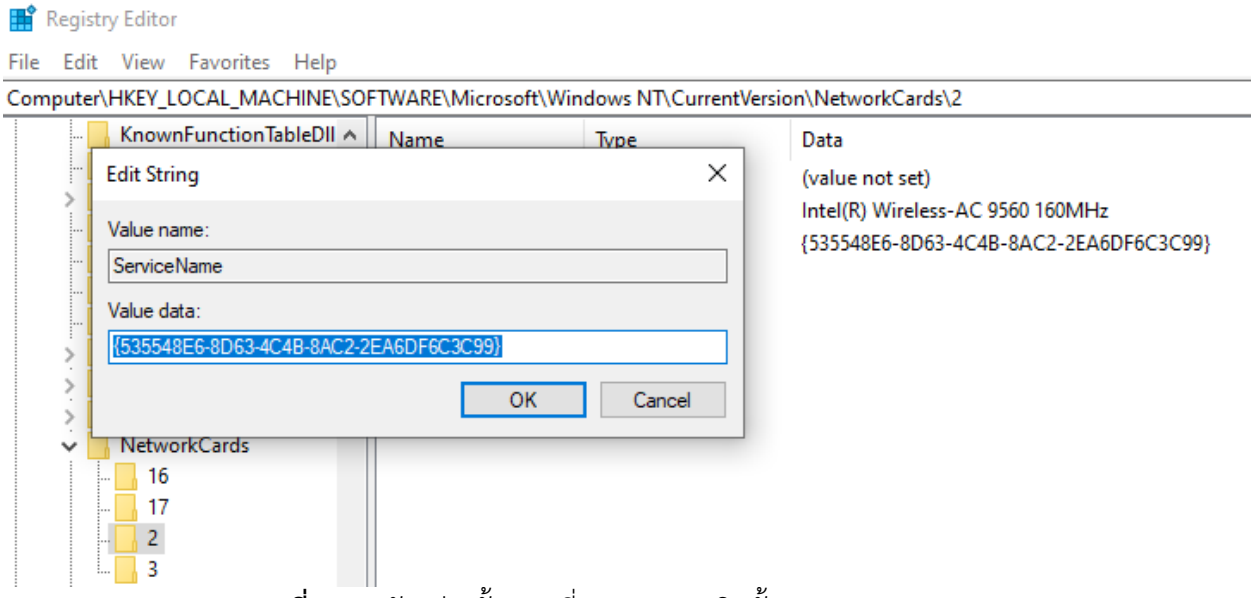
ภาพที่ ก.13 ตัวอย่างขั้นตอนที่ 12 ของการติดตั้ง Script agent listen

13) จากนั้น คลิกขวา “ServiceName” คลิกเลือก “Modify…” เพื่อทำการเข้าไปดูรายละเอียด



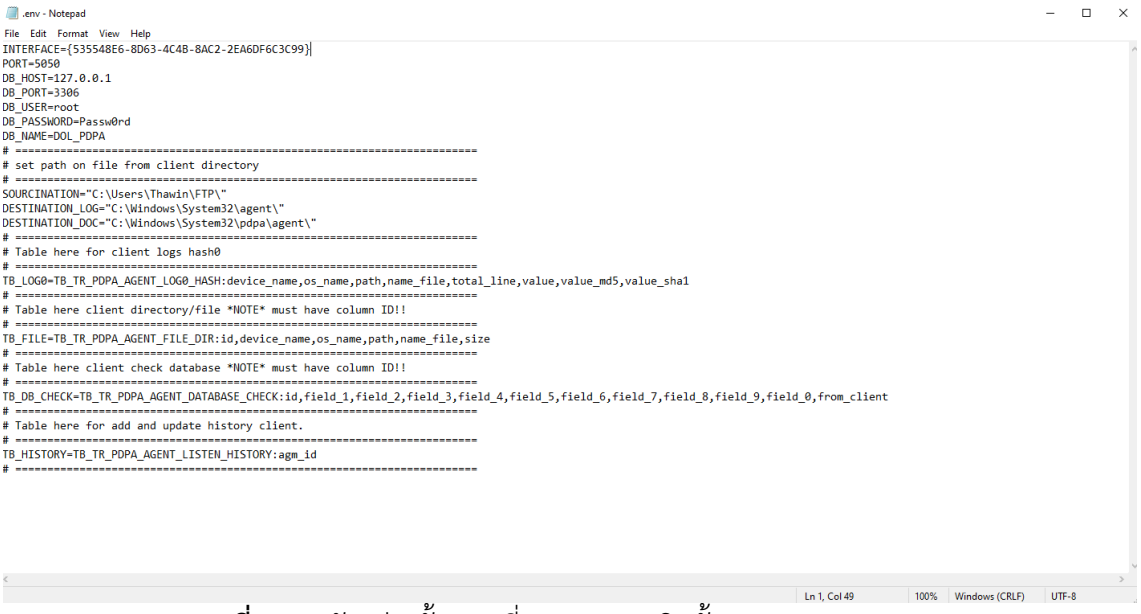
ภาพที่ ก.14 ตัวอย่างขั้นตอนที่ 13 ของการติดตั้ง Script agent listen

14) เมื่อหน้าต่างแสดงขึ้นมาให้ทำการ คัดลอก โดยการ คลิกขวา “Copy” หรือ “Ctrl+c” ข้อความที่ถูกคุมดำไว้



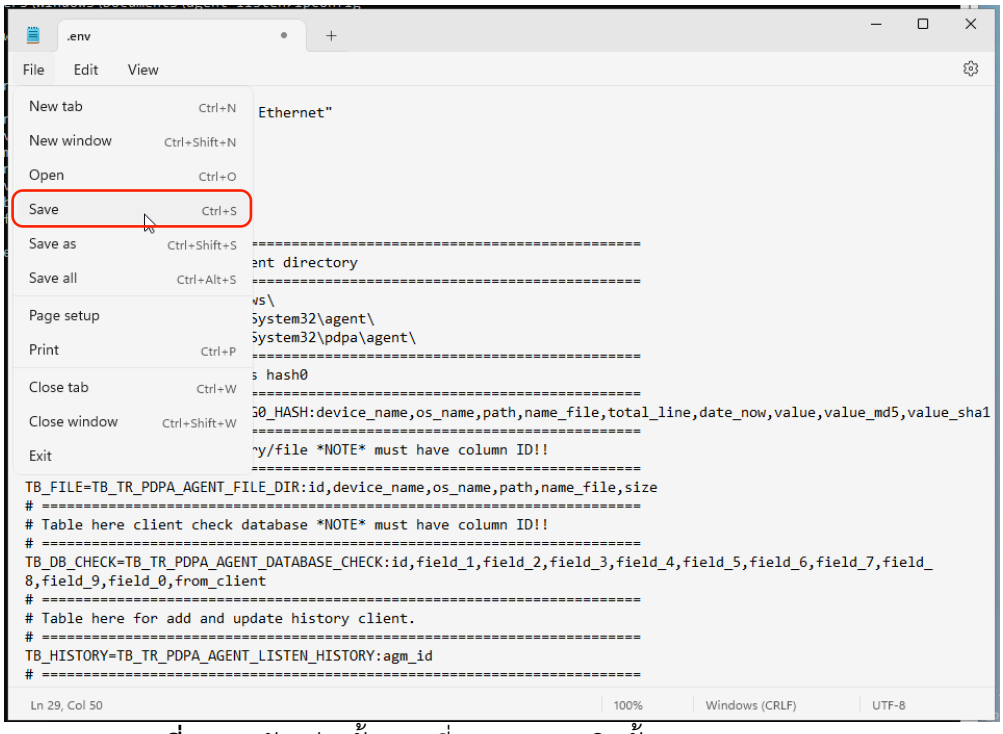
ภาพที่ ก.15 ตัวอย่างขั้นตอนที่ 14 ของการติดตั้ง Script agent listen

15) และกลับมายังไฟล์ “.env” ที่เปิดไว้แล้วเติมข้อความที่คัดลอกจากข้อก่อนหน้า วาง โดยการ คลิกขวา “Paste” หรือ “Ctrl+v” ที่ด้านหลัง “=” ของข้อความ “INTERFACE” และแก้ไขส่วนอื่นให้ครบเป็นอันเสร็จสิ้น



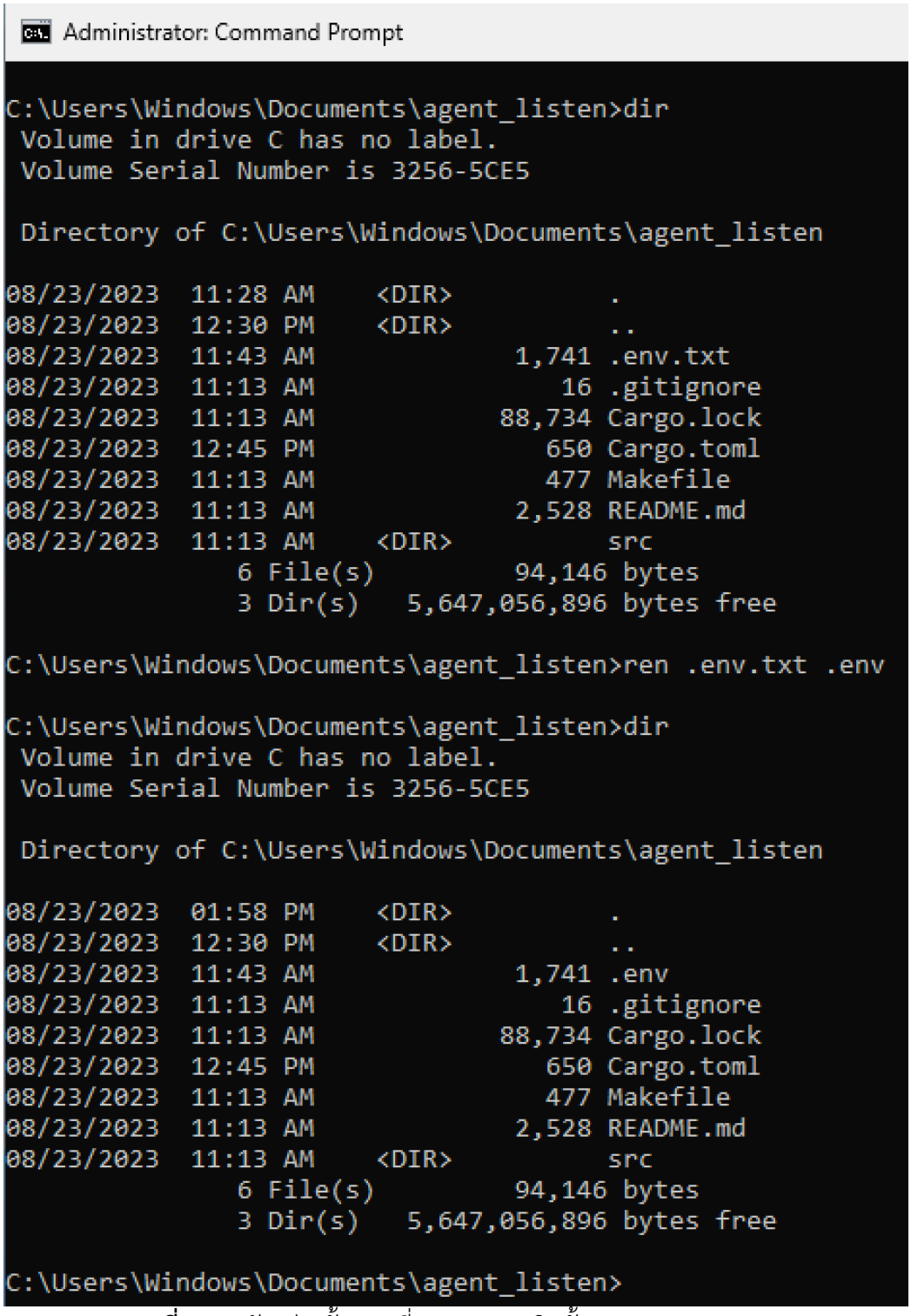
ภาพที่ ก.16 ตัวอย่างขั้นตอนที่ 15 ของการติดตั้ง Script agent listen

16) จากนั้น คลิกมุมบนซ้าย “File” คลิกเลือก “Save” หรือจะทำการ กดปุ่ม “Ctrl+s” เพื่อเป็นการบันทึกไฟล์



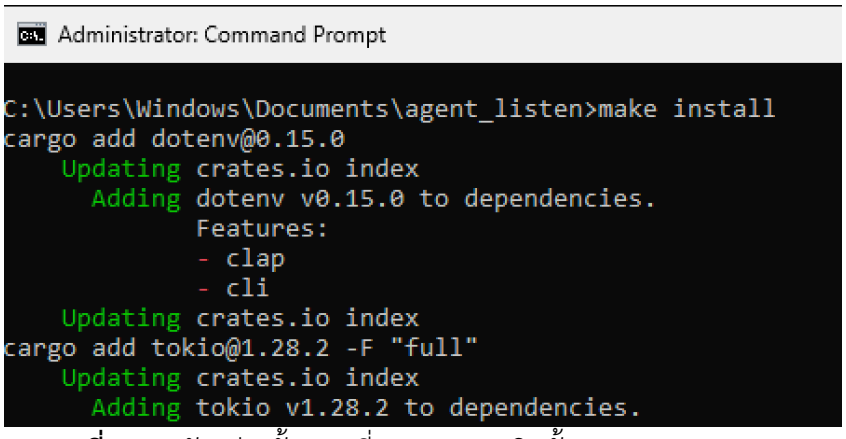
ภาพที่ ก.17 ตัวอย่างขั้นตอนที่ 16 ของการติดตั้ง Script agent listen

17) พิมพ์ “dir” สังเกตว่า .env มีนามสกุลเป็น “.txt” หรือไม่ ถ้าจริงทำการพิมพ์ “ren .env.txt .env” เพื่อทำการเปลี่ยนชื่อไฟล์ จากนั้นพิมพ์ “dir” อีกครั้ง เพื่อตรวจสอบอีกครั้ง

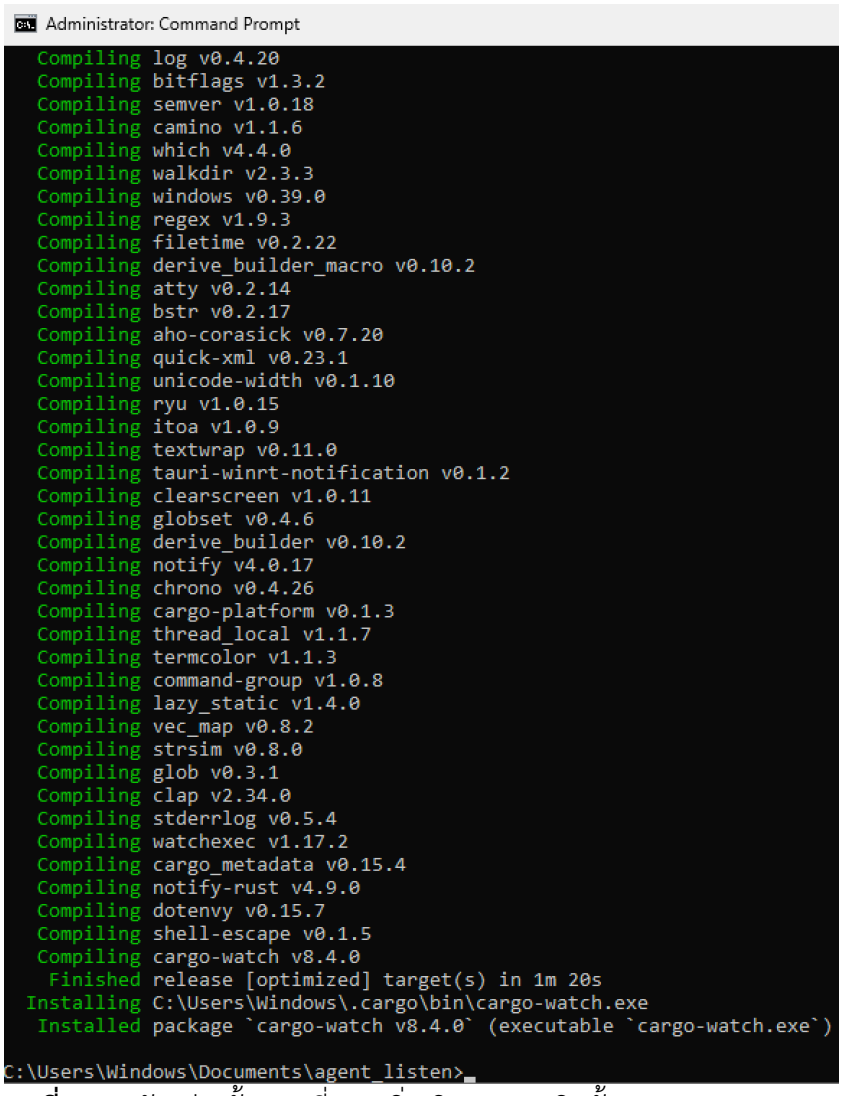


ภาพที่ ก.18 ตัวอย่างขั้นตอนที่ 17 ของการติดตั้ง Script agent listen

18) พิมพ์ “make install” จากนั้นรอจนเสร็จ

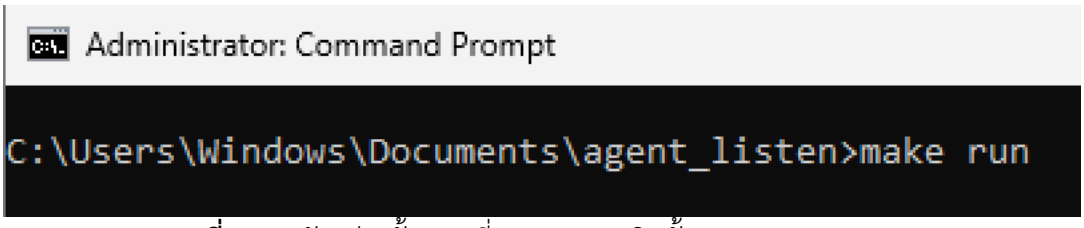


ภาพที่ ก.19 ตัวอย่างขั้นตอนที่ 18 ของการติดตั้ง Script agent listen



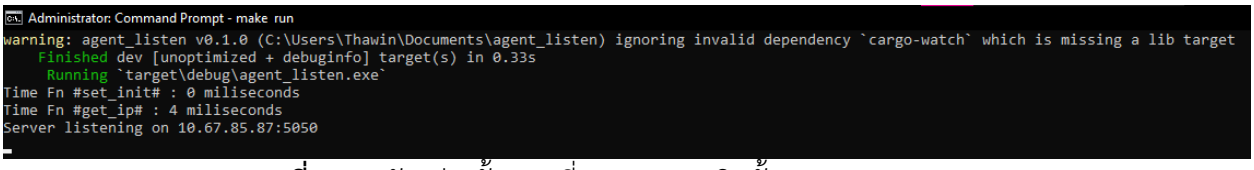
ภาพที่ ก.20 ตัวอย่างขั้นตอนที่ 18 เพิ่มเติมของการติดตั้ง Script agent listen

19) พิมพ์ “make run” เพื่อทำการทำงานระบบ agent listen



ภาพที่ ก.21 ตัวอย่างขั้นตอนที่ 19 ของการติดตั้ง Script agent listen

20) เมื่อทำงานครั้งแรกจะมีการดาวน์โหลดข้อมูลต่างๆ และเมื่อมีการแสดงถึง “IP:PORT” ดังภาพด้านล่างเป็นอันติดตั้งเสร็จสิ้น

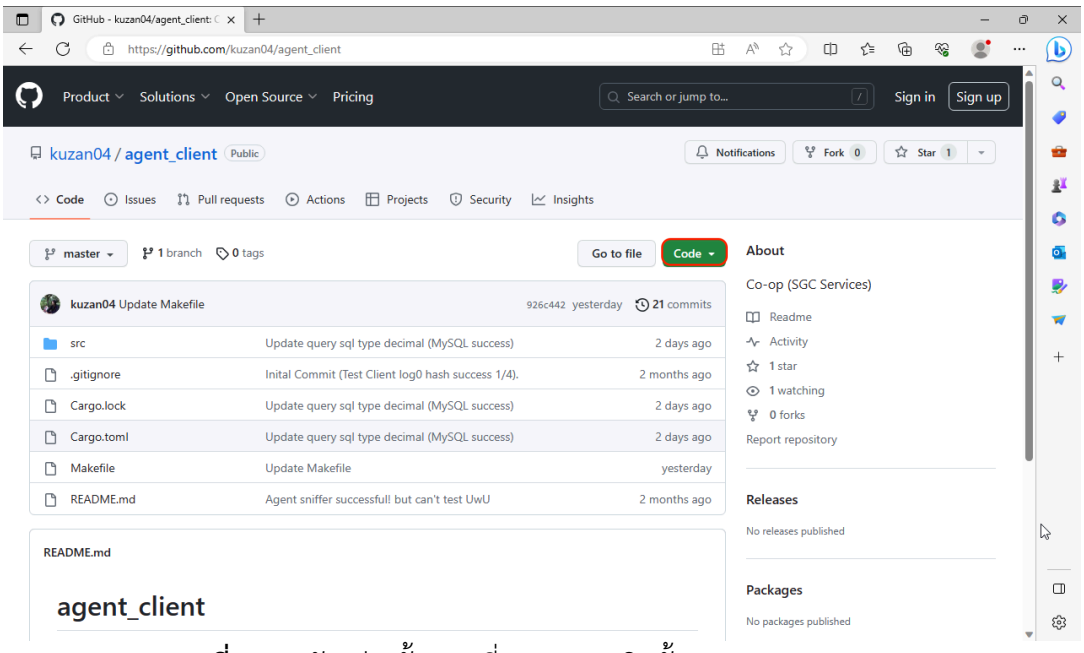


ภาพที่ ก.22 ตัวอย่างขั้นตอนที่ 20 ของการติดตั้ง Script agent listen

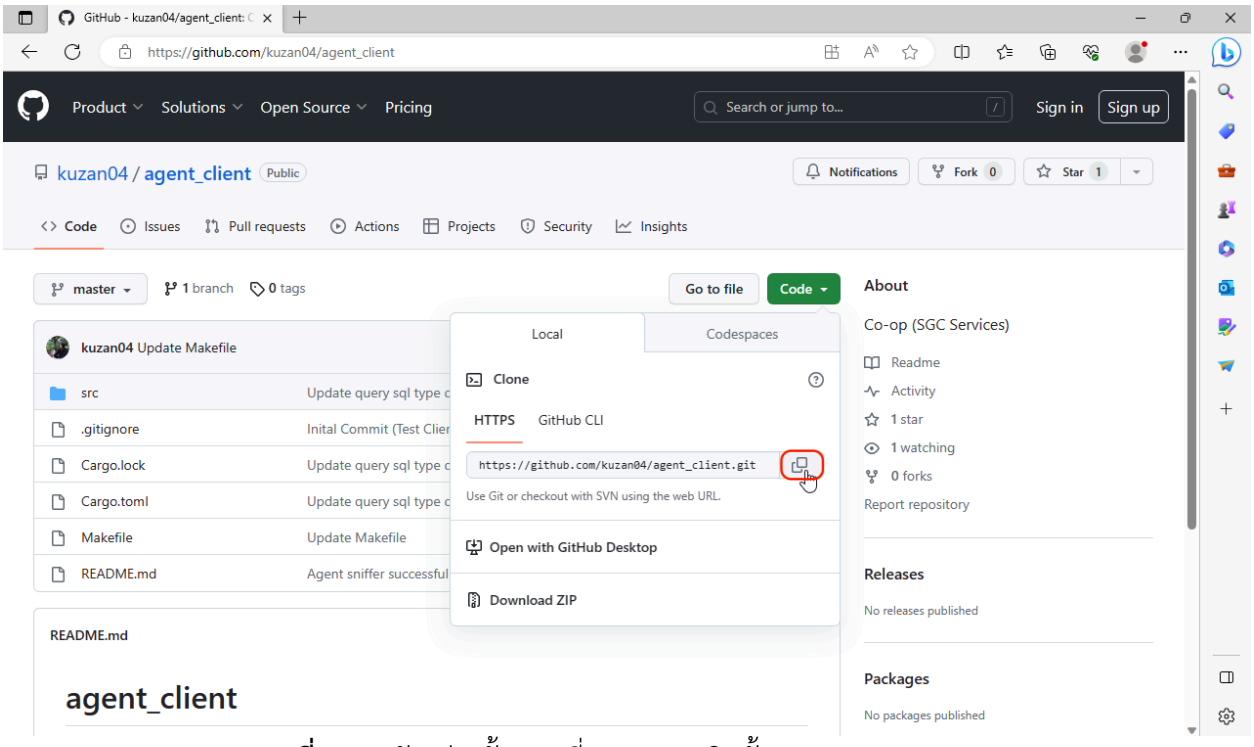
ติดตั้ง Script agent client

เนื่องจากตอนนี้ยังไม่รองรับ Windows อย่างเต็มรูปแบบ อาจจะมีปัญหาเกิดขึ้นได้เสมอ และไม่รองรับประเภท Sniffer

1) เปิดเว็บเบราว์เซอร์ชนิดใดก็ได้จากนั้นเข้าไปยังเว็บ “https://github.com/kuzan04/agent_client” เมื่อเข้าไป คลิกปุ่มสีเขียว “Code” จากนั้น คลิก “สี่เหลี่ยมซ้อนกัน” เพื่อทำการคัดวองข้อความด้านหน้านั้น

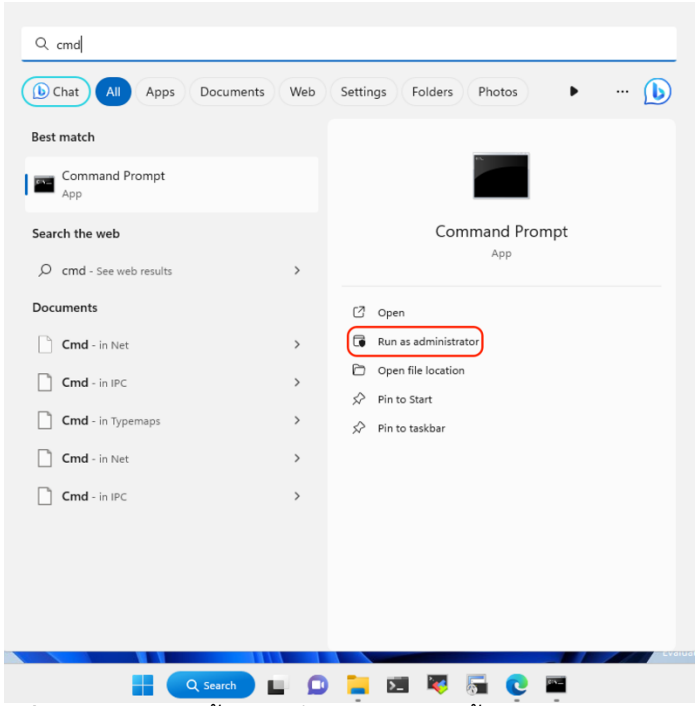


ภาพที่ ก.23 ตัวอย่างขั้นตอนที่ 1 ของการติดตั้ง Script agent client



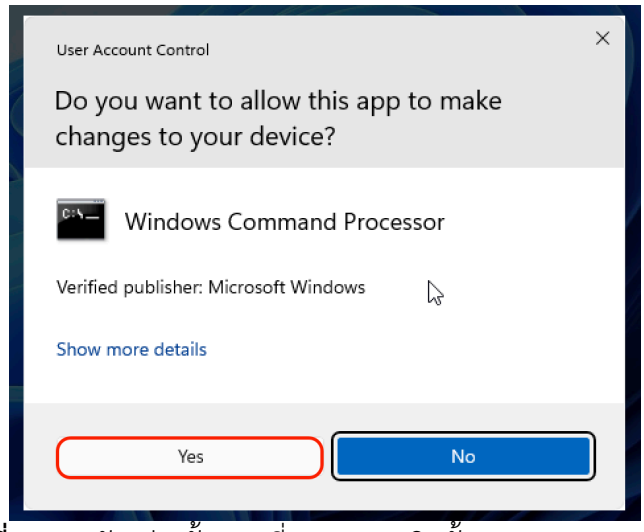
ภาพที่ ก.24 ตัวอย่างขั้นตอนที่ 1 ของการติดตั้ง Script agent client

2) ทำการ คลิก “Search” หรือรูป “แว่นขยาย” จากนั้นพิมพ์ “CMD” เลือก “Command Prompt” โดยให้ คลิก “Run as administrator”



ภาพที่ ก.25 ตัวอย่างขั้นตอนที่ 2 ของการติดตั้ง Script agent client

3) ในบางครั้งอาจจะมีหน้าต่าง “User Account Control” ให้ทำการ คลิก “Yes”



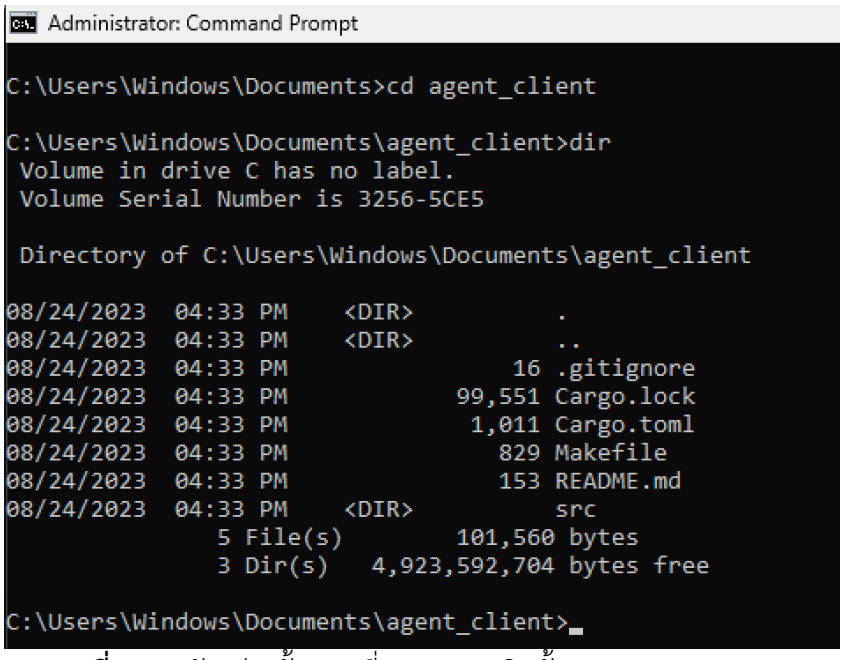
ภาพที่ ก.26 ตัวอย่างขั้นตอนที่ 3 ของการติดตั้ง Script agent client

4) จากนั้นทำการย้ายที่อยู่ของโฟลเดอร์ “cd ..\..\Users\##User##\Documents” จากนั้นเมื่อย้ายแล้ว พิมพ์ “git clone และข้อความที่ได้จากคัดลอกมาวาง” กดปุ่ม “Enter” ต่อมา พิมพ์ “dir” เพื่อแสดงรายการโฟลเดอร์และไฟล์มีอยู่ขณะตรงนี้

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.25247.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd ..\..\Users\Windows\Documents

C:\Users\Windows\Documents>git clone https://github.com/kuzan04/agent_client.git
Cloning into 'agent_client'...
remote: Enumerating objects: 151, done.
remote: Counting objects: 100% (151/151), done.
remote: Compressing objects: 100% (103/103), done.
Receiving objects: 100% (151/151), 65.32 KiB | 302.00 KiB/s, done.
remote: Total 151 (delta 95), reused 97 (delta 45), pack-reused 0
Resolving deltas: 100% (95/95), done.

C:\Users\Windows\Documents>dir
 Volume in drive C has no label.
 Volume Serial Number is 3256-5CE5

 Directory of C:\Users\Windows\Documents

08/24/2023  04:33 PM    <DIR>          .
08/23/2023  12:32 PM    <DIR>          ..
08/24/2023  04:33 PM    <DIR>          agent_client
08/23/2023  12:25 PM    <DIR>          IISExpress
08/23/2023  12:25 PM    <DIR>          My Web Sites
08/23/2023  12:32 PM    <DIR>          Visual Studio 2022
               0 File(s)              0 bytes
               6 Dir(s)   4,924,428,288 bytes free

C:\Users\Windows\Documents>
```

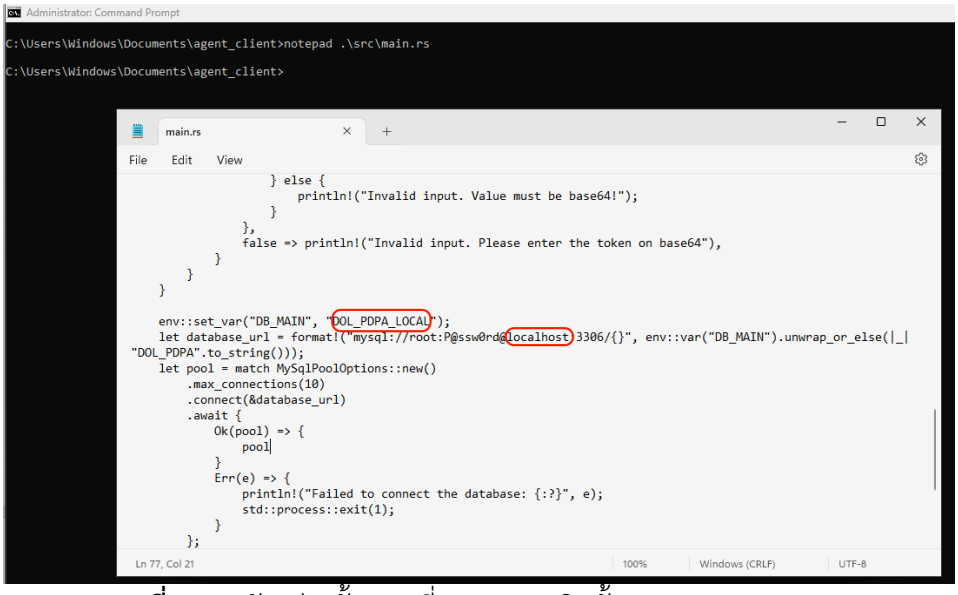
ภาพที่ ก.27 ตัวอย่างขั้นตอนที่ 4 ของการติดตั้ง Script agent client

5) พิมพ์ “cd agent_client” เพื่อเข้าไปยังโฟลเดอร์นั้น จากนั้นเพื่อทำการตรวจสอบรายการไฟล์ พิมพ์ “dir”



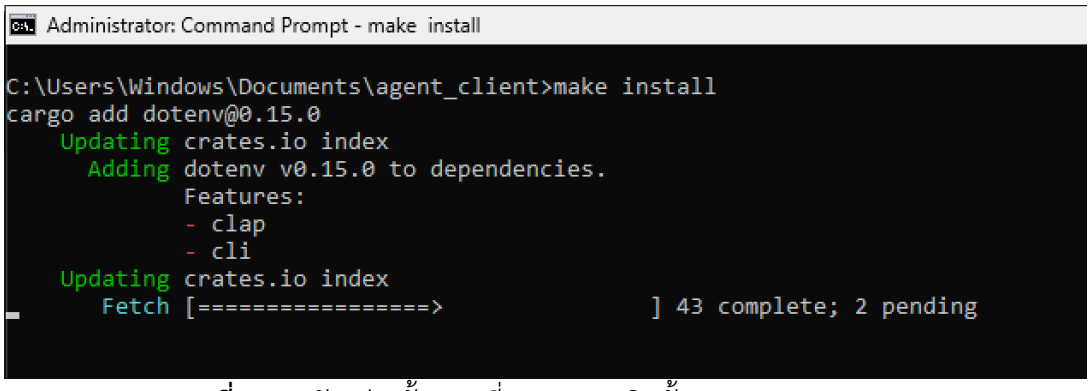
ภาพที่ ก.28 ตัวอย่างขั้นตอนที่ 5 ของการติดตั้ง Script agent client

6) พิมพ์ “notepad .\src\main.rs” เพื่อเข้าไปแก้ไขไฟล์ และให้ค้นหาหรือสังเกตหา เพื่อทำการแก้ไขตัวแปรให้ถูกต้อง “env::set_var("DB_MAIN", "DOL_PDPA_LOCAL");” และอีกหนึ่งที่ต้องแก้ “let database_url = format!("mysql://root:P@ssw0rd@localhost:3306/{}"....” เป็นไปตามที่ต้องการ โดยอย่างแรกคือ ชื่อฐานข้อมูลหลัก และ อย่างที่สอง คือ IP ของฐานข้อมูลหลัก

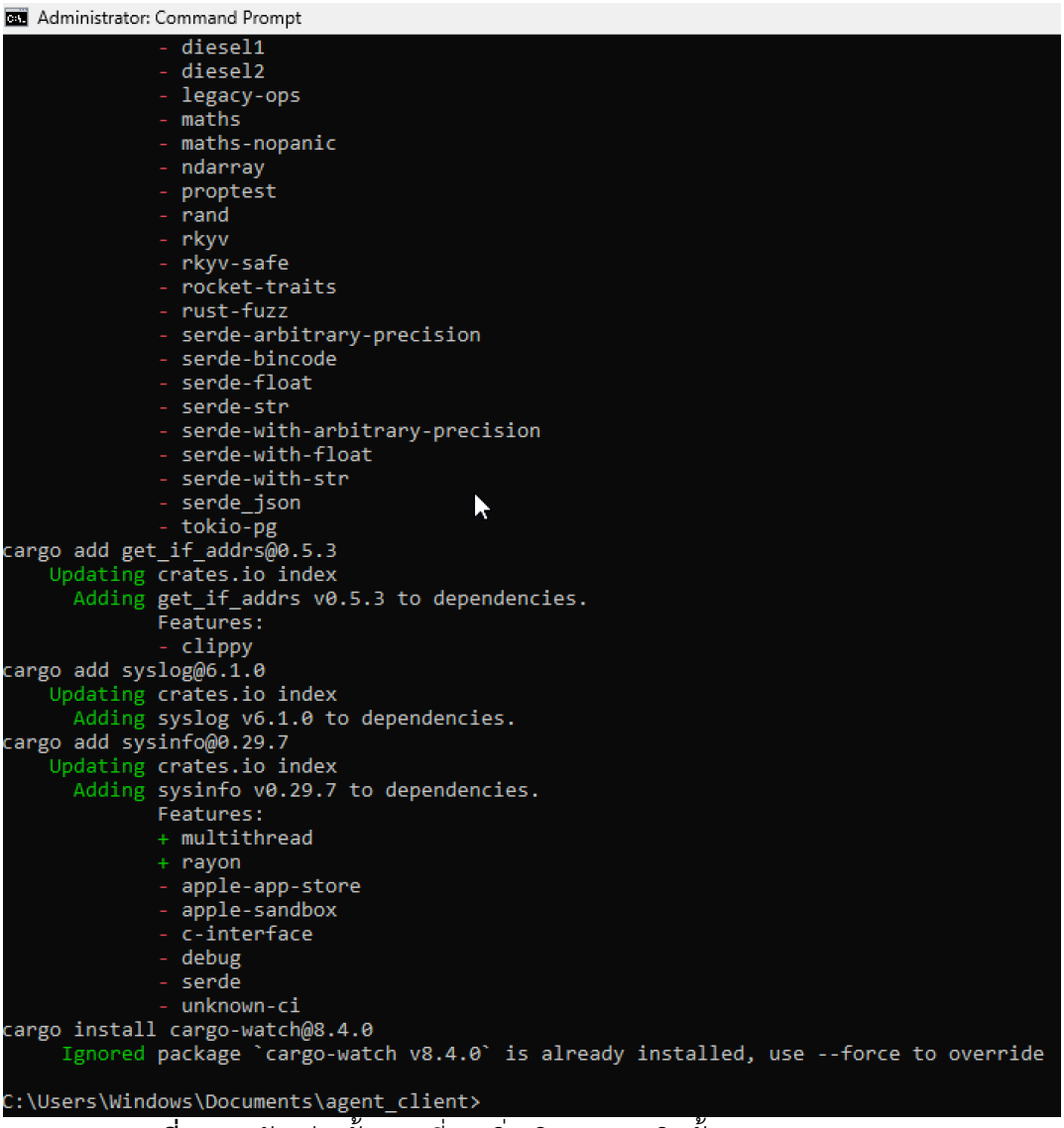


ภาพที่ ก.29 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent client

7) พิมพ์ “make install” จากนั้นรอจนเสร็จ

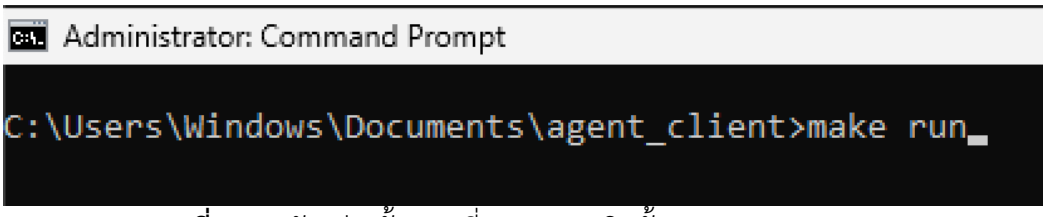


ภาพที่ ก.30 ตัวอย่างขั้นตอนที่ 7 ของการติดตั้ง Script agent client

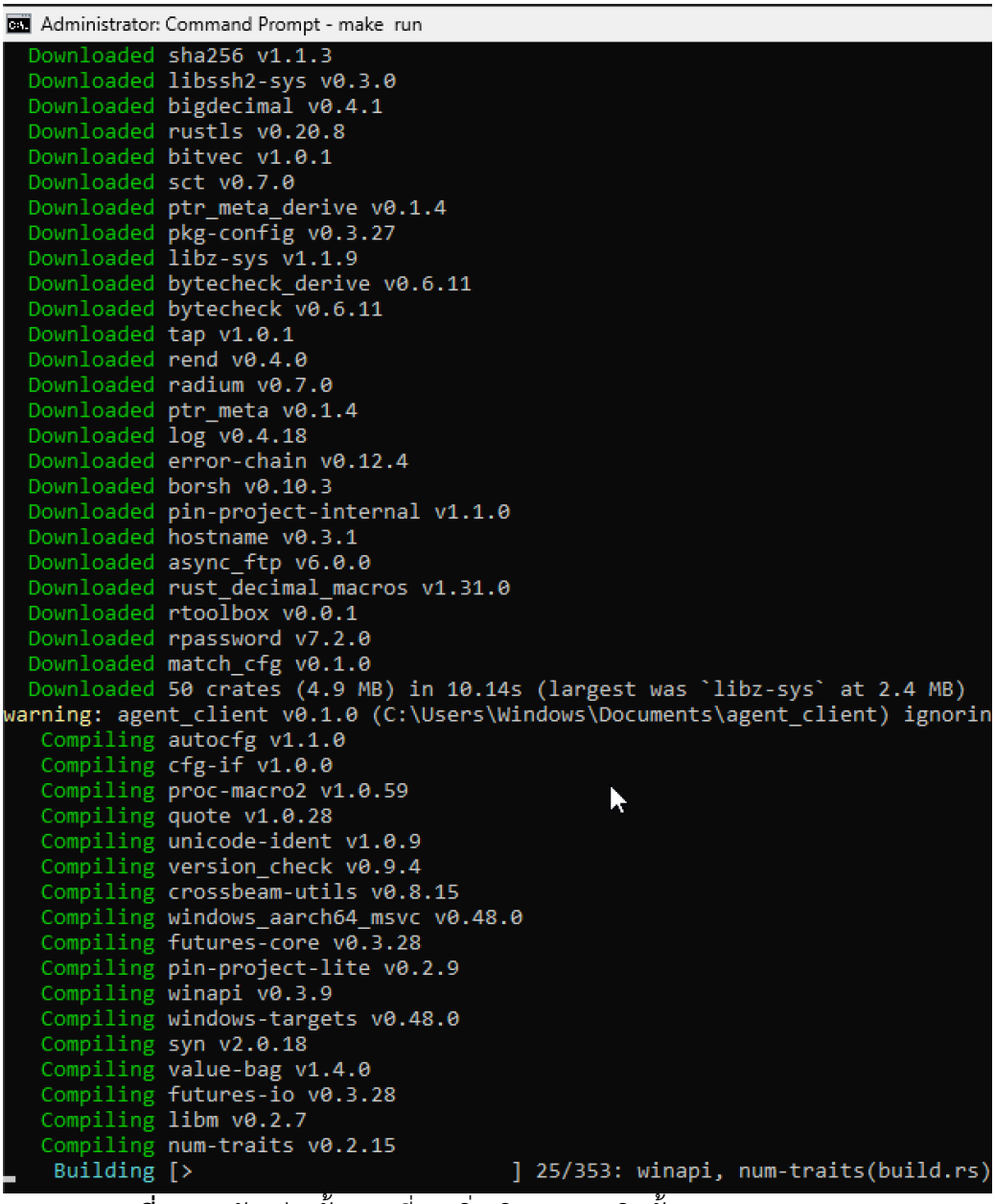


ภาพที่ ก.31 ตัวอย่างขั้นตอนที่ 7 เพิ่มเติมของการติดตั้ง Script agent client

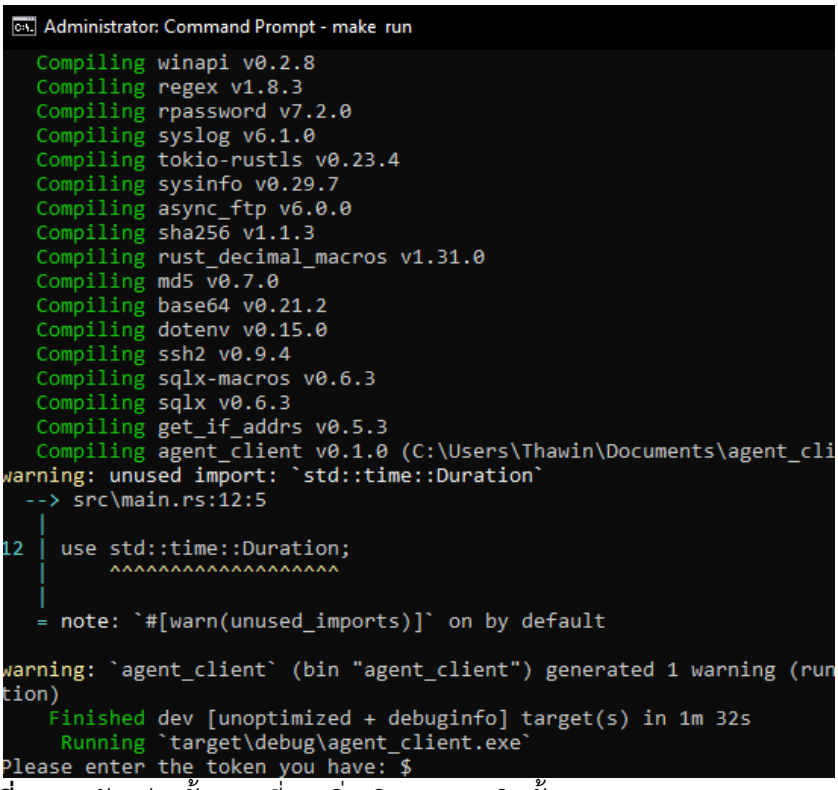
8) พิมพ์ “make run” เพื่อทำการทำงานระบบ agent client (ทำงานครั้งแรกอาจจะมีการดาวน์โหลดและจัดาการไฟล์ต่างๆ) จากนั้น script จะต้องการ token



ภาพที่ ก.32 ตัวอย่างขั้นตอนที่ 8 ของการติดตั้ง Script agent client

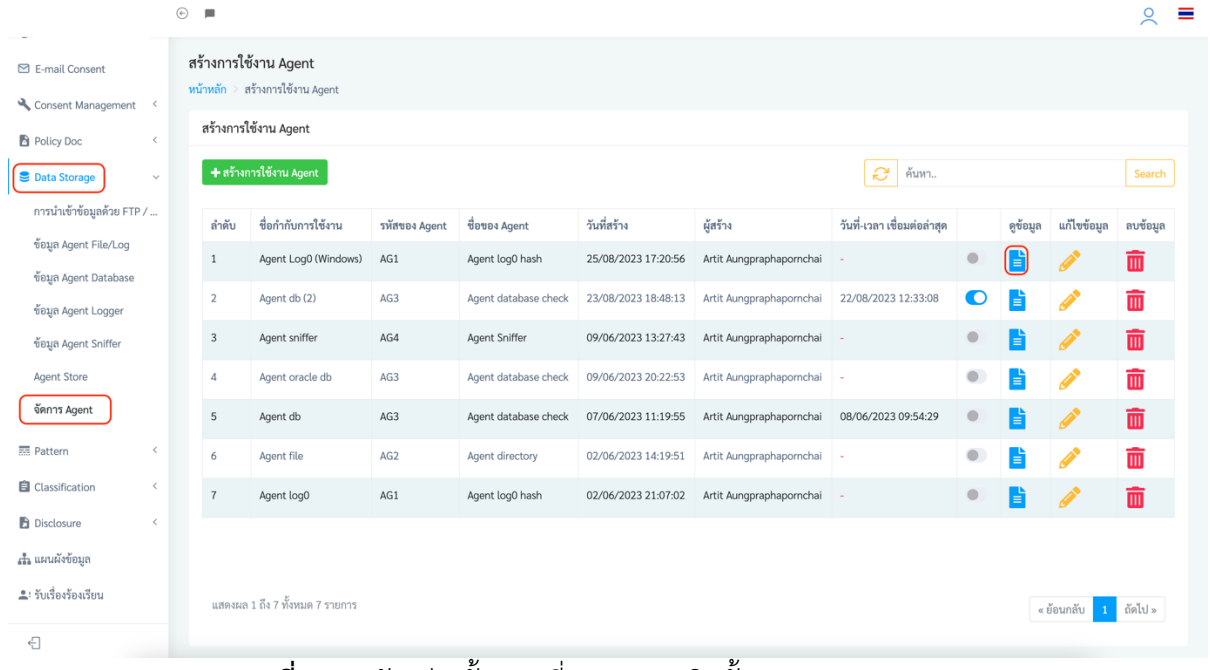


ภาพที่ ก.33 ตัวอย่างขั้นตอนที่ 8 เพิ่มเติมของการติดตั้ง Script agent client



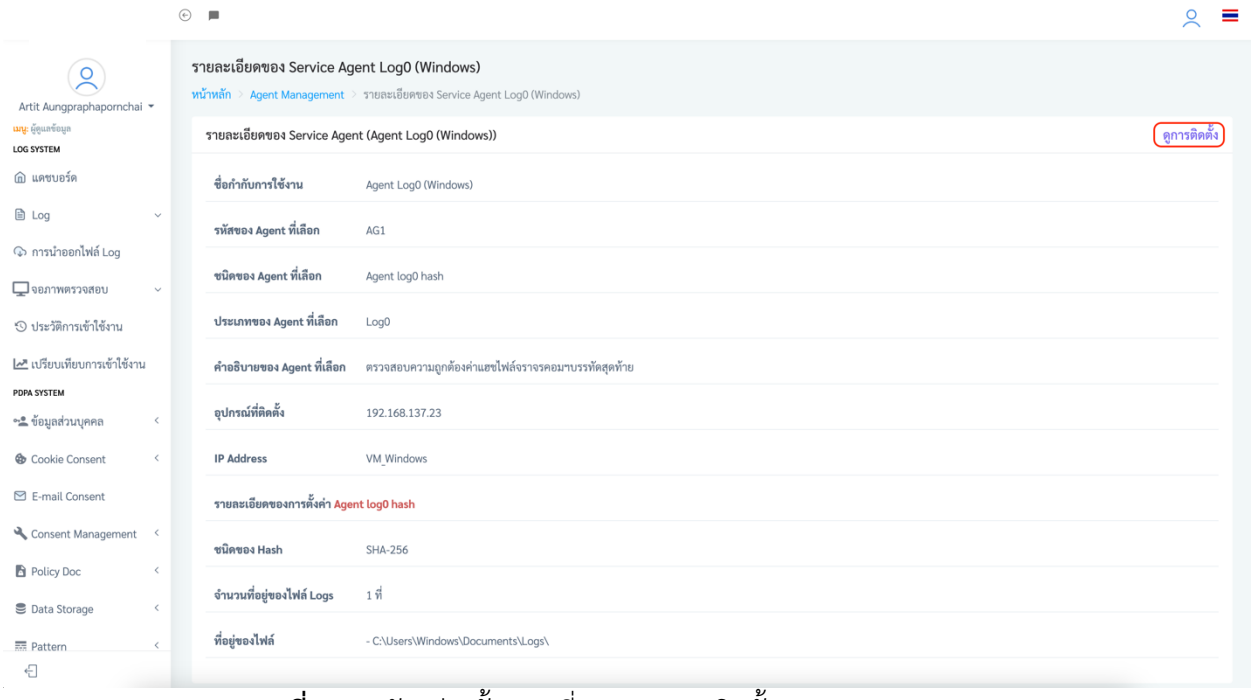
ภาพที่ ก.34 ตัวอย่างขั้นตอนที่ 8 เพิ่มเติมของการติดตั้ง Script agent client

9) กลับเข้าไปยังเว็บแอพพลิเคชัน เมื่อเข้าไป คลิกแถบเมนูด้านซ้ายมือ “Data Store” จากนั้น คลิก “จัดการ Agent” คลิกรูป “เอกสารสีน้ำเงิน” เพื่อเข้าไปยังหน้ารายละเอียดของที่เลือกนั้นๆ



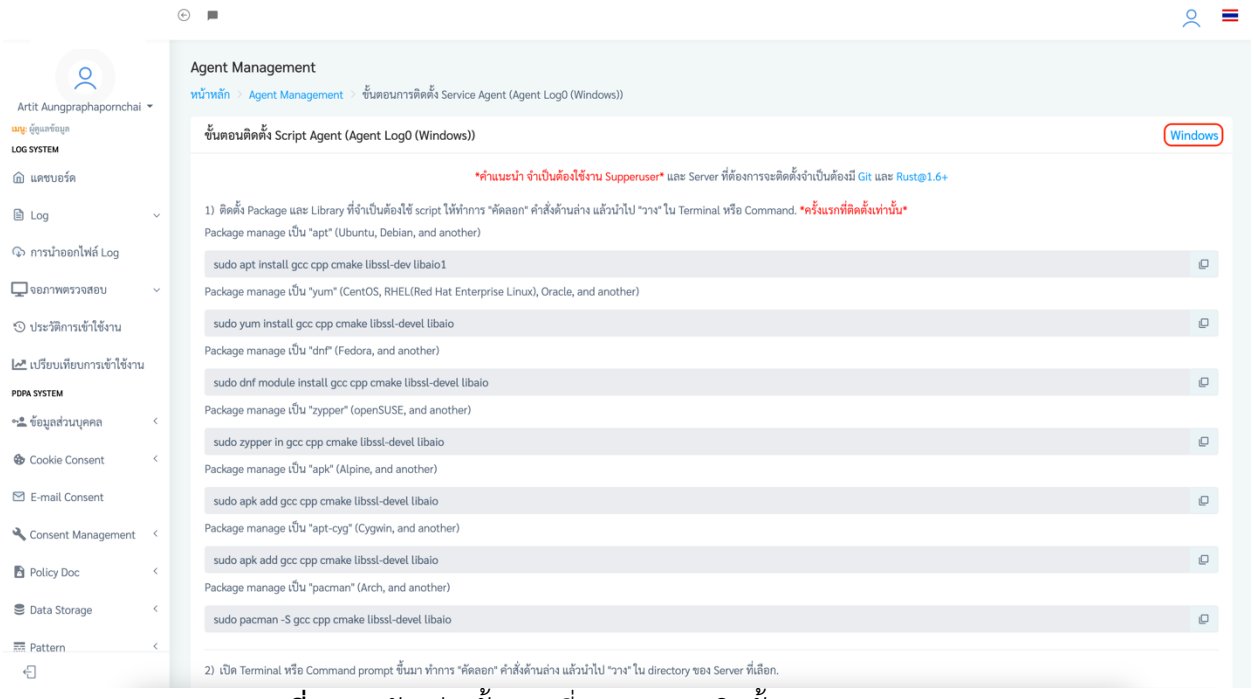
ภาพที่ ก.35 ตัวอย่างขั้นตอนที่ 9 ของการติดตั้ง Script agent client

10) ด้านมุมบน คลิก “ดูการติดตั้ง” เพื่อทำการเข้าไปคัดลอก “Token” เพื่อนำมาใช้งาน



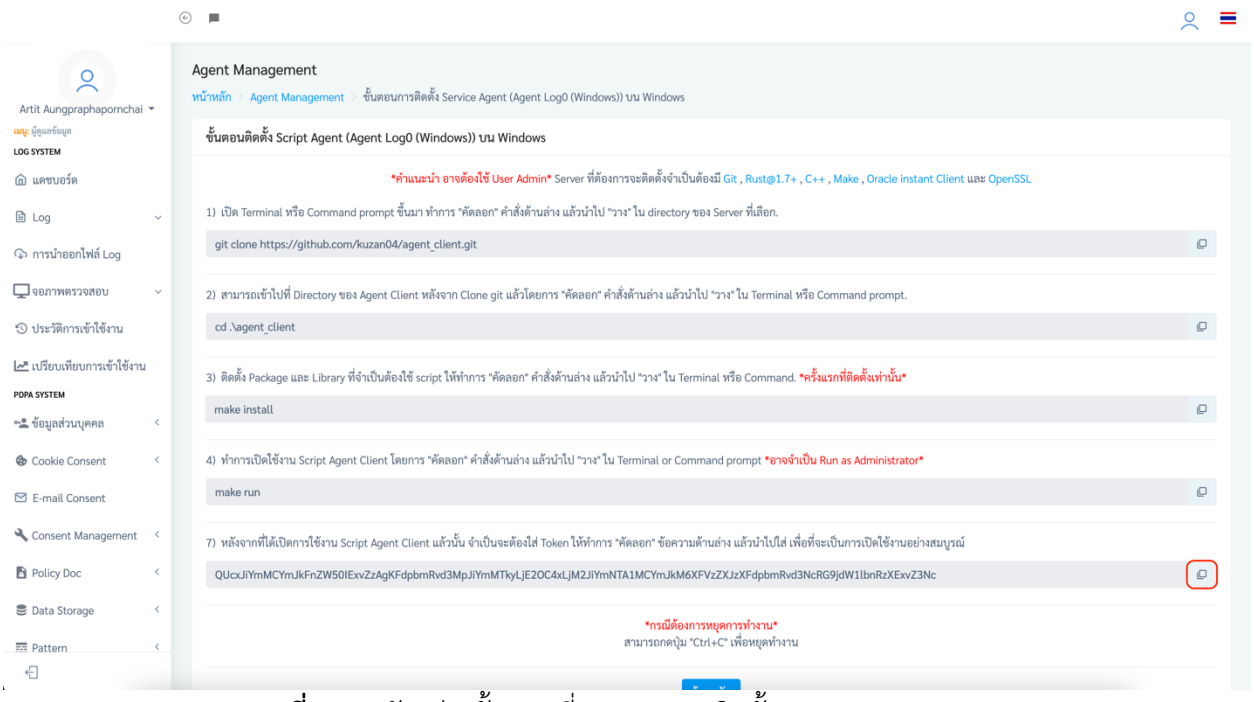
ภาพที่ ก.36 ตัวอย่างขั้นตอนที่ 10 ของการติดตั้ง Script agent client

11) จากนั้นด้านมุมบนอีกครั้งหนึ่ง คลิก “Windows” เพื่อเข้าไปยังส่วนของ “Windows”



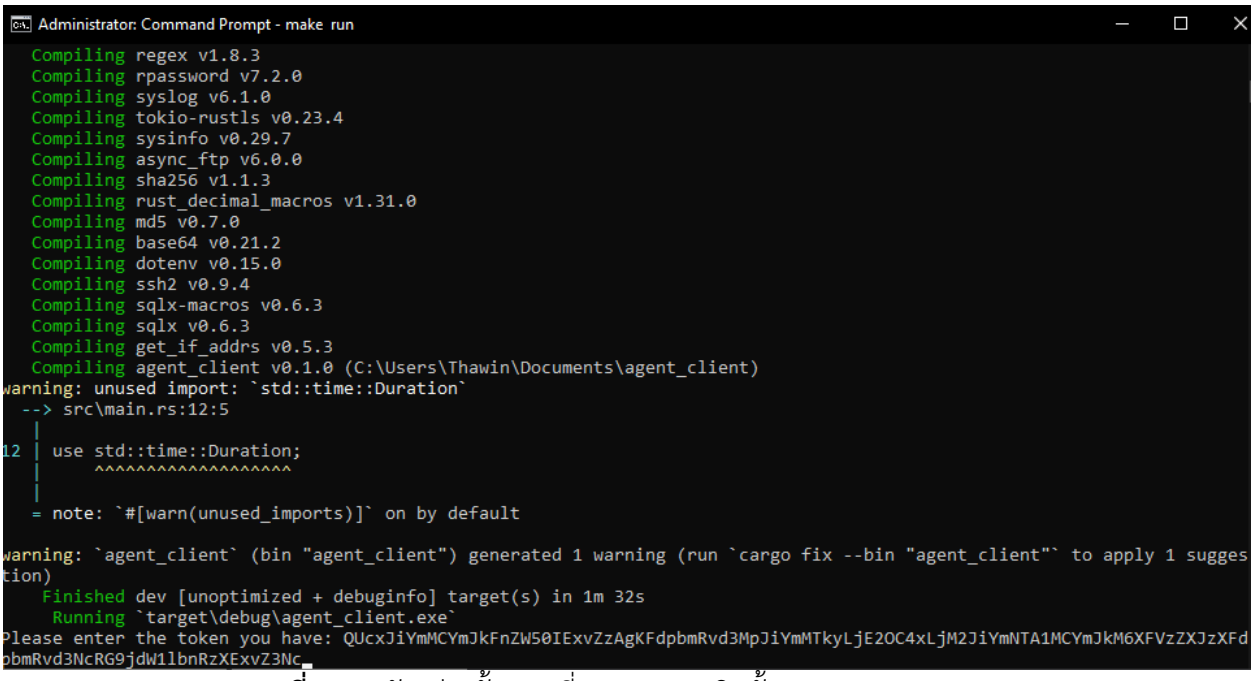
ภาพที่ ก.37 ตัวอย่างขั้นตอนที่ 11 ของการติดตั้ง Script agent client

12) ทำการคลิกรูป “สี่เหลี่ยมซ้อนกัน” เพื่อเป็นการคัดลอกข้อความด้านหน้านั้น



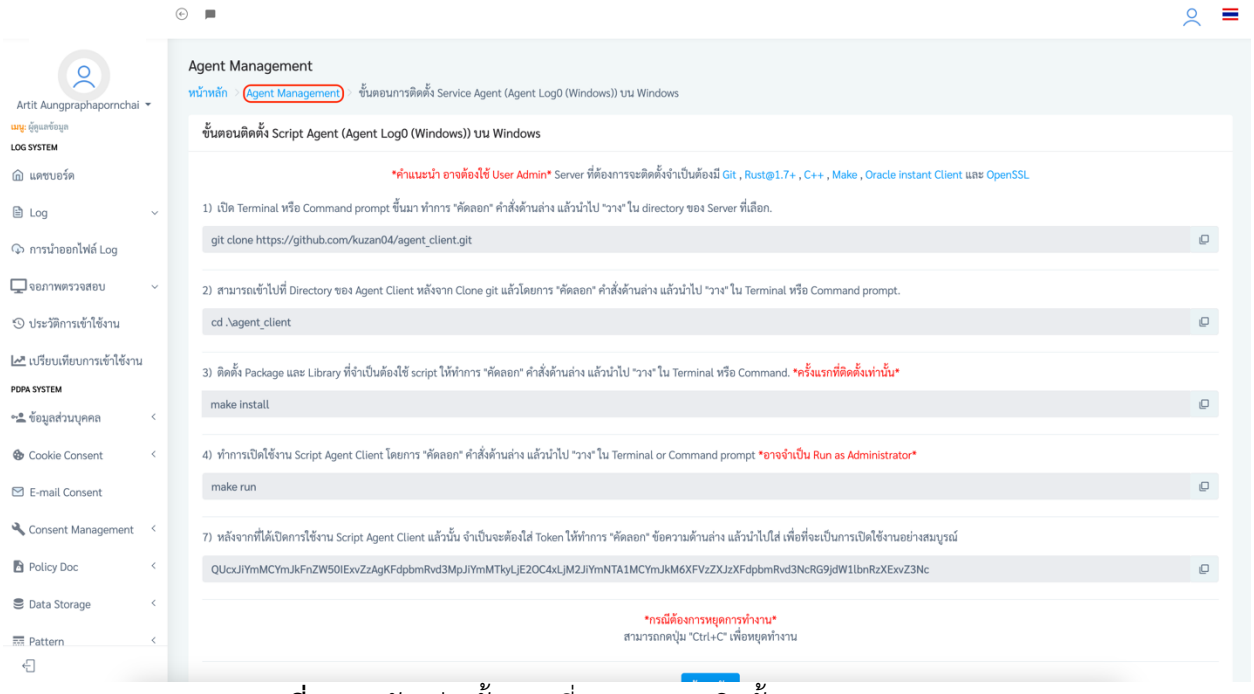
ภาพที่ ก.38 ตัวอย่างขั้นตอนที่ 12 ของการติดตั้ง Script agent client

13) ทำการนำข้อความที่ได้คัดลอกจากข้อด้านบน วางลงไป กดปุ่ม “Enter”



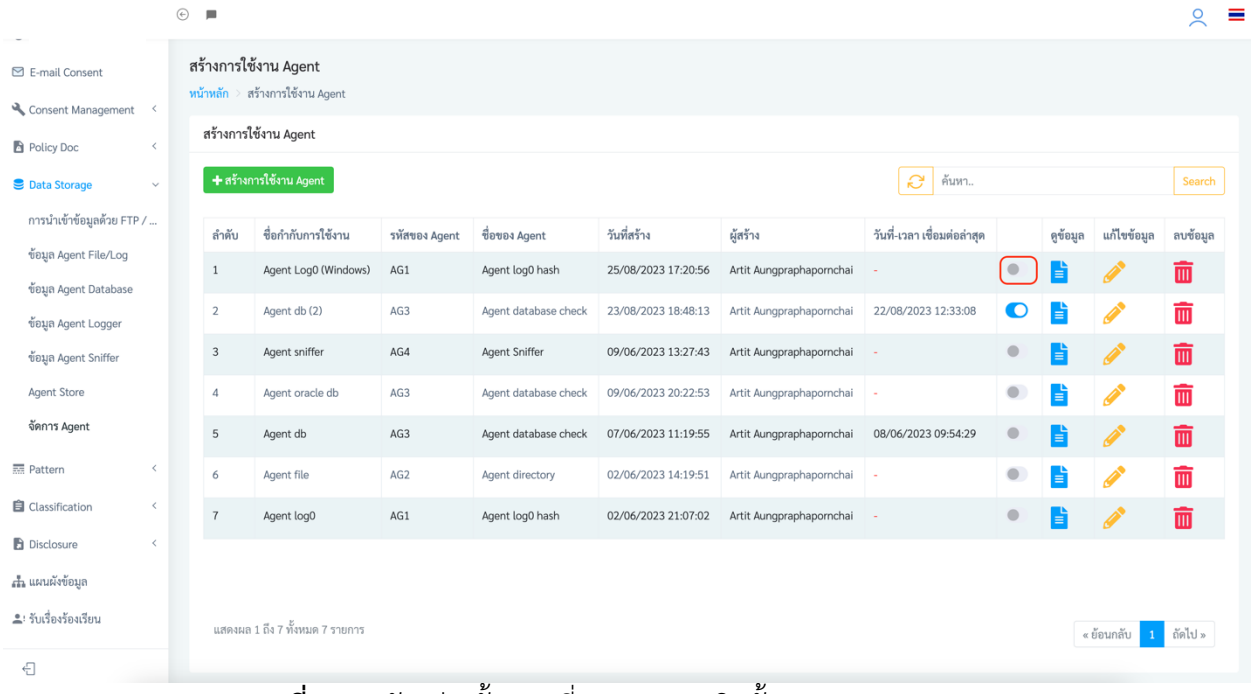
ภาพที่ ก.39 ตัวอย่างขั้นตอนที่ 13 ของการติดตั้ง Script agent client

14) กลับไปยังเว็บเบราว์เซอร์ จากนั้นทำการคลิก คำว่า “Agent Management” ด้านบนดังภาพด้านล่างนี้

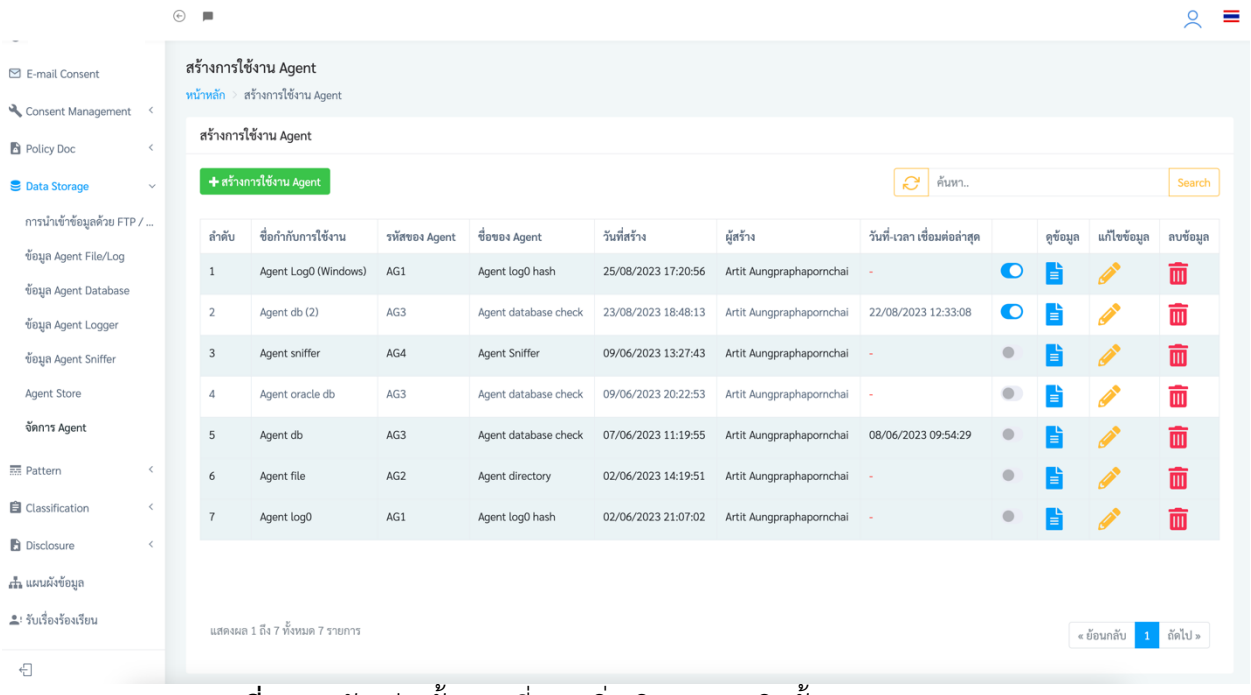


ภาพที่ ก.40 ตัวอย่างขั้นตอนที่ 14 ของการติดตั้ง Script agent client

15) จากนั้นคลิกปุ่มที่คล้ายกับสวิตช์เพื่อเปิดใช้งาน script นั้นๆ เป็นอันเสร็จสิ้น



ภาพที่ ก.41 ตัวอย่างขั้นตอนที่ 15 ของการติดตั้ง Script agent client



ภาพที่ ก.42 ตัวอย่างขั้นตอนที่ 15 เพิ่มเติมของการติดตั้ง Script agent client

คู่มือการติดตั้ง Agent listen & client เฉพาะ Ubuntu server 22.04 หรือใหม่กว่า

ติดตั้ง Script agent listen

1) พิมพ์ “cd” เพื่อกลับมายังที่อยู่ของผู้ใช้งานก่อน “sudo su” จากนั้นเมื่อยังไม่เคยใช้งานซุปเปอร์ผู้ใช้งานหรือใช้งานไปแล้วนานๆ ทำการกรอกรหัสผ่านของผู้ใช้งานที่ใช้งานอยู่ ขณะนั้น กดปุ่ม “Enter” ทำการพิมพ์ หรือ คัดลอก วาง “git clone https://github.com/kuzan04/agent_listen.git” กดปุ่ม “Enter” รอจนเสร็จสิ้น

```
administrator@aplha:~$ sudo su
[sudo] password for administrator:
root@aplha:/home/administrator# git clone https://github.com/kuzan04/agent_listen.git
Cloning into 'agent_listen'...
remote: Enumerating objects: 207, done.
remote: Counting objects: 100% (207/207), done.
remote: Compressing objects: 100% (131/131), done.
remote: Total 207 (delta 133), reused 139 (delta 69), pack-reused 0
Receiving objects: 100% (207/207), 56.34 KiB | 1.08 MiB/s, done.
Resolving deltas: 100% (133/133), done.
root@aplha:/home/administrator# _
```

ภาพที่ ก.43 ตัวอย่างขั้นตอนที่ 1 ของการติดตั้ง Script agent listen

2) พิมพ์ “cd ./agent_listen” เพื่อเข้าไปยัง directory ของ agent listen จากนั้น พิมพ์ “ls -l” เพื่อทำการแสดงรายการไฟล์ต่างๆ

```
root@aplha:/home/administrator# cd agent_listen/
root@aplha:/home/administrator/agent_listen# ls -l
total 100
-rw-r--r-- 1 root root 85314 Aug 15 09:29 Cargo.lock
-rw-r--r-- 1 root root   650 Aug 15 09:29 Cargo.toml
-rw-r--r-- 1 root root   462 Aug 15 09:29 Makefile
-rw-r--r-- 1 root root  2543 Aug 15 09:29 README.md
drwxr-xr-x 3 root root  4096 Aug 15 09:29 src
root@aplha:/home/administrator/agent_listen# _
```

ภาพที่ ก.44 ตัวอย่างขั้นตอนที่ 2 ของการติดตั้ง Script agent listen

3) พิมพ์ “ip a” จากนั้นดูผลลัพธ์ในอินเตอร์เฟสเน็ตเวิร์คที่ต้องการเปิดใช้งาน script agent listen จดจำไว้ หรือคัดลอก ชื่อของอินเตอร์เฟสนั้นๆ

```
root@aplha:/home/administrator/agent_listen# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 56:da:ee:49:25:26 brd ff:ff:ff:ff:ff:ff
    inet 192.168.137.37/24 brd 192.168.137.255 scope global enp0s1
       valid_lft forever preferred_lft forever
    inet6 fd9f:eaa2:1988:9421:54da:eeff:fe49:2526/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 2591992sec preferred_lft 604792sec
    inet6 fe80::54da:eeff:fe49:2526/64 scope link
       valid_lft forever preferred_lft forever
root@aplha:/home/administrator/agent_listen#
```

ภาพที่ ก.45 ตัวอย่างขั้นตอนที่ 3 ของการติดตั้ง Script agent listen

4) โดยจะใช้งาน file editor Vim หรือ Nano ก็สามารถเลือกได้โดย Vim พิมพ์ “vi .env” หรือ “nano .env” กดปุ่ม “Enter” เพื่อทำการสร้างไฟล์ .env ไว้ให้ script agent listen ใช้งาน (ตัวอย่างจะใช้เป็น Vim)

```
root@aplha:/home/administrator/agent_listen# vi .env
```

ภาพที่ ก.46 ตัวอย่างขั้นตอนที่ 4 ของการติดตั้ง Script agent listen

5) เมื่อเปิดไฟล์ .env เสร็จ ทำการเข้าไปยัง URL “https://github.com/kuzan04/agent_listen” เพื่อทำการคัดลอก วาง หรือ พิมพ์เหมือนกับตัวอย่างดังภาพด้านล่าง

Example

```
INTERFACE= ขาอินเตอร์เน็ตที่ต้องการเปิดใช้งาน
PORT= port ที่ต้องการให้ listener
DB_HOST= database ip host
DB_PORT= database port
DB_USER= database username
DB_PASSWORD= database password
DB_NAME= database name
# ========================================================================
# set path on file from client directory
# ========================================================================
SOURCINATION= ที่อยู่ของการจัดเก็บไฟล์ FTP in listener
DESTINATION_LOG= ที่อยู่หลังจากรับไฟล์ Log from FTP in lister
DESTINATION_DOC= ที่อยู่หลังจากรับไฟล์ Doc, CSV, XCSV from FTP in lister
# ========================================================================
# Table here for client logs hash0
# ========================================================================
TB_LOG0= ชื่อ:ตามโดย columns or fields ต่อกันด้วย ',' ไปจนถึงตัวสุดท้ายก็ไม่ต้องใส่
# ========================================================================
# Table here client directory/file *NOTE* must have column ID!!
# ========================================================================
TB_FILE= ชื่อ:ตามโดย columns or fields ต่อกันด้วย ',' ไปจนถึงตัวสุดท้ายก็ไม่ต้องใส่
# ========================================================================
# Table here client check database *NOTE* must have column ID!!
# ========================================================================
TB_DB_CHECK= ชื่อ:ตามโดย columns or fields ต่อกันด้วย ',' ไปจนถึงตัวสุดท้ายก็ไม่ต้องใส่
# ========================================================================
# ========================================================================
TB_HISTORY= ชื่อ:ตามโดย columns id ของ agent manager เท่านั้น
# ========================================================================
```

ภาพที่ ก.47 ตัวอย่างขั้นตอนที่ 5 ของการติดตั้ง Script agent listen

6) กลับไปยัง file editor ที่ได้เปิดไว้ ไม่ว่าจะเป็น Nano หรือ Vim จากนั้นทำการกรอกค่าที่เข้าไปยังไฟล์ .env ที่กำลังเปิดไว้อยู่ด้วยการพิมพ์ หรือ คัดลอกวาง (โดยตัวอย่างจะใช้เป็น Vim)

```
INTERFACE=enp0s1
PORT=5050
DB_HOST=192.168.137.37
DB_PORT=3306
DB_USER=alltra
DB_PASSWORD=P@ssw0rd
DB_NAME=alltra
#=======================================
# set path on file from client direcotry
#=======================================
SOURCINATION=/home/ftpuser/
DESTINATION_LOG=/var/log/agent/
DESTINATION_DOC=/var/pdpa/agent/
#=======================================
# Table here for client logs hash0
#=======================================
TB_LOG0=TB_TR_PDPA_AGENT_LOG0_HASH:device_name,os_name,path,name_file,total_line,        value,value_md5,value_sha1
#=======================================
# Table here for client directory/file *NOTE* must have column ID!!!
#=======================================
TB_FILE=TB_TR_PDPA_AGENT_FILE_DIR:id,device_name,os_name,path,name_file
#=======================================
# Table here for client database *NOTE* must have column ID!!!
#=======================================
TB_DB_CHECK=TB_TR_PDPA_AGENT_DATABASE_CHECK:id,field_1,field_2,field_3,field_3,field_4,field_5,field_6,field_7,field_8,field_9,field_0,from_client
#=======================================
#=======================================
TB_HISTORY=TB_TR_PDPA_AGENT_LISTEN_HISTORY:agm_id
#=======================================
```

ภาพที่ ก.48 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent listen

7) ทำการบันทึกไฟล์ โดยถ้าเป็น Nano คือ กดปุ่ม “Ctrl+x” กดปุ่ม “y” กดปุ่ม “Enter” แต่ถ้าเป็น Vim พิมพ์ “:x” กดปุ่ม “Enter” จากนั้นตรวจสอบดูไฟล์ว่าบันทึกหรือไม่ พิมพ์ “ls -al”

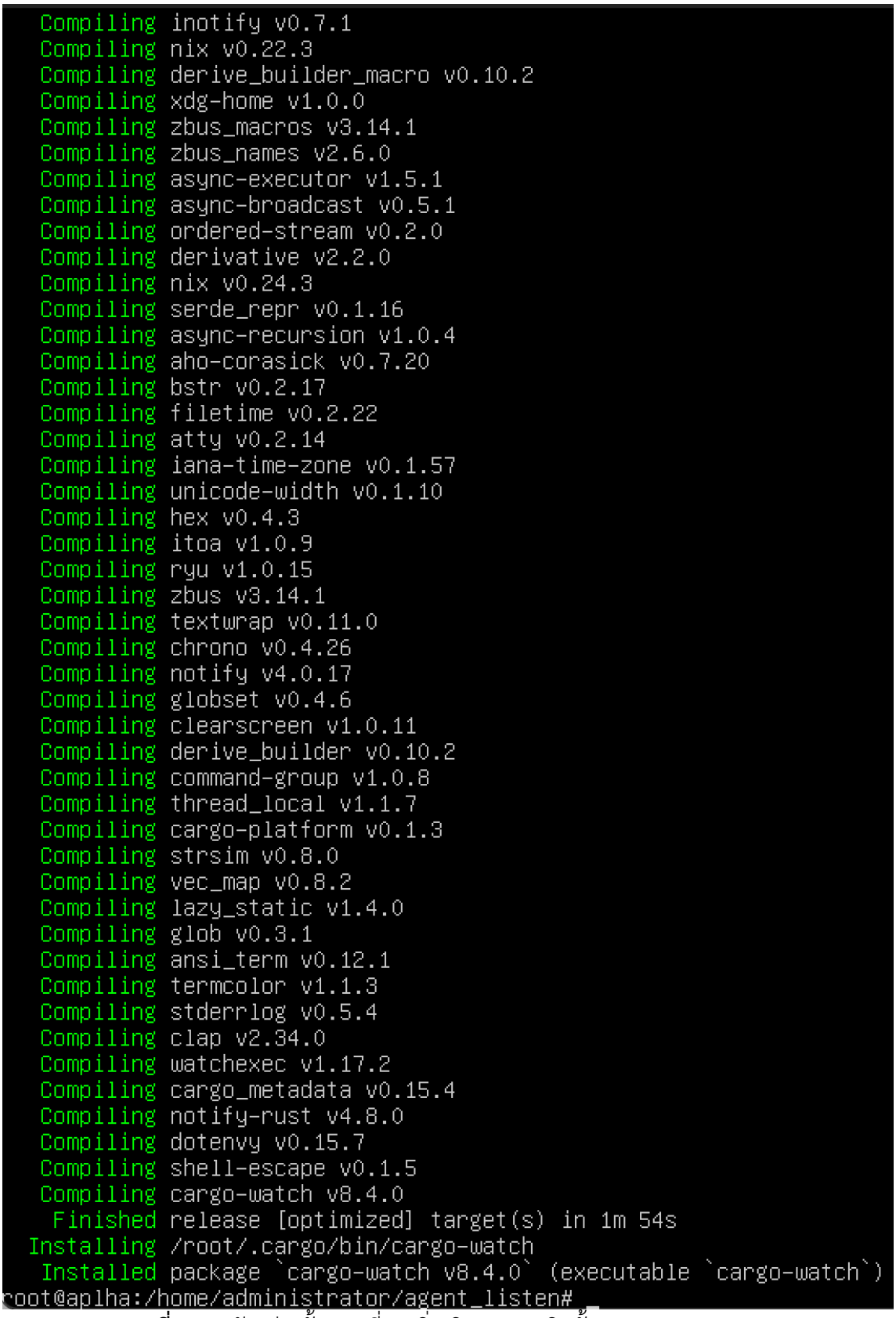
```
root@aplha:/home/administrator/agent_listen# ls -al
total 120
drwxr-xr-x 4 root          root           4096 Aug 15 13:18 .
drwxr-x--- 5 administrator administrator  4096 Aug 15 09:29 ..
-rw-r--r-- 1 root          root          85314 Aug 15 09:29 Cargo.lock
-rw-r--r-- 1 root          root            650 Aug 15 09:29 Cargo.toml
-rw-r--r-- 1 root          root           1258 Aug 15 13:08 .env
drwxr-xr-x 8 root          root           4096 Aug 15 09:29 .git
-rw-r--r-- 1 root          root             14 Aug 15 09:29 .gitignore
-rw-r--r-- 1 root          root            462 Aug 15 09:29 Makefile
-rw-r--r-- 1 root          root           2543 Aug 15 09:29 README.md
drwxr-xr-x 3 root          root           4096 Aug 15 09:29 src
root@aplha:/home/administrator/agent_listen#
```

ภาพที่ ก.49 ตัวอย่างขั้นตอนที่ 7 ของการติดตั้ง Script agent listen

8) พิมพ์ “make install” จากนั้นรอจนติดตั้ง library ของ rust language จนเสร็จ

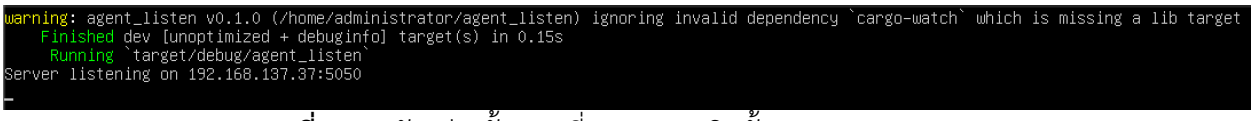
```
root@aplha:/home/administrator/agent_listen# make install
cargo add dotenv@0.15.0
    Updating crates.io index
      Adding dotenv v0.15.0 to dependencies.
             Features:
             - clap
             - cli
cargo add tokio@1.28.2 -F "full"
    Updating crates.io index
      Adding tokio v1.28.2 to dependencies.
```

ภาพที่ ก.50 ตัวอย่างขั้นตอนที่ 8 ของการติดตั้ง Script agent listen



ภาพที่ ก.51 ตัวอย่างขั้นตอนที่ 8 เพิ่มเติมของการติดตั้ง Script agent listen

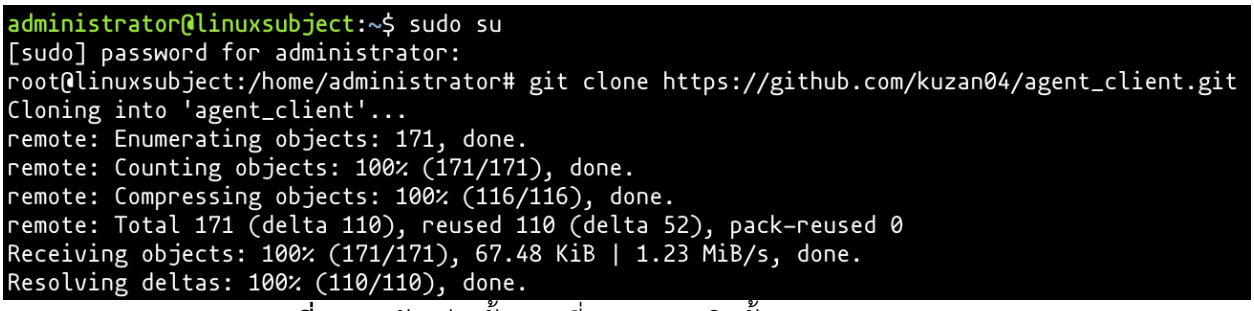
9) เมื่อเสร็จสิ้น พิมพ์ “make run” ทำงานครั้งแรก *จำเป็นต้องใช้อินเตอร์เน็ต* จากนั้นเมื่อแสดง “IP:PORT” ก็เป็นอันเสร็จสิ้น



ภาพที่ ก.52 ตัวอย่างขั้นตอนที่ 9 ของการติดตั้ง Script agent listen

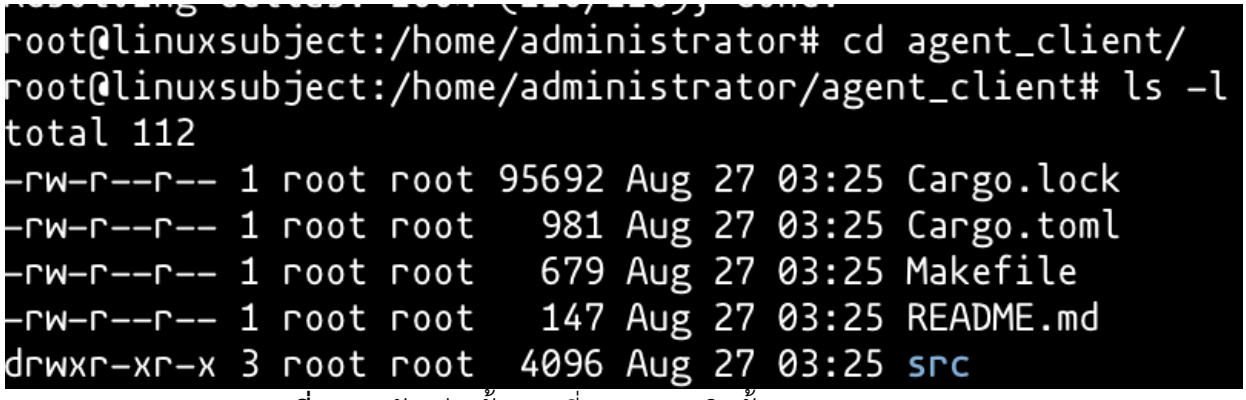
ติดตั้ง Script agent client

1) พิมพ์ “cd” เพื่อกลับมายังที่อยู่ของผู้ใช้งานก่อน พิมพ์ “sudo su” จากนั้นเมื่อยังไม่เคยใช้งานซุปเปอร์ผู้ใช้งานหรือใช้งานไปแล้วนานๆ ทำการกรอกรหัสผ่านของผู้ใช้งานที่ใช้งานอยู่ ขณะนั้น กดปุ่ม “Enter” ทำการพิมพ์ หรือ คัดลอก วาง “git clone https://github.com/kuzan04/agent_client.git” กดปุ่ม “Enter” รอจนเสร็จสิ้น



ภาพที่ ก.53 ตัวอย่างขั้นตอนที่ 1 ของการติดตั้ง Script agent client

2) พิมพ์ “cd ./agent_client” เพื่อเข้าไปยัง directory ของ agent listen จากนั้น พิมพ์ “ls -l” เพื่อทำการแสดงรายการไฟล์ต่างๆ



ภาพที่ ก.54 ตัวอย่างขั้นตอนที่ 2 ของการติดตั้ง Script agent client

3) พิมพ์ “make install” จากนั้นรอจนติดตั้ง library ของ rust language จนเสร็จ

```
root@linuxsubject:/home/administrator/agent_client# make install
cargo add dotenv@0.15.0
    Updating crates.io index
      Adding dotenv v0.15.0 to dependencies.
             Features:
             - clap
             - cli
    Updating crates.io index
       Fetch [=======================>         ] 325 complete; 18 pending
```

ภาพที่ ก.55 ตัวอย่างขั้นตอนที่ 3 ของการติดตั้ง Script agent client

```
  Compiling textwrap v0.11.0
  Compiling notify v4.0.17
  Compiling globset v0.4.6
  Compiling clearscreen v1.0.11
  Compiling derive_builder v0.10.2
  Compiling command-group v1.0.8
  Compiling thread_local v1.1.7
  Compiling cargo-platform v0.1.3
  Compiling vec_map v0.8.2
  Compiling termcolor v1.1.3
  Compiling ansi_term v0.12.1
  Compiling lazy_static v1.4.0
  Compiling strsim v0.8.0
  Compiling glob v0.3.1
  Compiling watchexec v1.17.2
  Compiling clap v2.34.0
  Compiling stderrlog v0.5.4
  Compiling cargo_metadata v0.15.4
  Compiling notify-rust v4.9.0
  Compiling shell-escape v0.1.5
  Compiling dotenvy v0.15.7
  Compiling cargo-watch v8.4.0
   Finished release [optimized] target(s) in 1m 58s
  Installing /root/.cargo/bin/cargo-watch
   Installed package `cargo-watch v8.4.0` (executable `cargo-watch`)
root@linuxsubject:/home/administrator/agent_client#
```

ภาพที่ ก.56 ตัวอย่างขั้นตอนที่ 3 เพิ่มเติมของการติดติดตั้ง Script agent client

4) พิมพ์ “nano ./src/main.rs” หรือ “vi ./src/main.rs” เพื่อเข้าไปแก้ไขไฟล์ และให้ค้นหาหรือสังเกตหา เพื่อทำการแก้ไขตัวแปรให้ถูกต้อง “env::set_var("DB_MAIN", "DOL_PDPA_LOCAL");” และอีกหนึ่งที่ต้องแก้ “let database_url = format!("mysql://root:P@ssw0rd@localhost:3306/{}"....” เป็นไปตามที่ต้องการ โดยอย่างแรกคือ ชื่อ ฐานข้อมูลหลัก และ อย่างที่สอง คือ IP ของฐานข้อมูลหลัก

```
root@linuxsubject:/home/administrator/agent_client# vi ./src/main.rs
```

ภาพที่ ก.57 ตัวอย่างขั้นตอนที่ 4 ของการติดตั้ง Script agent client

```rust
#[tokio::main]
async fn main() {
    dotenv().ok();

    if metadata(".env").is_err() {
        loop {
            print!("Please enter the token you have: ");
            io::stdout().flush().unwrap();

            let mut input = String::new();
            io::stdin().read_line(&mut input).expect("Failed to read input!");

            match !input.trim().is_empty() && general_purpose::STANDARD.decode(input.trim()).is_ok() {
                true => {
                    if let Ok(decoded_bytes) = general_purpose::STANDARD.decode(input.trim()) {
                        let base = String::from_utf8_lossy(&decoded_bytes).to_string();
                        let mut mix = base.split("&&&").collect::<Vec<&str>>();
                        mix.push(input.trim());
                        // function on test only!!
                        time_function(|| set_env(mix.to_vec()), "set_env");

                        break
                    } else {
                        println!("Invalid input. Value must be base64!");
                    }
                },
                false => println!("Invalid input. Please enter the token on base64"),
            }
        }
    }

    env::set_var("DB_MAIN", "DOL_PDPA_LOCAL");
    let database_url = format!("mysql://root:P@ssw0rd@localhost:3306/{}", env::var("DB_MAIN").unwrap_or_else(|_| "DOL_PDPA".to_string()));
    let pool = match MySqlPoolOptions::new()
        .max_connections(10)
        .connect(&database_url)
        .await {
            Ok(pool) => {
                pool
            }
            Err(e) => {
                println!("Failed to connect the database: {:?}", e);
                std::process::exit(1);
            }
        };

    // Main process.
    Handler::new(
        pool,
        dotenv::var("HOST").unwrap_or_else(|_| "127.0.0.1".to_string()),
        dotenv::var("PORT").unwrap_or_else(|_| 5050.to_string()),
    ).task().await;

    // Option send details.
    // dotenv::vars().collect::<HashMap<String, String>>()
}
```

ภาพที่ ก.58 ตัวอย่างขั้นตอนที่ 4 เพิ่มเติมของการติดตั้ง Script agent client

5) **เพิ่มเติม** กรณีเป็น script agent client ประเภทหรือชนิด “Sniffer” ทำการพิมพ์ “sudo nano ./src/module/handles.rs” หรือ “sudo vi ./src/module/handles.rs” เพื่อเข้าไปแก้ไขไฟล์ จากนั้นค้นหาหรือ สังเกตหา “let selected = “en0”.to_string();” แก้ไขให้เป็นอินเตอร์เฟสที่ต้องการ Sniffer

```
root@linuxsubject:/home/administrator/agent_client# vi ./src/module/handles.rs
```

ภาพที่ ก.59 ตัวอย่างขั้นตอนที่ 5 ของการติดตั้ง Script agent client

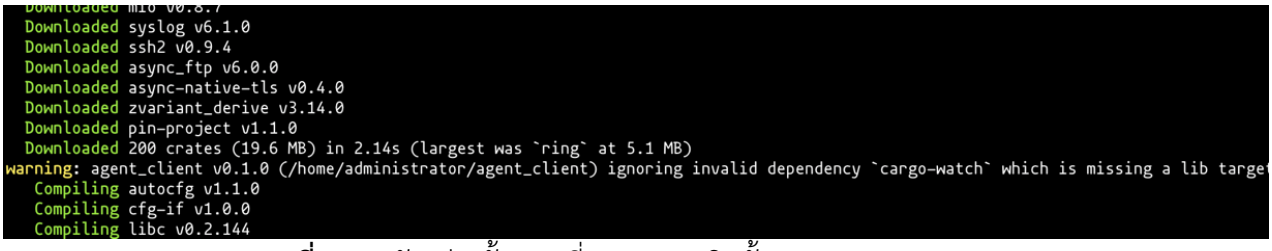
```
                // match DirectoryFile::new(
                //     mix,
                //     details,
                //     env::var("HOST").unwrap(),
                //     21,
                //     "ftpuser".to_string(),
                //     "ftpuser".to_string(),
                // ).build().await {
                //         Ok(result) => result,
                //         Err(err) => vec![format!("[Failed] {}", err)]
                //     }
                // function on test only!!
                let start = DirectoryFile::new(mix, details, env::var("HOST").unwrap(), 21, "ftpuser".to_string(), "ftpuser".to_string());
                match time_function(|| start.build(), "file_start").await {
                    Ok(result) => result,
                    Err(err) => vec![format!("[Failed] {}", err)]
                }
            },
            "AG3" => {
                let mut details = env::var("DETAILS").unwrap().split('&').map(|s| s.to_string()).collect::<Vec<String>>();
                let db_type = details.remove(0).parse::<i32>().unwrap_or(-1);
                // match DatabaseCheck::new(
                //     self.db.clone(),
                //     db_type,
                //     details,
                // ).build().await {
                //         Ok(result) => result,
                //         Err(err) => vec![format!("[Failed] {}", err)],
                //     }
                //  function on test only!!
                let start = DatabaseCheck::new(self.db.clone(), db_type, details);
                match time_function(|| start.build(), "database_start").await {
                    Ok(result) => result,
                    Err(err) => vec![format!("[Failed] {}", err)],
                }
            },
            "AG4" => {
                let selected = "en0".to_string();
                let details = env::var("DETAILS").unwrap().split(',').map(|s| s.to_string()).collect::<Vec<String>>();
                TaskSniffer::new(self.db.clone(), details, selected).run().await;
                vec!["Skip listener".to_string()]
            },
            _ => {
                println!("[Error] Type of agent client overdue!!!");
                process::exit(1);
            }
        }
    },
    Err(err) => {
        println!("[Error] {}", err);
        process::exit(1);
    }
  }
}

pub async fn task(&mut self) {
```

ภาพที่ ก.60 ตัวอย่างขั้นตอนที่ 5 เพิ่มเติมของการติดตั้ง Script agent client

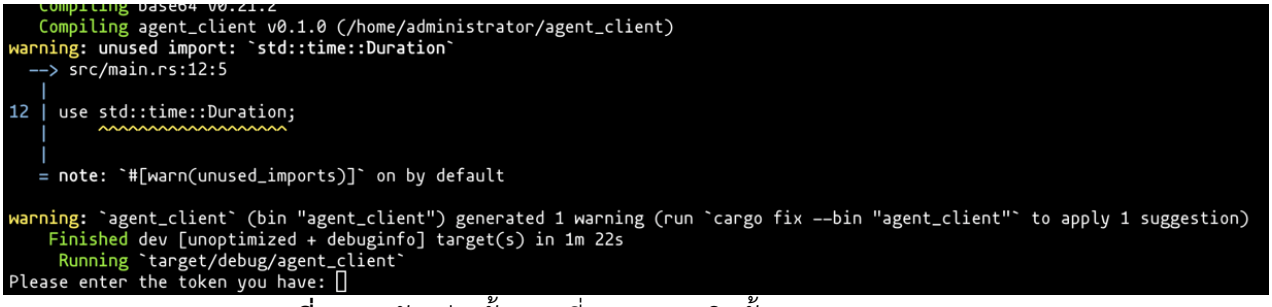
6) พิมพ์ “make run” เพื่อทำการทำงานระบบ agent client (ทำงานครั้งแรกอาจจะมีการดาวน์โหลดและจัดการไฟล์ต่างๆ) จากนั้น script จะต้องการ token

```
root@linuxsubject:/home/administrator/agent_client# make run
```

ภาพที่ ก.61 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent client

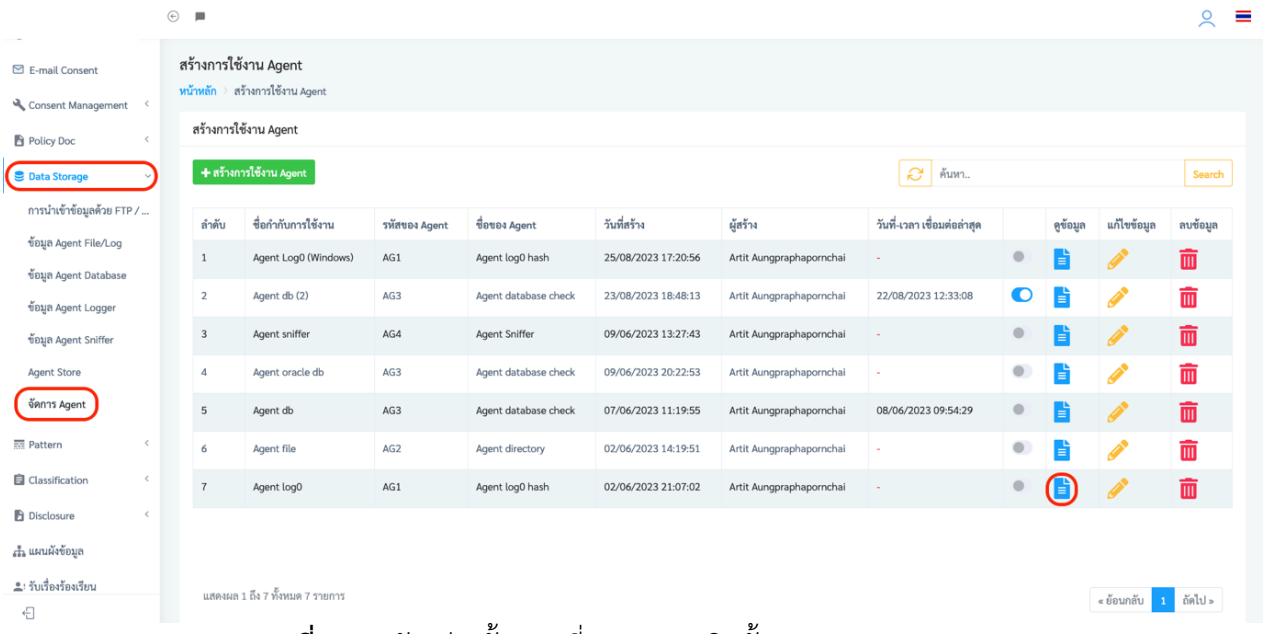


ภาพที่ ก.62 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent client



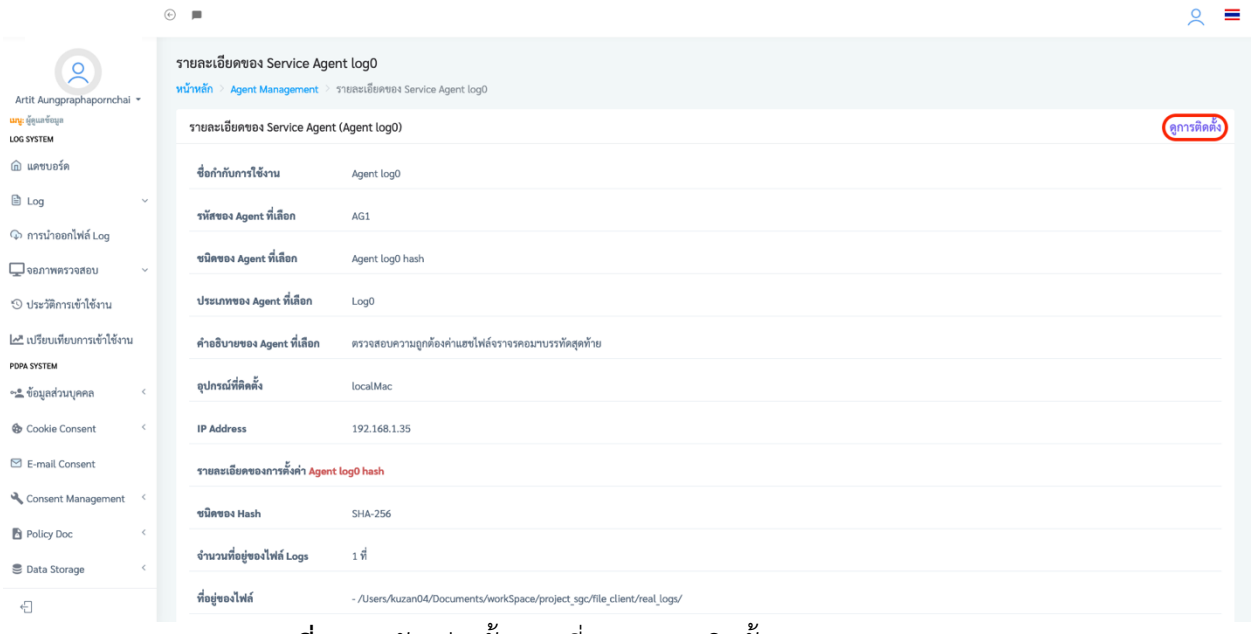
ภาพที่ ก.63 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent client

7) กลับเข้าไปยังเว็บแอพพลิเคชัน เมื่อเข้าไป คลิกแถบเมนูด้านซ้ายมือ “Data Store” จากนั้นคลิก “จัดการ Agent” คลิกรูป “เอกสารสีน้ำเงิน” เพื่อเข้าไปยังหน้ารายละเอียดของที่เลือกนั้นๆ



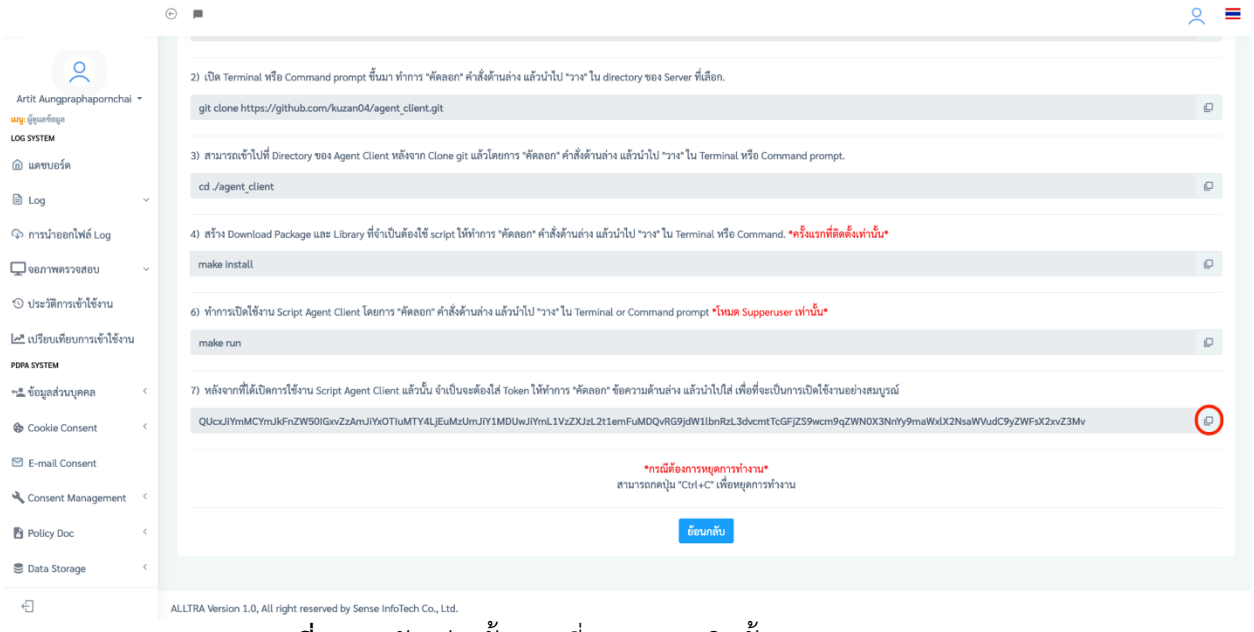
ภาพที่ ก.64 ตัวอย่างขั้นตอนที่ 7 ของการติดตั้ง Script agent client

8) ด้านมุมบน คลิก “ดูการติดตั้ง” เพื่อทำการเข้าไปคัดลอก “Token” เพื่อนำมาใช้งาน



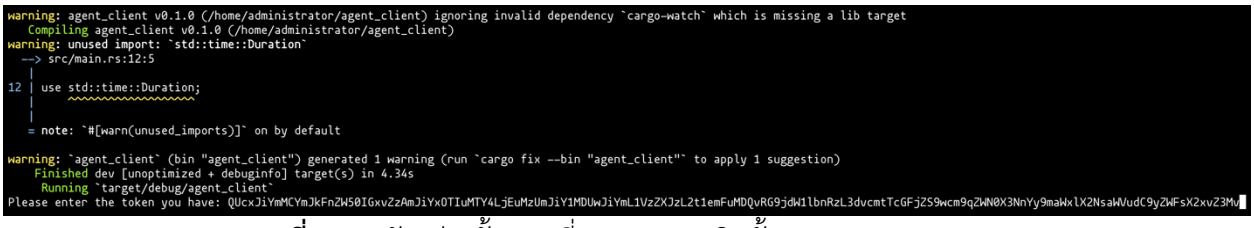
ภาพที่ ก.65 ตัวอย่างขั้นตอนที่ 8 ของการติดตั้ง Script agent client

9) ทำการเลื่อนลงไปยังล่างสุด คลิกรูป “สี่เหลี่ยมซ้อนกัน” เพื่อเป็นการคัดลอกข้อความด้านหน้านั้น



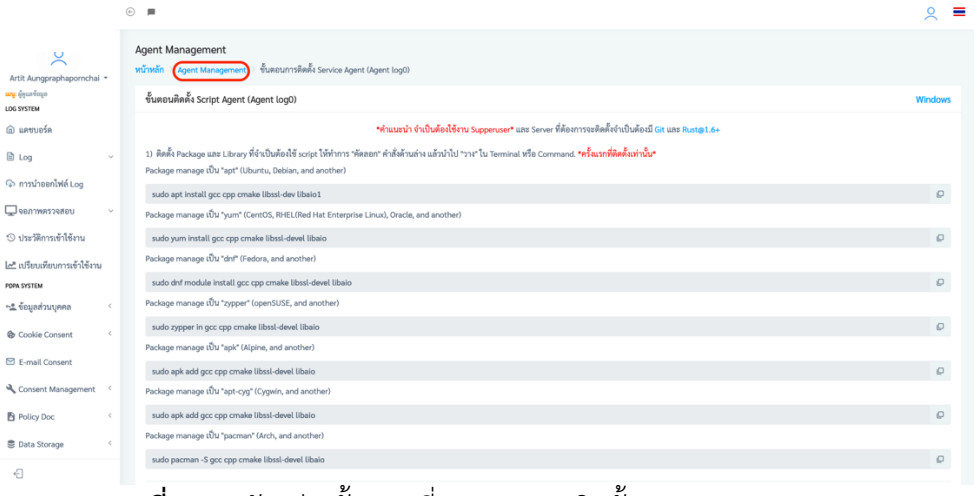
ภาพที่ ก.66 ตัวอย่างขั้นตอนที่ 9 ของการติดตั้ง Script agent client

10) ทำการนำข้อความที่ได้คัดลอกจากข้อด้านบน วางลงไป กดปุ่ม “Enter”



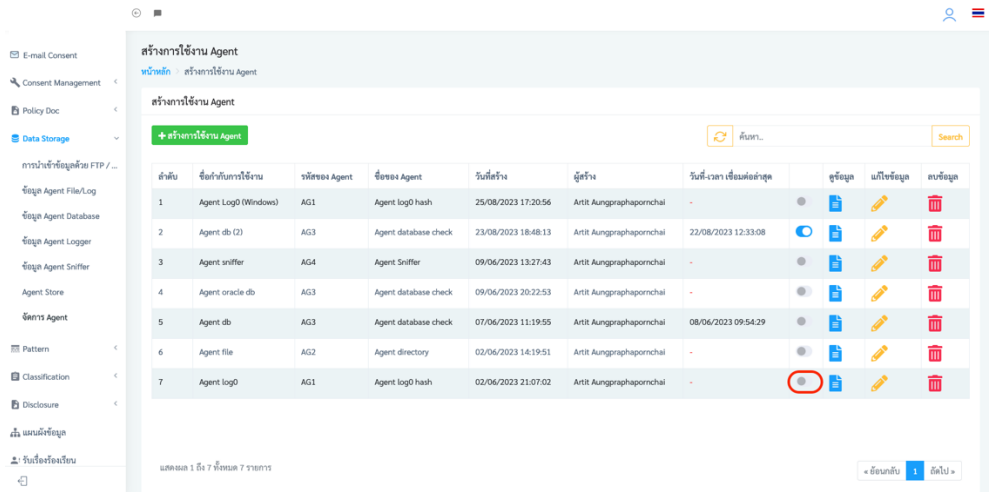
ภาพที่ ก.67 ตัวอย่างขั้นตอนที่ 10 ของการติดตั้ง Script agent client

11) กลับไปยังเว็บเบราว์เซอร์ จากนั้นทำการคลิก คำว่า “Agent Management” ด้านบนดังภาพด้านล่างนี้

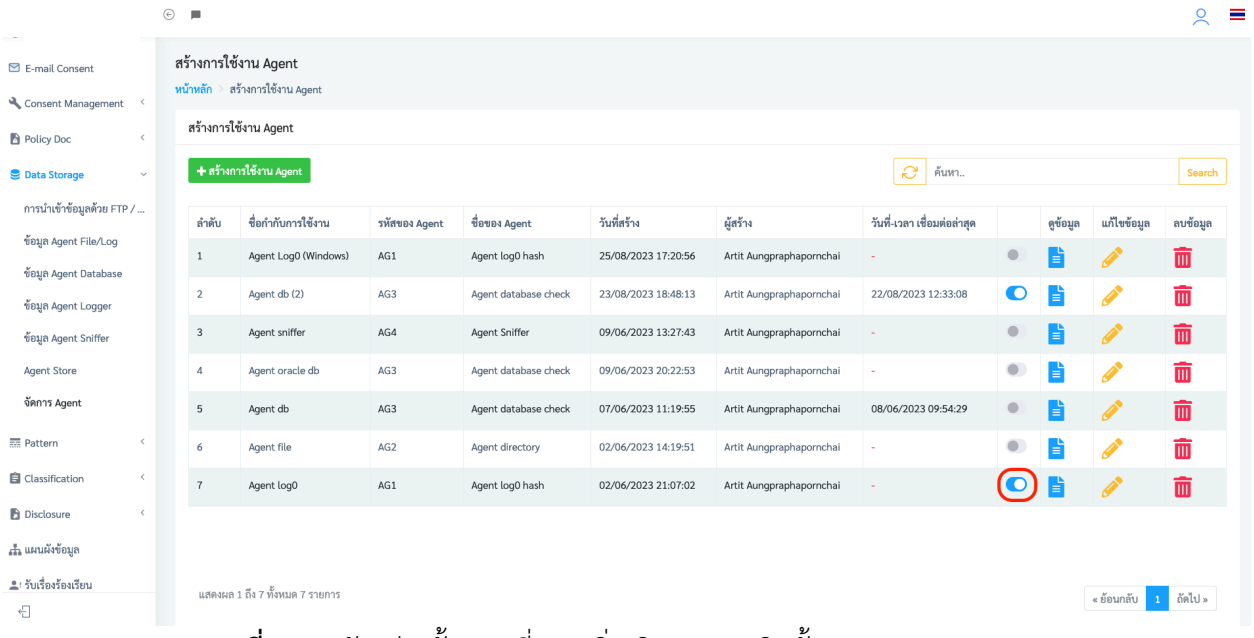


ภาพที่ ก.68 ตัวอย่างขั้นตอนที่ 11 ของการติดตั้ง Script agent client

12) จากนั้นคลิกปุ่มที่คล้ายกับสวิตช์เพื่อเปิดใช้งาน script นั้นๆ เป็นอันเสร็จสิ้น



ภาพที่ ก.69 ตัวอย่างขั้นตอนที่ 12 ของการติดตั้ง Script agent client



ภาพที่ ก.70 ตัวอย่างขั้นตอนที่ 12 เพิ่มเติมของการติดตั้ง Script agent client

คู่มือการติดตั้ง Software Agent listen & client เฉพาะ Oracle Linux 8.7 หรือใหม่กว่า

ติดตั้ง Script agent listen

1) พิมพ์ “cd” เพื่อกลับมายังที่อยู่ของผู้ใช้งานก่อน “sudo su” จากนั้นเมื่อยังไม่เคยใช้งานซุปเปอร์ผู้ใช้งานหรือใช้งานไปแล้วนานๆ ทำการกรอกรหัสผ่านของผู้ใช้งานที่ใช้งานอยู่ ขณะนั้น กดปุ่ม “Enter” พิมพ์ “cd ./Documents” เพื่อเข้าไปเก็บไฟล์และโฟลเดอร์ต่างๆ ในนั้น ทำการพิมพ์ หรือ คัดลอก วาง “git clone https://github.com/kuzan04/agent_listen.git” กดปุ่ม “Enter” รอจนเสร็จสิ้น

```
[test@localhost oracle]$ cd
[test@localhost ~]$ sudo su
[sudo] password for test:
[root@localhost test]# cd ./Documents/
[root@localhost Documents]# git clone https://github.com/kuzan04/agent_listen.git
Cloning into 'agent_listen'...
remote: Enumerating objects: 237, done.
remote: Counting objects: 100% (237/237), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 237 (delta 150), reused 163 (delta 81), pack-reused 0
Receiving objects: 100% (237/237), 62.72 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (150/150), done.
[root@localhost Documents]#
```

ภาพที่ ก.71 ตัวอย่างขั้นตอนที่ 1 ของการติดตั้ง Script agent listen

2) พิมพ์ “cd ./agent_listen/” เพื่อเข้าไปยัง directory ของ agent listen จากนั้น พิมพ์ “ls -l” เพื่อทำการแสดงรายการไฟล์ต่างๆ

```
[root@localhost Documents]# cd ./agent_listen/
[root@localhost agent_listen]# ls -l
total 96
-rw-r--r--. 1 root root 85314 Aug 27 15:01 Cargo.lock
-rw-r--r--. 1 root root   650 Aug 27 15:01 Cargo.toml
-rw-r--r--. 1 root root   462 Aug 27 15:01 Makefile
-rw-r--r--. 1 root root  2488 Aug 27 15:01 README.md
drwxr-xr-x. 3 root root    51 Aug 27 15:01 src
[root@localhost agent_listen]#
```

ภาพที่ ก.72 ตัวอย่างขั้นตอนที่ 2 ของการติดตั้ง Script agent listen

3) พิมพ์ “ifconfig” จากนั้นดูผลลัพธ์ในอินเตอร์เฟสเน็ตเวิร์คที่ต้องการเปิดใช้งาน script agent listen จดจำไว้ หรือคัดลอก ชื่อของอินเตอร์เฟสนั้นๆ

```
[root@localhost agent_listen]# ifconfig
enp0s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.137.3  netmask 255.255.255.0  broadcast 192.168.137.255
        inet6 fd9f:eaa2:1988:9421:50a8:3fff:fed4:8bf  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::50a8:3fff:fed4:8bf  prefixlen 64  scopeid 0x20<link>
        ether 52:a8:3f:d4:08:bf  txqueuelen 1000  (Ethernet)
        RX packets 1697374  bytes 2496598244 (2.3 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 130859  bytes 13172522 (12.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[root@localhost agent_listen]#
```

ภาพที่ ก.73 ตัวอย่างขั้นตอนที่ 3 ของการติดตั้ง Script agent listen

4) โดยจะใช้งาน file editor Vim หรือ Nano ก็สามารถเลือกได้โดย Vim พิมพ์ “vi .env” หรือ “nano .env” กดปุ่ม “Enter” เพื่อทำการสร้างไฟล์ .env ไว้ให้ script agent listen ใช้งาน (ตัวอย่างจะใช้เป็น Vim)

```
[root@localhost agent_listen]# vi .env
```

ภาพที่ ก.74 ตัวอย่างขั้นตอนที่ 4 ของการติดตั้ง Script agent listen

5) เมื่อเปิดไฟล์ .env เสร็จ ทำการเข้าไปยัง URL “https://github.com/kuzan04/agent_listen” เพื่อทำการคัดลอก วาง หรือ พิมพ์เหมือนกับตัวอย่างดังภาพด้านล่าง

Example

```
INTERFACE= ขาอินเตอร์เน็ตที่ต้องการเปิดใช้งาน
PORT= port ที่ต้องการให้ listener
DB_HOST= database ip host
DB_PORT= database port
DB_USER= database username
DB_PASSWORD= database password
DB_NAME= database name
# ========================================================================
# set path on file from client directory
# ========================================================================
SOURCINATION= ที่อยู่ของการจัดเก็บไฟล์ FTP in listener
DESTINATION_LOG= ที่อยู่หลังจากรับไฟล์ Log from FTP in lister
DESTINATION_DOC= ที่อยู่หลังจากรับไฟล์ Doc, CSV, XCSV from FTP in lister
# ========================================================================
# Table here for client logs hash0
# ========================================================================
TB_LOG0= ชื่อ:ตามโดย columns or fields ต่อกันด้วย ',' ไปจนถึงตัวสุดท้ายก็ไม่ต้องใส่
# ========================================================================
# Table here client directory/file *NOTE* must have column ID!!
# ========================================================================
TB_FILE= ชื่อ:ตามโดย columns or fields ต่อกันด้วย ',' ไปจนถึงตัวสุดท้ายก็ไม่ต้องใส่
# ========================================================================
# Table here client check database *NOTE* must have column ID!!
# ========================================================================
TB_DB_CHECK= ชื่อ:ตามโดย columns or fields ต่อกันด้วย ',' ไปจนถึงตัวสุดท้ายก็ไม่ต้องใส่
# ========================================================================
# ========================================================================
TB_HISTORY= ชื่อ:ตามโดย columns id ของ agent manager เท่านั้น
# ========================================================================
```

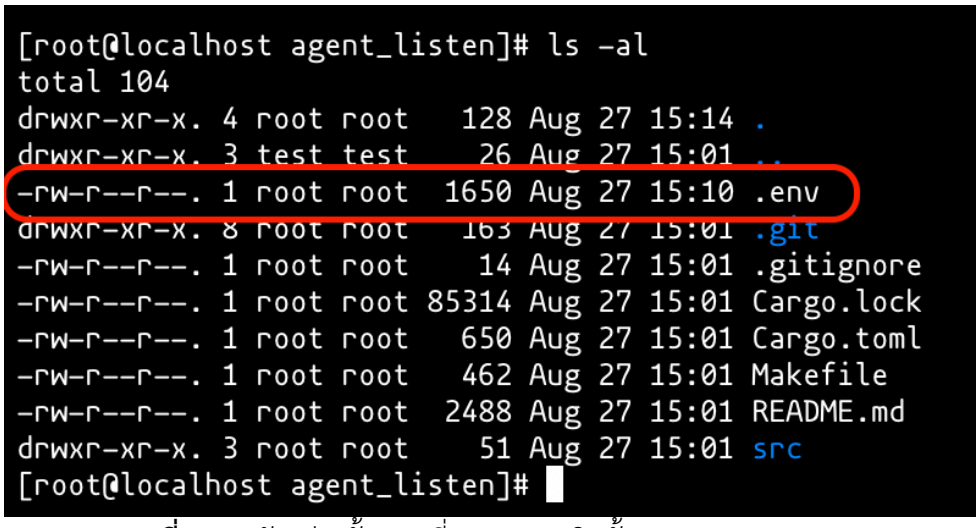
ภาพที่ ก.75 ตัวอย่างขั้นตอนที่ 5 ของการติดตั้ง Script agent listen

6) กลับไปยัง file editor ที่ได้เปิดไว้ ไม่ว่าจะเป็น Nano หรือ Vim จากนั้นทำการกรอกค่าที่เข้าไปยังไฟล์ .env ที่กำลังเปิดไว้อยู่ด้วยการพิมพ์ หรือ คัดลอกวาง (โดยตัวอย่างจะใช้เป็น Vim)

```
INTERFACE=enp0s1
PORT=5050
DB_HOST=127.0.0.1
DB_PORT=3306
DB_USER=root
DB_PASSWORD=Passw0rd
DB_NAME=DOL_PDPA
# ========================================================================
# set path on file from client directory
# ========================================================================
SOURCINATION=/home/ftpuser/
DESTINATION_LOG=/var/log/agent/
DESTINATION_DOC=/var/pdpa/agent/
# ========================================================================
# Table here for client logs hash0
# ========================================================================
TB_LOG0=TB_TR_PDPA_AGENT_LOG0_HASH:device_name,os_name,path,name_file,total_line,value,value_md5,value_sha1
# ========================================================================
# Table here client directory/file *NOTE* must have column ID!!
# ========================================================================
TB_FILE=TB_TR_PDPA_AGENT_FILE_DIR:id,device_name,os_name,path,name_file
# ========================================================================
# Table here client check database *NOTE* must have column ID!!
# ========================================================================
TB_DB_CHECK=TB_TR_PDPA_AGENT_DATABASE_CHECK:id,field_1,field_2,field_3,field_4,field_5,field_6,field_7,field_8,field_9,field_0,from_client
# ========================================================================
# Table here for add and update history client.
# ========================================================================
TB_HISTORY=TB_TR_PDPA_AGENT_LISTEN_HISTORY:agm_id
# ========================================================================
```

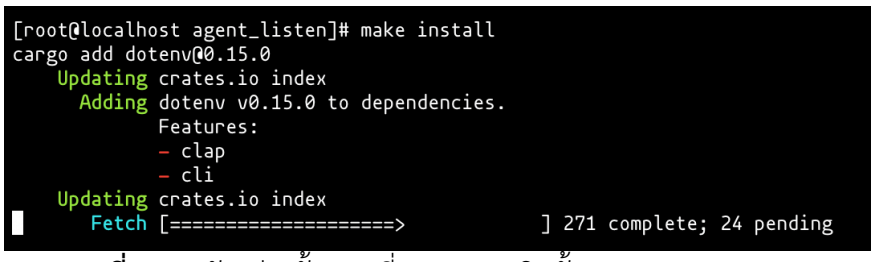
ภาพที่ ก.76 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent listen

7) ทำการบันทึกไฟล์ โดยถ้าเป็น Nano คือ กดปุ่ม “Ctrl+x” กดปุ่ม “y” กดปุ่ม “Enter” แต่ถ้าเป็น Vim พิมพ์ “:x” กดปุ่ม “Enter” จากนั้นตรวจสอบดูไฟล์ว่าบันทึกหรือไม่ พิมพ์ “ls -al”

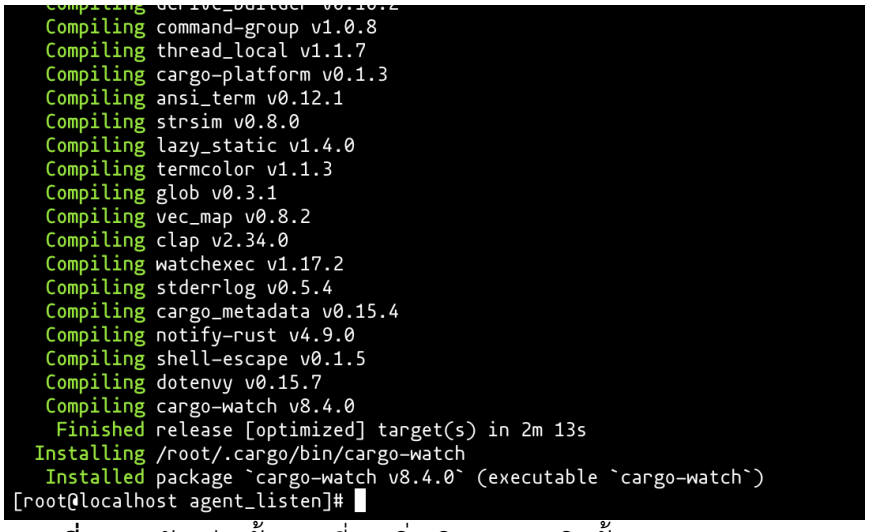


ภาพที่ ก.77 ตัวอย่างขั้นตอนที่ 7 ของการติดตั้ง Script agent listen

8) พิมพ์ “make install” จากนั้นรอจนติดตั้ง library ของ rust language จนเสร็จ

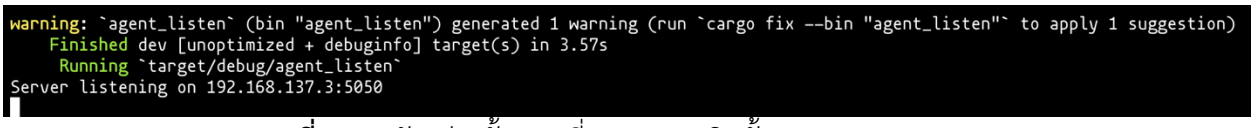


ภาพที่ ก.78 ตัวอย่างขั้นตอนที่ 8 ของการติดตั้ง Script agent listen



ภาพที่ ก.79 ตัวอย่างขั้นตอนที่ 8 เพิ่มเติมของการติดตั้ง Script agent listen

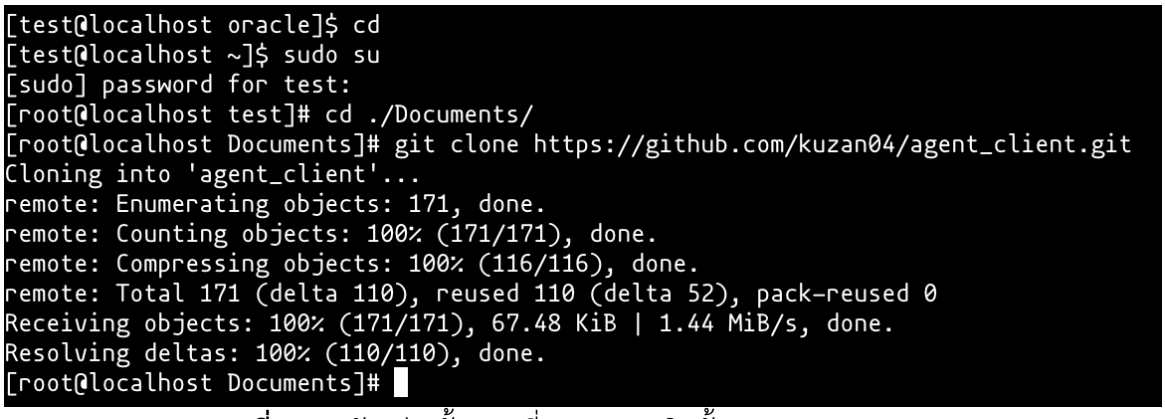
9) เมื่อเสร็จสิ้น พิมพ์ “make run” ทำงานครั้งแรก *จำเป็นต้องใช้อินเตอร์เน็ต* จากนั้นเมื่อแสดง “IP:PORT” ก็เป็นอันเสร็จสิ้น



ภาพที่ ก.80 ตัวอย่างขั้นตอนที่ 9 ของการติดตั้ง Script agent listen

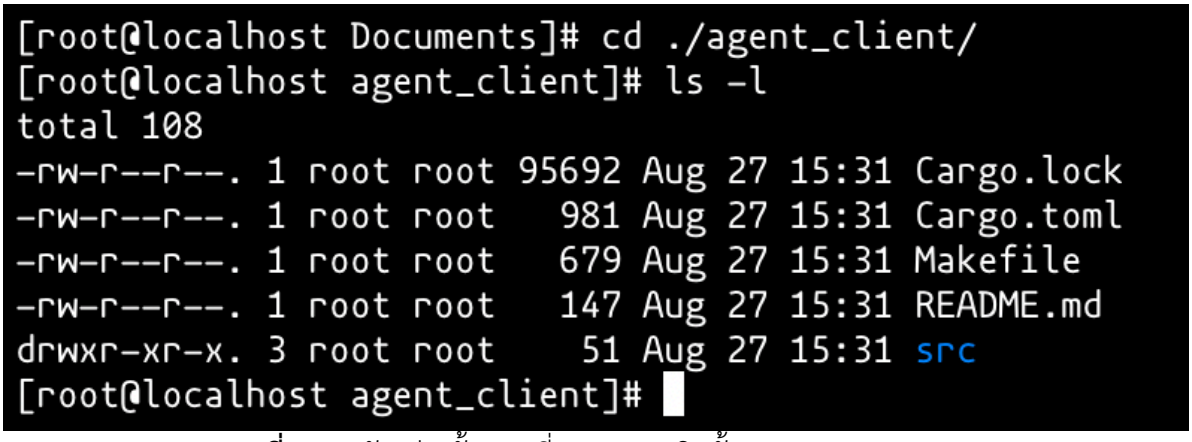
ติดตั้ง Script agent client

1) พิมพ์ “cd” เพื่อกลับมายังที่อยู่ของผู้ใช้งานก่อน “sudo su” จากนั้นเมื่อยังไม่เคยใช้งานซุปเปอร์ผู้ใช้งานหรือใช้งานไปแล้วนานๆ ทำการกรอกรหัสผ่านของผู้ใช้งานที่ใช้งานอยู่ ขณะนั้น กดปุ่ม “Enter” พิมพ์ “cd ./Documents” เพื่อเข้าไปเก็บไฟล์และโฟลเดอร์ต่างๆ ในนั้น ทำการพิมพ์ หรือ คัดลอก วาง “git clone https://github.com/kuzan04/agent_client.git” กดปุ่ม “Enter” รอจนเสร็จสิ้น



ภาพที่ ก.81 ตัวอย่างขั้นตอนที่ 1 ของการติดตั้ง Script agent client

2) พิมพ์ “cd ./agent_client/” เพื่อเข้าไปยัง directory ของ agent listen จากนั้น พิมพ์ “ls -l” เพื่อทำการแสดงรายการไฟล์ต่างๆ



ภาพที่ ก.82 ตัวอย่างขั้นตอนที่ 2 ของการติดตั้ง Script agent client

3) พิมพ์ “make install” จากนั้นรอจนติดตั้ง library ของ rust language จนเสร็จ

```
[root@localhost agent_client]# make install
cargo add dotenv@0.15.0
    Updating crates.io index
      Adding dotenv v0.15.0 to dependencies.
             Features:
             - clap
             - cli
    Updating crates.io index
cargo add tokio@1.28.2 -F "full"
    Updating crates.io index
      Adding tokio v1.28.2 to dependencies.
             Features:
```

ภาพที่ ก.83 ตัวอย่างขั้นตอนที่ 3 ของการติดตั้ง Script agent client

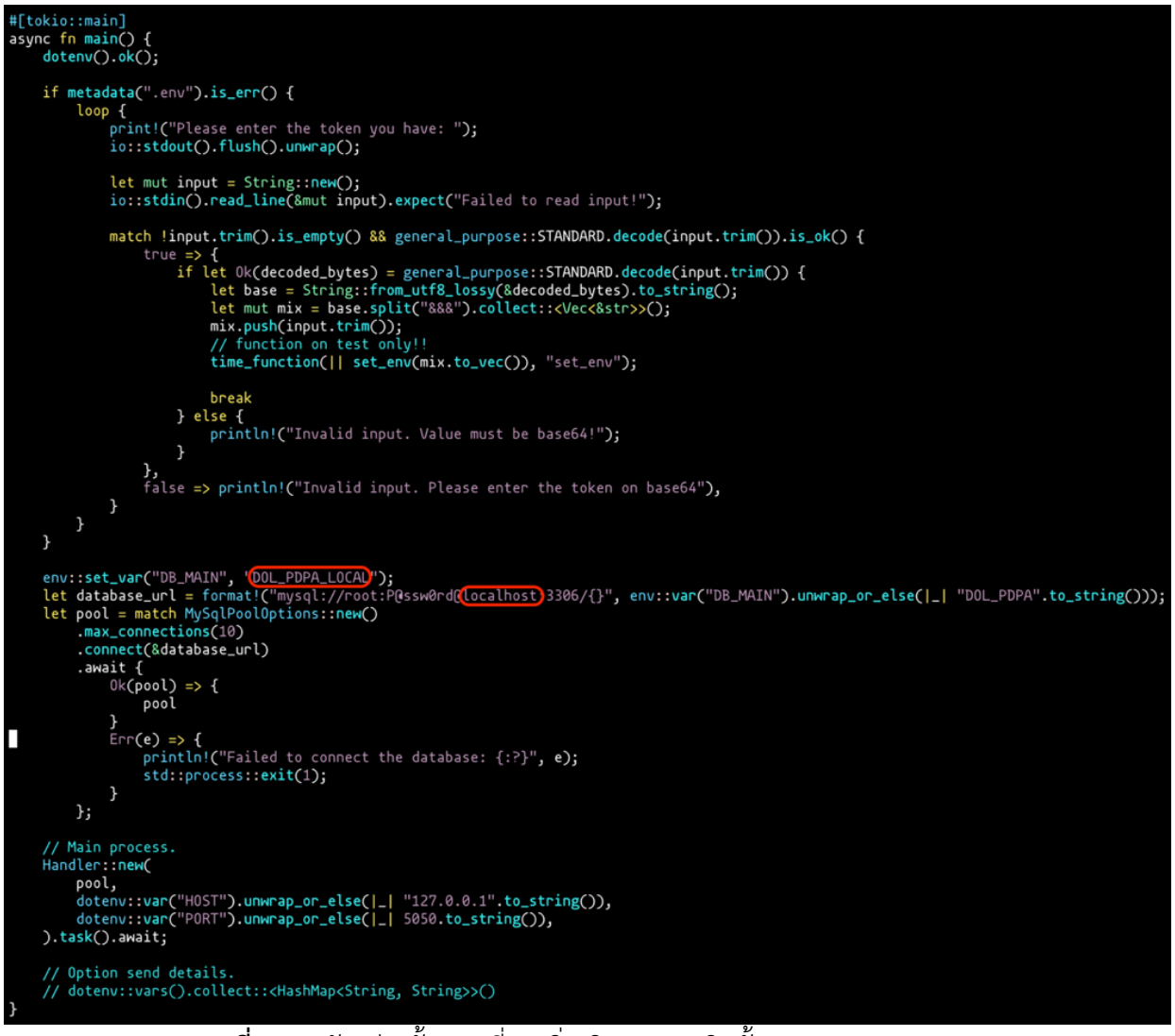
```
cargo add syslog@6.1.0
    Updating crates.io index
      Adding syslog v6.1.0 to dependencies.
cargo add sysinfo@0.29.7
    Updating crates.io index
      Adding sysinfo v0.29.7 to dependencies.
             Features:
             + multithread
             + rayon
             - apple-app-store
             - apple-sandbox
             - c-interface
             - debug
             - serde
             - unknown-ci
cargo install cargo-watch@8.4.0
    Ignored package `cargo-watch v8.4.0` is already installed, use --force to override
[root@localhost agent_client]#
```

ภาพที่ ก.84 ตัวอย่างขั้นตอนที่ 3 เพิ่มเติมของการติดตั้ง Script agent client

4) พิมพ์ “nano ./src/main.rs” หรือ “vi ./src/main.rs” เพื่อเข้าไปแก้ไขไฟล์ และให้ค้นหาหรือสังเกตหา เพื่อทำการแก้ไขตัวแปรให้ถูกต้อง “env::set_var("DB_MAIN", "DOL_PDPA_LOCAL");” และอีกหนึ่งที่ต้องแก้ “let database_url = format!("mysql://root:P@ssw0rd@localhost:3306/{}"....” เป็นไปตามที่ต้องการ โดยอย่างแรกคือ ชื่อ ฐานข้อมูลหลัก และ อย่างที่สอง คือ IP ของฐานข้อมูลหลัก

```
[root@localhost agent_client]# vi ./src/main.rs
```

ภาพที่ ก.85 ตัวอย่างขั้นตอนที่ 4 ของการติดตั้ง Script agent client



```rust
#[tokio::main]
async fn main() {
    dotenv().ok();

    if metadata(".env").is_err() {
        loop {
            print!("Please enter the token you have: ");
            io::stdout().flush().unwrap();

            let mut input = String::new();
            io::stdin().read_line(&mut input).expect("Failed to read input!");

            match !input.trim().is_empty() && general_purpose::STANDARD.decode(input.trim()).is_ok() {
                true => {
                    if let Ok(decoded_bytes) = general_purpose::STANDARD.decode(input.trim()) {
                        let base = String::from_utf8_lossy(&decoded_bytes).to_string();
                        let mut mix = base.split("&&&").collect::<Vec<&str>>();
                        mix.push(input.trim());
                        // function on test only!!
                        time_function(|| set_env(mix.to_vec()), "set_env");

                        break
                    } else {
                        println!("Invalid input. Value must be base64!");
                    }
                },
                false => println!("Invalid input. Please enter the token on base64"),
            }
        }
    }

    env::set_var("DB_MAIN", "DOL_PDPA_LOCAL");
    let database_url = format!("mysql://root:P@ssw0rd@localhost:3306/{}", env::var("DB_MAIN").unwrap_or_else(|_| "DOL_PDPA".to_string()));
    let pool = match MySqlPoolOptions::new()
        .max_connections(10)
        .connect(&database_url)
        .await {
            Ok(pool) => {
                pool
            }
            Err(e) => {
                println!("Failed to connect the database: {:?}", e);
                std::process::exit(1);
            }
        };

    // Main process.
    Handler::new(
        pool,
        dotenv::var("HOST").unwrap_or_else(|_| "127.0.0.1".to_string()),
        dotenv::var("PORT").unwrap_or_else(|_| 5050.to_string()),
    ).task().await;

    // Option send details.
    // dotenv::vars().collect::<HashMap<String, String>>()
}
```

ภาพที่ ก.86 ตัวอย่างขั้นตอนที่ 4 เพิ่มเติมของการติดตั้ง Script agent client

5) **เพิ่มเติม** กรณีเป็น script agent client ประเภทหรือชนิด “Sniffer” ทำการพิมพ์ “sudo su ./src/module/handles.rs” หรือ “sudo vi ./src/module/handles.rs” เพื่อเข้าไปแก้ไขไฟล์ จากนั้นค้นหา หรือ สังเกตหา “let selected = “en0”.to_string();” แก้ไขให้เป็นอินเตอร์เฟสที่ต้องการ Sniffer

```
[root@localhost agent_client]# vi ./src/module/handles.rs
```

ภาพที่ ก.87 ตัวอย่างขั้นตอนที่ 5 ของการติดตั้ง Script agent client

```
                    // match DirectoryFile::new(
                    //     mix,
                    //     details,
                    //     env::var("HOST").unwrap(),
                    //     21,
                    //     "ftpuser".to_string(),
                    //     "ftpuser".to_string(),
                    // ).build().await {
                    //         Ok(result) => result,
                    //         Err(err) => vec![format!("[Failed] {}", err)]
                    //     }
                    // function on test only!!
                    let start = DirectoryFile::new(mix, details, env::var("HOST").unwrap(), 21, "ftpuser".to_string(), "ftpuser".to_string());
                    match time_function(|| start.build(), "file_start").await {
                        Ok(result) => result,
                        Err(err) => vec![format!("[Failed] {}", err)]
                    }
                },
                "AG3" => {
                    let mut details = env::var("DETAILS").unwrap().split('&').map(|s| s.to_string()).collect::<Vec<String>>();
                    let db_type = details.remove(0).parse::<i32>().unwrap_or(-1);
                    // match DatabaseCheck::new(
                    //     self.db.clone(),
                    //     db_type,
                    //     details,
                    // ).build().await {
                    //         Ok(result) => result,
                    //         Err(err) => vec![format!("[Failed] {}", err)],
                    //     }
                    //  function on test only!!
                    let start = DatabaseCheck::new(self.db.clone(), db_type, details);
                    match time_function(|| start.build(), "database_start").await {
                        Ok(result) => result,
                        Err(err) => vec![format!("[Failed] {}", err)],
                    }
                },
                "AG4" => {
                    let selected = "en0".to_string();
                    let details = env::var("DETAILS").unwrap().split(',').map(|s| s.to_string()).collect::<Vec<String>>();
                    TaskSniffer::new(self.db.clone(), details, selected).run().await;
                    vec!["Skip listener".to_string()]
                },
                _ => {
                    println!("[Error] Type of agent client overdue!!!");
                    process::exit(1);
                }
            }
        },
        Err(err) => {
            println!("[Error] {}", err);
            process::exit(1);
        }
    }
}

pub async fn task(&mut self) {
```

ภาพที่ ก.88 ตัวอย่างขั้นตอนที่ 5 เพิ่มเติมของการติดตั้ง Script agent client

6) พิมพ์ “make run” เพื่อทำการทำงานระบบ agent client (ทำงานครั้งแรกอาจจะมีการดาวน์โหลดและจัดการไฟล์ต่างๆ) จากนั้น script จะต้องการ token

```
[root@localhost agent_client]# make run
```

ภาพที่ ก.89 ตัวอย่างขั้นตอนที่ 6 ของการติดตั้ง Script agent client

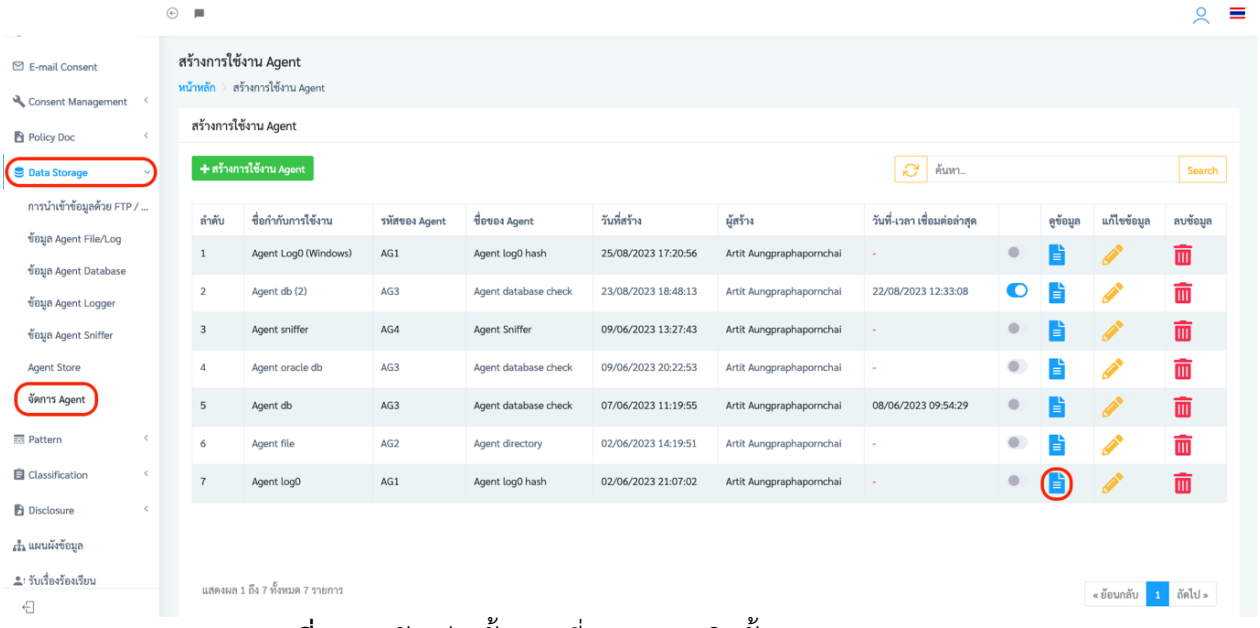
```
warning: agent_client v0.1.0 (/home/test/Documents/agent_client) ignoring invalid dependency `cargo-watch` which is missing a lib target
   Compiling syn v2.0.18
   Compiling libc v0.2.144
   Compiling futures-core v0.3.28
   Compiling value-bag v1.4.0
   Compiling crossbeam-utils v0.8.15
   Compiling once_cell v1.17.1
```

ภาพที่ ก.90 ตัวอย่างขั้นตอนที่ 6 เพิ่มเติมของการติดตั้ง Script agent client

```
   Compiling md5 v0.7.0
   Compiling agent_client v0.1.0 (/home/test/Documents/agent_client)
warning: unused import: `std::time::Duration`
  --> src/main.rs:12:5
   |
12 | use std::time::Duration;
   |     ^^^^^^^^^^^^^^^^^^^
   |
   = note: `#[warn(unused_imports)]` on by default

warning: `agent_client` (bin "agent_client") generated 1 warning (run `cargo fix --bin "agent_client"` to apply 1 suggestion)
    Finished dev [unoptimized + debuginfo] target(s) in 1m 16s
     Running `target/debug/agent_client`
Please enter the token you have:
```

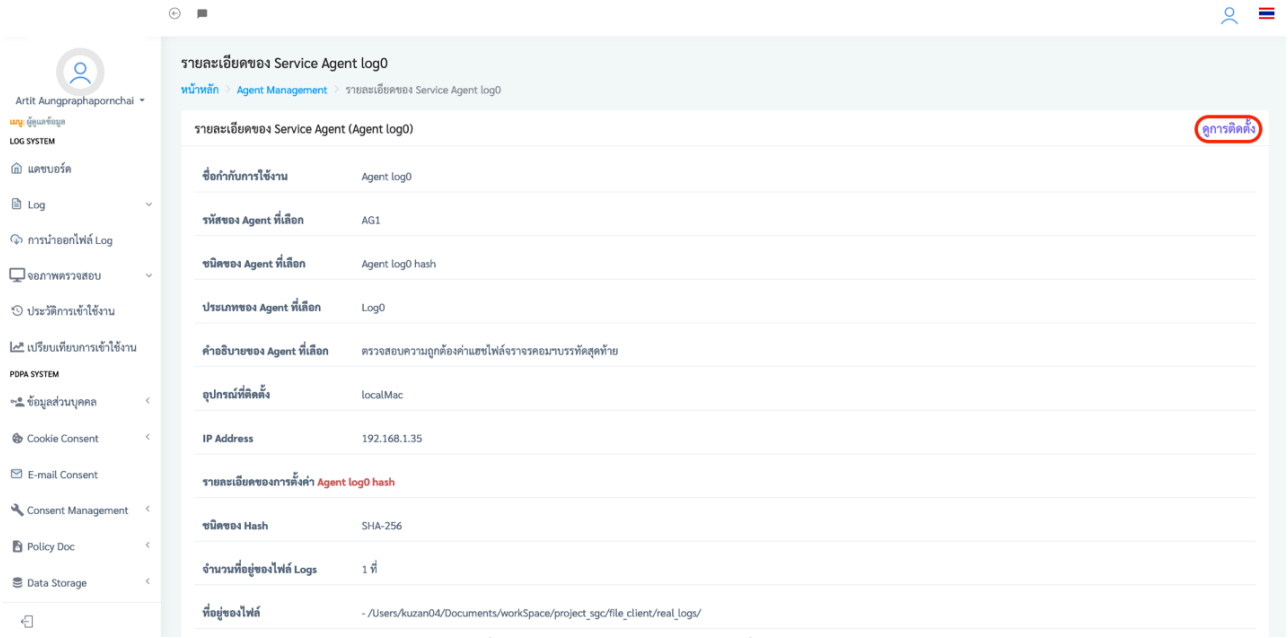
ภาพที่ ก.91 ตัวอย่างขั้นตอนที่ 6 เพิ่มเติมของการติดตั้ง Script agnet client

7) กลับเข้าไปยังเว็บแอพพลิเคชัน เมื่อเข้าไปคลิก คลิกแถบเมนูด้านซ้ายมือ “Data Store” จากนั้นคลิก “จัดการ Agent” คลิกรูป “เอกสารสีน้ำเงิน” เพื่อเข้าไปยังหน้ารายละเอียดของที่เลือกนั้นๆ



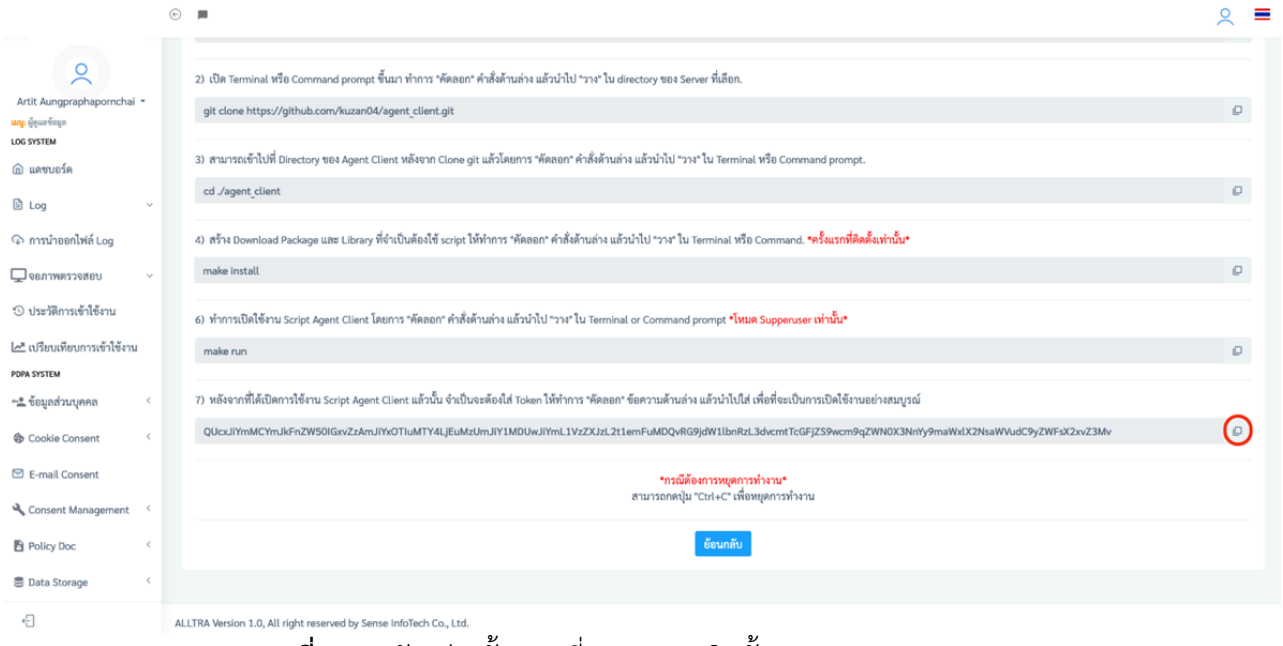
ภาพที่ ก.92 ตัวอย่างขั้นตอนที่ 7 ของการติดตั้ง Script agent client

8) ด้านมุมบน คลิก “ดูการติดตั้ง” เพื่อทำการเข้าไปคัดลอก “Token” เพื่อนำไปใช้งาน



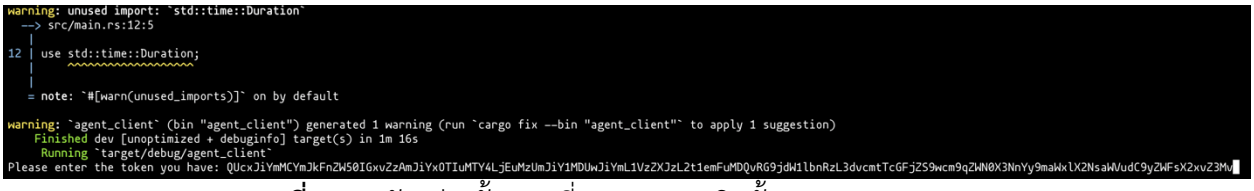
ภาพที่ ก.93 ตัวอย่างขั้นตอนที่ 8 ของการติดตั้ง Script agent client

9) ทำทำการเลื่อนลงไปยังล่างสุด คลิกรูป “สี่เหลี่ยมซ้อนกัน” เพื่อเป็นการคัดลอกข้อความด้านหน้านั้น



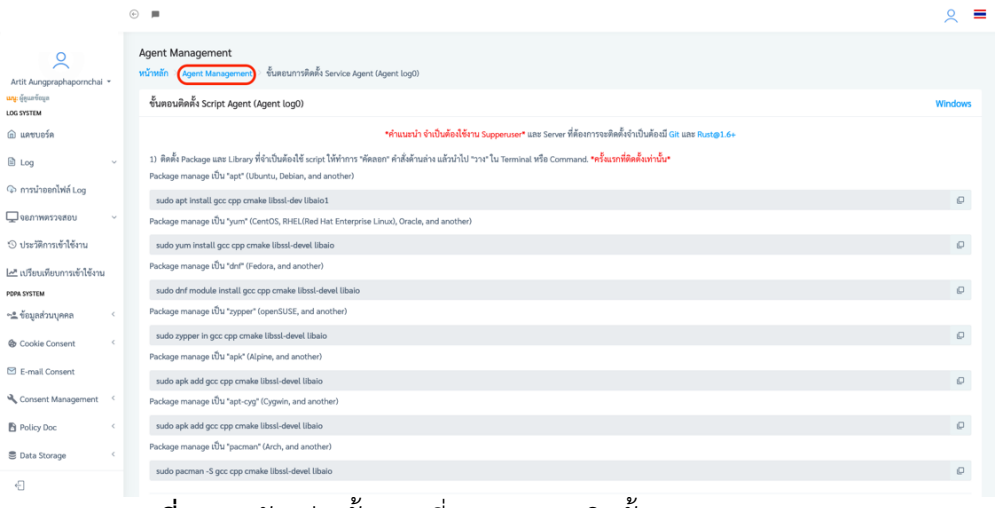
ภาพที่ ก.94 ตัวอย่างขั้นตอนที่ 9 ของการติดตั้ง Script agent client

10) ทำการนำข้อความที่ได้คัดลอกจากข้อด้านบน วางลงไป กดปุ่ม “Enter”



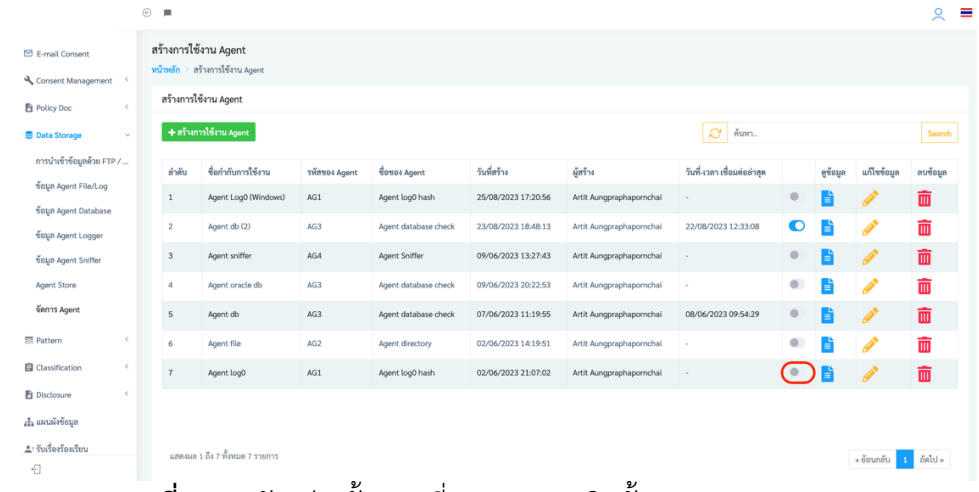
ภาพที่ ก.95 ตัวอย่างขั้นตอนที่ 10 ของการติดตั้ง Script agent client

11) กลับไปยังเว็บเบราว์เซอร์ จากนั้นทำการคลิก คำว่า “Agent Management” ด้านบนดังภาพด้านล่างนี้

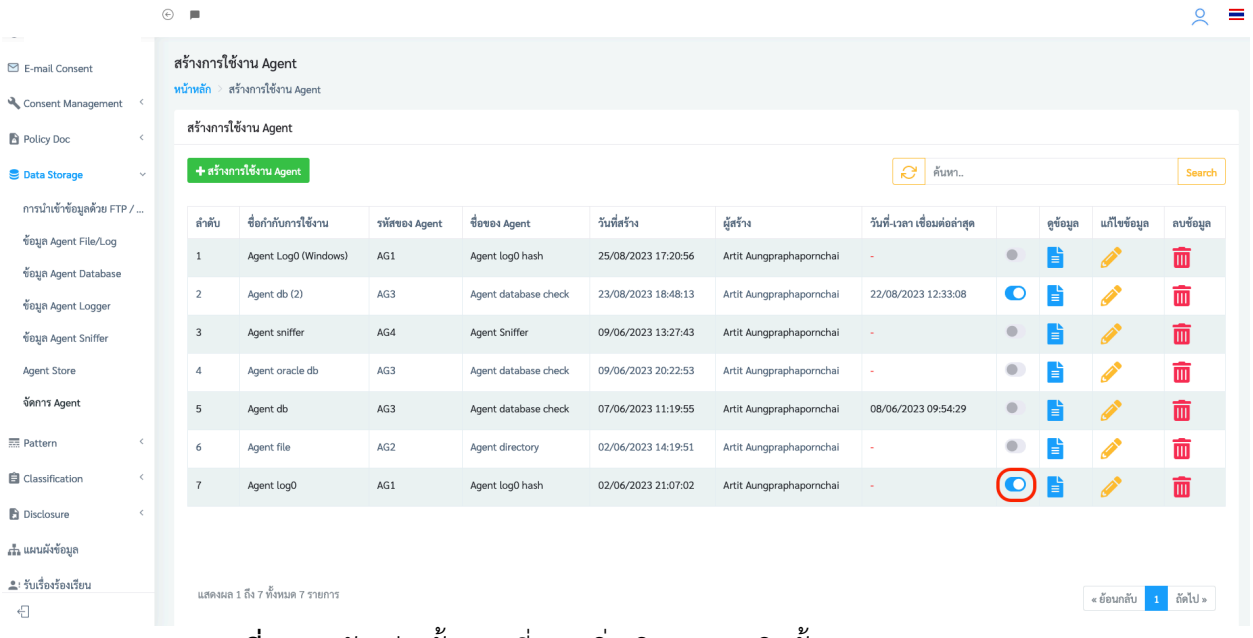


ภาพที่ ก.96 ตัวอย่างขั้นตอนที่ 11 ของการติดตั้ง Script agent client

12) จากนั้นคลิกปุ่มที่คล้ายกับสวิตช์เพื่อเปิดใช้งาน script นั้นๆ เป็นอันเสร็จสิ้น



ภาพที่ ก.97 ตัวอย่างขั้นตอนที่ 12 ของการติดตั้ง Script agent client



ภาพที่ ก.98 ตัวอย่างขั้นตอนที่ 12 เพิ่มเติมของการติดตั้ง Script agent client

ภาคผนวก ข

รายละเอียดชุดข้อมูลโค้ดระบบระบบตรวจสอบและการนำเข้าข้อมูลจราจรคอมพิวเตอร์และข้อมูลส่วนบุคคล

รายละเอียชุดข้อมูลโค้ดในส่วนหลักของแม่ข่าย

```
extern crate dotenv;

use dotenv::dotenv;
use get_if_addrs::get_if_addrs;
// use sqlx::mysql::MySqlPoolOptions;
// use sqlx::mysql::MySqlPool;
use std::fs;
use std::thread;
use std::time::Duration;

mod module;
mod model;
use crate::module::listen::Recevie;



// Use test only!
// use crate::module::test::*;

fn set_init() -> String {
    match fs::metadata(".env").is_ok() {
        true => "Success".to_string(),
        false => {
            println!("[Error] Please check file .env");
            std::process::exit(1);
        }
    }
}

fn get_ip(name: String) -> String {
    let mut ip = String::new();
    if let Ok(interfaces) = get_if_addrs() {
        for interface in interfaces {
            if !interface.is_loopback() && interface.name == name && interface.ip().is_ipv4() {
                ip =  interface.ip().to_string();
                break
            } else {
                ip = "None".to_string();
            }
        }
    } else {
        ip = "Failed".to_string();
    }
    ip
```

```
}

#[allow(unused_must_use)]
#[tokio::main]
async fn main() {
    // Get varriable from file .env.
    dotenv().ok();
    // Check file .env
    set_init();
    // function on test only!!!
    // time_function(|| set_init, "set_init");
    // Check and create tls to use socket.
    let ip: String;
    loop {
        let i = get_ip(dotenv::var("INTERFACE").unwrap_or_else(|_| "ens192".to_string()));
        // function on test only!!
        // let i = time_function(|| get_ip(dotenv::var("INTERFACE").unwrap_or_else(|_| "ens192".to_string())), "get_ip");
        match i.to_owned().as_str() {
            "None" => {
                println!("[Warning] Unknow ip from interfaces wait for 15 seconds script rebooting.");
                thread::sleep(Duration::from_secs(15));
            },
            "Failed" => {
                println!("Failed to retrieve interface addresses");
            },
            _ => {
                break ip = i
            },
        }
    }
    // Main database to use.
    let database_url = format!("mysql://{}:{}@{}:{}/{}",
        dotenv::var("DB_USER").unwrap_or_else(|_| "root".to_string()),
        dotenv::var("DB_PASSWORD").unwrap_or_else(|_| "P@ssw0rd".to_string()),
        dotenv::var("DB_HOST").unwrap_or_else(|_| "127.0.0.1".to_string()),
        dotenv::var("DB_DB_PORT").unwrap_or_else(|_| "3306".to_string()),
        dotenv::var("DB_NAME").unwrap_or_else(|_| "DOL_PDPA".to_string()),
    );
    // Create connect pool.
    // let pool = match MySqlPoolOptions::new()
    //     .connect(&database_url)
    //     .await {
    //         Ok(pool) => {
    //             pool
```

```
    //         }
    //         Err(e) => {
    //             println!("Failed to connect the database: {:?}", e);
    //             std::process::exit(1);
    //         }
    //     };

    // Start listener socket.
    // Recevie::new(ip, dotenv::var("PORT").unwrap_or_else(|_| 5050.to_string())).listen(pool).await;
    Recevie::new(ip, dotenv::var("PORT").unwrap_or_else(|_| 5050.to_string())).listen(database_url).await;
}
```

รายละเอียดชุดข้อมูลโค้ดส่วนโมเดลของแม่ข่าย

```
use serde::{Deserialize, Serialize};
use sqlx::{FromRow, mysql::MySqlRow, Row};

#[derive(Debug, Deserialize, Serialize)]
#[allow(non_snake_case)]
pub struct AgentStore {
  name: String,
  pub code: String,
  _type_: String,
  _limit_: i32,
  pub status: i32,
  hide: i32,
}
impl FromRow<'_, MySqlRow> for AgentStore {
  fn from_row(row: &MySqlRow) -> Result<Self, sqlx::Error> {
    let name: String = row.try_get("name")?;
    let code: String = row.try_get("code")?;
    let _type_: String = row.try_get("type")?;
    let _limit_: i32 = row.try_get("_limit_")?;
    let status: i32 = row.try_get("status")?;
    let hide: i32 = row.try_get("hide")?;

    Ok(Self { name, code, _type_, _limit_, status, hide})
  }
}

#[derive(Debug, Deserialize, Serialize, Clone, PartialEq)]
#[allow(non_snake_case)]
pub struct AgentManage {
  pub agm_id: i32,
  pub agm_name: String,
```

```
  pub code: String,
}
impl FromRow<'_, MySqlRow> for AgentManage {
  fn from_row(row: &MySqlRow) -> Result<Self, sqlx::Error> {
    let agm_id: i32 = row.try_get("agm_id")?;
    let agm_name: String = row.try_get("agm_name")?;
    let code: String = row.try_get("code")?;

    Ok(Self{ agm_id, agm_name, code })
  }
}
impl Default for AgentManage {
    fn default() -> Self {
        Self { agm_id: -1, agm_name: "NULL".to_string(), code: "NULL".to_string() }
    }
}

#[derive(Debug, Deserialize, Serialize)]
#[allow(non_snake_case)]
pub struct AgentHistory {
  pub agm_id: i32,
}

impl FromRow<'_, MySqlRow> for AgentHistory {
  fn from_row(row: &MySqlRow) -> Result<Self, sqlx::Error> {
    let agm_id: i32 = row.try_get("agm_id")?;
    Ok(Self{ agm_id })
  }
}
```

รายละเอียดชุดข้อมูลโค้ดในส่วนเชื่อมต่อลูกข่ายของแม่ข่าย

```
use tokio::io::{AsyncReadExt, AsyncWriteExt};
use tokio::net;
use sqlx::{MySqlPool, FromRow};

// use std::time::Duration;

use crate::module::{
    log0::LogHash,
    file::FileDirectory,
    db::{TestConnect, DatabaseCheck}
};

use crate::model::{AgentStore, AgentManage, AgentHistory};
```

```
// On test
// use crate::module::test::*;

#[derive(Debug)]
pub struct Recevie {
    pub host: String,
    pub port: String,
}

impl Recevie {
    pub fn new(host: String, port: String) -> Recevie {
        Recevie { host, port }
    }

    fn split_string(input: &str, delimiter: &str) -> Vec<String> {
        input.split(delimiter)
            .map(|s| s.to_string())
            .collect()
    }

    async fn test_connect(_type: String, host: String, user: String, passwd: String, database: String) -> String {
        let success: String;
        match _type.parse::<i32>() {
            Ok(t) => {
                let to_start = TestConnect::new(host, user, passwd, database);
                match t {
                    1 => {
                        if let Ok(res) = to_start.mysql().await {
                            success = res
                        } else {
                            success = "Query details table error.".to_string()
                        }
                    // function on test only!!
                        // if let Ok(res) = time_function(|| to_start.mysql(), "test_connect_mysql").await {
                        //     success = res
                        // } else {
                        //     success = "Query details table error.".to_string()
                        // }
                    },
                    0 => {
                        if let Ok(res) = to_start.oracle().await {
                            success = res
                        } else {
```

```
                    success = "Query details table error.".to_string()
                }
                // if let Ok(res) = time_function(|| to_start.oracle(), "test_connect_oracle").await {
                //     success = res
                // } else {
                //     success = "Query details table error.".to_string()
                // }
            },
            _ => success = "Hello from Server!".to_string(),
        };
    }
    Err(_) => {
        success = "Failed to read type from client!".to_string()
    }
};
success
}

fn status_store(query: Vec<AgentStore>, select: String) -> bool {
    let mut result = false;
    let mut i = 0;
    while i != query.len() {
        match &query[i] {
            a if a.code == select => {
                if a.status == 1 {
                    result = true
                }
                i += 1
            },
            _ => i += 1,
        }
    }
    result
}

fn set_manage(manager: Vec<AgentManage>, code: String, name: String) -> Result<AgentManage, String> {
    let mut i = 0;
    let mut selected = AgentManage::default();
    while i < manager.len() {
        if manager[i].agm_name == name && manager[i].code == code {
            selected = manager[i].clone();
        }
        i += 1
    }
```

```
        if selected == AgentManage::default() {
            Err("Not Found".to_string())
        } else{
            Ok(selected)
        }
    }

    async fn set_history(db: MySqlPool, manager: Vec<AgentManage>, code: String, name: String) -> String {
        let env_history = Self::split_string(&dotenv::var("TB_HISTORY").unwrap_or_else(|_|
"TB_TR_PDPA_AGENT_LISTEN_HISTORY:agm_id".to_string()), ":");
        // function on test only!!
        // let env_history = time_function(|| Self::split_string(&dotenv::var("TB_HISTORY").unwrap_or_else(|_|
"TB_TR_PDPA_AGENT_LISTEN_HISTORY:agm_id".to_string()), ":"), "split_string#10");
        let history: Vec<AgentHistory> = sqlx::query(
            format!(
                "SELECT {} FROM {} GROUP BY {}",
                env_history[1].clone(),
                env_history[0].clone(),
                env_history[1].clone()
            ).as_str())
            .fetch_all(&db)
            .await.unwrap()
            .into_iter()
            .map(|row| AgentHistory::from_row(&row).unwrap())
            .collect();
        let selected = Self::set_manage(manager, code, name);
        // function on test only!!
        // let selected = time_function(|| Self::set_manage(manager, code, name), "set_manage");
        match selected {
            Ok(agm) => {
                let mut message = String::new(); //
                let mut i = 0;
                while i < history.len() {
                    if history[i].agm_id == agm.agm_id {
                        sqlx::query(format!("UPDATE {} SET _get_ = NOW() WHERE {} = ?", env_history[0], env_history[1]).as_str())
                            .bind(agm.agm_id)
                            .execute(&db)
                            .await.unwrap();
                        message = "Success".to_string()
                    }
                    i += 1
                }
                match message.chars().count() {
                    0 => {
```

```
                    sqlx::query(format!("INSERT INTO {} ({}) VALUES (?)", env_history[0], env_history[1]).as_str())
                        .bind(agm.agm_id)
                        .execute(&db)
                        .await.unwrap();
                    "Success".to_string()
                }
                _ => "Success".to_string()
            }
        },
        Err(err) => format!("[Error] {} agent client from web alltra", err)
    }
}

async fn main_task(status: bool, details: Vec<String>, db: MySqlPool) -> String {
    match status {
        true => {
            match details[0].as_str() {
                "AG1" => {
                    let mut log0_table_all = Self::split_string(
                        &dotenv::var("TB_LOG0").unwrap_or_else(|_| "TB_TR_PDPA_AGENT_LOG0_HASH:device_name, os_name, path, name_file,
total_line, value, value_md5, value_sha1".to_string()),
                        ":"
                    );
                    let log0_columns = Self::split_string(&log0_table_all.pop().unwrap(), ",");
                    let log0_table = log0_table_all.pop().unwrap();
                    let content = Self::split_string(&details[details.len() - 1], "|||");
                    match LogHash::new(db, log0_table, log0_columns, content).build().await {
                        Ok(s) => s,
                        Err(e) => e.to_string(),
                    }

                    // function on test only!!
                    // let mut log0_table_all = time_function(|| Self::split_string(
                    //     &dotenv::var("TB_LOG0").unwrap_or_else(|_| "TB_TR_PDPA_AGENT_LOG0_HASH:device_name, os_name, path,
name_file, total_line, value, value_md5, value_sha1".to_string()),
                    //     ":"
                    // ), "split_string#1");
                    // let log0_columns = time_function(|| Self::split_string(&log0_table_all.pop().unwrap(), ","), "split_string#2");
                    // let log0_table = log0_table_all.pop().unwrap();
                    // let content = time_function(|| Self::split_string(&details[details.len() - 1], "|||"), "split_string#3");
                    // let mut start = LogHash::new(db, log0_table, log0_columns, content);
                    // match time_function(|| start.build(), "main_log0").await {
                    //     Ok(s) => s,
                    //     Err(e) => e.to_string(),
```

```
                // }
            },
            "AG2" => {
                let mut file_table_all = Self::split_string(
                    &dotenv::var("TB_FILE").unwrap_or_else(|_| "TB_TR_PDPA_AGENT_FILE_DIR:id, device_name, os_name, path, name_file,
size".to_string()),
                    ":"
                );
                let file_columns = Self::split_string(&file_table_all.pop().unwrap(), ",");
                let file_table = file_table_all.pop().unwrap();
                let content = Self::split_string(&details[details.len() - 1], "|||");

                match FileDirectory::new(
                    db,
                    file_table,
                    file_columns,
                    dotenv::var("SOURCINATION").unwrap_or_else(|_| "/home/ftpuser/".to_string()),
                    dotenv::var("DESTINATION_LOG").unwrap_or_else(|_| "/home/ftpuser/".to_string()),
                    dotenv::var("DESTINATION_DOC").unwrap_or_else(|_| "/var/pdpa/agent/".to_string()),
                    content
                ).build()
                    .await {
                        Ok(s) => s,
                        Err(e) => e.to_string()
                    }
                // function on test only!!
                // let mut file_table_all = time_function(|| Self::split_string(
                //     &dotenv::var("TB_FILE").unwrap_or_else(|_| "TB_TR_PDPA_AGENT_FILE_DIR:id, device_name, os_name, path, name_file,
size".to_string()),
                //     ":"
                // ), "split_string#4");
                // let file_columns = time_function(|| Self::split_string(&file_table_all.pop().unwrap(), ","), "split_string#5");
                // let file_table = file_table_all.pop().unwrap();
                // let content = time_function(|| Self::split_string(&details[details.len() - 1], "|||"), "split_string#6");
                // let mut start = FileDirectory::new(
                //     db,
                //     file_table,
                //     file_columns,
                //     dotenv::var("SOURCINATION").unwrap_or_else(|_| "/home/ftpuser/".to_string()),
                //     dotenv::var("DESTINATION_LOG").unwrap_or_else(|_| "/home/ftpuser/".to_string()),
                //     dotenv::var("DESTINATION_DOC").unwrap_or_else(|_| "/var/pdpa/agent/".to_string()),
                //     content
                // );
                // match time_function(|| start.build(), "main_file").await {
```

```
                    //         Ok(s) => s,
                    //         Err(e) => e.to_string()
                    //     }
                },
                "AG3" => {
                    let mut dbc_table_all = Self::split_string(
                        &dotenv::var("TB_DB")
                        .unwrap_or_else(|_| "TB_TR_PDPA_AGENT_DATABASE_CHECK:field_1, field_2, field_3, field_4, field_5, field_6, field_7,
field_8, field_9, field_0, from_client".to_string()),
                        ":"
                    );
                    let dbc_columns = Self::split_string(&dbc_table_all.pop().unwrap(), ",");
                    let dbc_table = dbc_table_all.pop().unwrap();
                    let mut content = Self::split_string(&details[details.len() - 1], "|||");

                    // function on test only!!
                    // let mut dbc_table_all = time_function(|| Self::split_string(
                    //     &dotenv::var("TB_DB")
                    //     .unwrap_or_else(|_| "TB_TR_PDPA_AGENT_DATABASE_CHECK:field_1, field_2, field_3, field_4, field_5, field_6, field_7,
field_8, field_9, field_0, from_client".to_string()),
                    //     ":"
                    // ), "split_string#7");
                    // let dbc_columns = time_function(|| Self::split_string(&dbc_table_all.pop().unwrap(), ","), "split_string#8");
                    // let dbc_table = dbc_table_all.pop().unwrap();
                    // let mut content = time_function(|| Self::split_string(&details[details.len() - 1], "|||"), "split_string#9");

                    // Convert message from_client and values
                    let from_client = content.remove(1);

                    DatabaseCheck::new(db, from_client, dbc_table, dbc_columns, content.clone());

                    // Beta.
                    content[0].to_string()
                },
                _ => {
                    "Failed".to_string()
                }
            }
        },
        false => "Close".to_string()
    }
}

#[allow(unused_must_use)]
```

```
    async fn handle_client(mut stream: net::TcpStream, db: MySqlPool) {
        let mut buffer = [0; 1024];
        match stream.read(&mut buffer).await {
            Ok(bytes_read) => {
                // Test only!!
                // println!("{}", &buffer[..bytes_read].len());
                //
                // Main Taskprocess.
                let get_response = String::from_utf8_lossy(&buffer[..bytes_read]);
                // Set message to response to client.
                let message: String;
                match get_response {
                    // Statement test connection sql.
                    s if s.contains('|') && !s.contains('#') => {
                        let conv_test = Self::split_string(&s, "|");
                        // Call check type sql.
                        message = Self::test_connect(conv_test[0].to_owned(), conv_test[1].to_owned(), conv_test[2].to_owned(),
conv_test[3].to_owned(), conv_test[4].to_owned()).await;
                        // message = time_function(|| Self::test_connect(conv_test[0].to_owned(), conv_test[1].to_owned(),
conv_test[2].to_owned(), conv_test[3].to_owned(), conv_test[4].to_owned()), "test_connect").await;
                        if let Err(error) = stream.write_all(message.as_bytes()).await {
                            println!("Failed to write to stream: {}", error);
                        }
                    },
                    // Statement main task process agent.
                    s if s.contains('#') => {
                        // Get query from agent store to check status.
                        let store: Vec<AgentStore> = sqlx::query("SELECT * FROM TB_TR_PDPA_AGENT_STORE")
                            .fetch_all(&db)
                            .await.unwrap()
                            .into_iter()
                            .map(|row| AgentStore::from_row(&row).unwrap())
                            .collect();
                        // Get query from agent manage to insert history.
                        let manager: Vec<AgentManage> = sqlx::query("SELECT pam.agm_id, pam.agm_name, pas.code FROM
TB_TR_PDPA_AGENT_MANAGE as pam JOIN TB_TR_PDPA_AGENT_STORE as pas ON pam.ags_id = pas.ags_id")
                            .fetch_all(&db)
                            .await.unwrap()
                            .into_iter()
                            .map(|row| AgentManage::from_row(&row).unwrap())
                            .collect();
                        // Convert message AG # Details to Vector.
                        let response = Self::split_string(&s, "#");
                        // Get AG_NAME after success.
```

```
                    let result = Self::main_task(Self::status_store(store, response[0].clone()), response.clone(), db.clone()).await;
                    // function on test only!!
                    // let result = time_function(|| Self::main_task(Self::status_store(store, response[0].clone()), response.clone(), db.clone()),
"main_task").await;
                    // Set message to response client.
                    Self::set_history(db.clone(), manager, response[0].to_owned(), result).await;
                    // function on test only!!
                    // time_function(|| Self::set_history(db.clone(), manager, response[0].to_owned(), result), "set_history").await;
                    // shutdown mysql.
                    db.close();
                    // if let Err(error) = stream.write_all(message.as_bytes()).await {
                    //     println!("Failed to write to stream: {}", error);
                    // }
                },
                _ => {
                    message = "Failed unknow type agent. Please check token, .env, or api from alltra again!!".to_string();
                    if let Err(error) = stream.write_all(message.as_bytes()).await {
                        println!("Failed to write to stream: {}", error);
                    }
                },
            }
        }
        Err(error) => {
            println!("Failed to read from stream: {}", error);
        }
    }

    // stream.shutdown().await.unwrap();
}

// pub async fn listen(&self, db: MySqlPool) {
pub async fn listen(&self, db_url: String) {
    let listener = net::TcpListener::bind(format!("{}:{}", &self.host, &self.port)).await.expect("Failed to bind to address");

    println!("Server listening on {}:{}", &self.host, &self.port);

    loop {
        if let Ok(sock) = listener.accept().await {
            let pool = match MySqlPool::connect(&db_url)
                .await {
                    Ok(pool) => pool,
                    Err(err) => {
                        println!("Failed to connect the database: {:?}", err);
                        std::process::exit(1);
```

```
                    }
                };
            tokio::spawn(async move {
                // function on test only!!
                // let (cpu, ram, disk_read, disk_write) = benchmark_env_usage(Duration::from_secs(1));
                // println!(
                //     "Average CPU Usage: {:.2}%, RAM Usage: {:.2} Mb, Disk Read: {:.2} Mb, Disk Write: {:.2} Mb.",
                //     cpu,
                //     ram,
                //     disk_read,
                //     disk_write
                // );
                // time_function(|| Self::handle_client(sock.0, pool), "handle_client").await;
                Self::handle_client(sock.0, pool).await;
            });
        }
        // else {
        //     println!("Failed accepting connection!");
        // }
    }
  }
}
```

รายละเอียดชุดข้อมูลโค้ดในส่วนจัดการตรวจสอบข้อมูลจราจรของแม่ข่าย

```
use sqlx::mysql::MySqlPool;

// use test!!
// use crate::module::test::*;

#[derive(Debug)]
pub struct LogHash {
    connection: MySqlPool,
    table: String,
    column: Vec<String>,
    content: Vec<String>,
}
impl LogHash {
    pub fn new(connection: MySqlPool, table: String, column: Vec<String>, content: Vec<String>) -> LogHash {
        LogHash { connection, table, column, content }
    }

    // Helper client rerun.
    async fn check(&mut self) -> bool {
        let value_where = self.content
```

```
            .iter()
            .zip(self.column.iter())
            .map(|(x, y)| format!("{}=\"{}\"", y.trim(), x.trim()))
            .collect::<Vec<String>>();
        let mut query2 = String::new();
        for i in 0..value_where.len() {
            if i == value_where.len() - 1 {
                query2.push_str(&value_where[i]);
            } else {
                query2.push_str(format!("{} AND ", value_where[i]).as_str());
            }
        }
        let query = "SELECT id FROM (SELECT * FROM TB_TR_PDPA_AGENT_LOG0_HASH ORDER BY id DESC) as log0";
        sqlx::query(format!("{} WHERE {} LIMIT 1", query, query2).as_str())
            .fetch_one(&self.connection)
            .await.is_ok()
    }

    pub async fn build(&mut self) -> Result<String, Box<dyn std::error::Error>> {
        let name = self.content.remove(0);
        match self.check().await {
        // match time_function(|| self.check(), "log0_check").await {
            true => Ok(name),
            false => {
                let values = self.content.iter().map(|s| format!("\"{}\"", s)).collect::<Vec<String>>().join(",");
                let query = format!("INSERT INTO {} ({}) VALUES ({})", self.table, self.column.join(","), values);
                sqlx::query(&query)
                    .execute(&self.connection)
                    .await?;
                Ok(name)
            }
        }
    }
}
```

รายละเอียชุดข้อมูลโค้ดในส่วนประเภทไฟล์ของแม่ข่าย

```
use sqlx::{mysql::{MySqlPool, MySqlRow}, Row};
use std::fs::{read_dir, ReadDir, rename};
use std::path::Path;

//use on test
// use crate::module::test::*;

#[derive(Debug)]
```

```
#[allow(dead_code)]
pub struct FileDirectory {
    connection: MySqlPool,
    table: String,
    column: Vec<String>,
    directory: String,
    log_path: String,
    doc_path: String,
    content: Vec<String>
}

impl FileDirectory {
    pub fn new(connection: MySqlPool, table: String, column: Vec<String>, directory: String, log_path: String, doc_path: String, content:
Vec<String>) -> FileDirectory {
        FileDirectory { connection, table, column, directory, log_path, doc_path, content }
    }

    fn set_query(column: Vec<String>, query: &MySqlRow) -> Vec<String> {
        let mut result = Vec::new();
        for i in column {
            let value: Result<Option<String>, sqlx::Error> = query.try_get(i.trim());
            match value {
                Ok(Some(val)) => result.push(val),
                Ok(None) => result.push("NULL".to_string()),
                Err(e) => {
                    let err: Vec<String> = e.to_string().split(``).into_iter().map(|s| s.to_string()).collect();
                    match err[err.len() - 2].as_str() {
                        "INT" => {
                            let new_value: i32 = query.get(i.trim());
                            result.push(new_value.to_string())
                        },
                        _ => result.push("Unknow".to_string()),
                    }
                },
            }
        }
        result
    }

    fn find_match(arr: Vec<Vec<String>>, val: Vec<String>) -> bool {
        let mut result = false;
        for i in arr {
            let mut equal_result = i[1..].iter().zip(val.iter()).map(|(a, b)| a == b).collect::<Vec<bool>>();
            equal_result.retain(|&value| value);
```

```
        if equal_result.len() > 3 {
            result = true;
        }
    }
    result
}

#[allow(clippy::needless_collect)]
fn reverse_name(files: ReadDir, name: String) -> bool {
    let mut result = false;
    for i in files {
        let filename: Vec<String> =  i.unwrap()
            .file_name()
            .to_string_lossy()
            .split('@')
            .map(|s| s.to_string())
            .collect();
        if filename.contains(&name) {
            result = true;
        }
    }
    result
}

pub async fn build(&mut self) -> Result<String, Box<dyn std::error::Error>> {
    let name = self.content.remove(0);
    let query_dir: Vec<Vec<String>> = sqlx::query(
        format!(
            "SELECT {} FROM {} ORDER BY {} ASC;",
            self.column.join(", "),
            self.table,
            self.column[0]
        ).as_str()
    )
        .fetch_all(&self.connection)
        .await.unwrap()
        .into_iter()
        .map(|row| {
            Self::set_query(self.column.clone(), &row)
            // function on test only!!
            // time_function(|| Self::set_query(self.column.clone(), &row), "file_set_query")
        })
        .collect();
    let values = self.content.clone().iter().map(|s| format!("\"{}\"", s)).collect::<Vec<String>>().join(", ");
```

```
    let files = read_dir(&self.directory).expect("Failed to read directory!");
    // Check result query dir/file == [] and if result query dir/file > 0
    // and check result == value if true = false , false = true
    if query_dir.is_empty() || !Self::find_match(query_dir, self.content.clone()) {
    // function on test only!!
    // if query_dir.is_empty() || time_function(|| !Self::find_match(query_dir, self.content.clone()), "file_find_match") {
        sqlx::query(format!("INSERT INTO {} ({}) VALUES ({})", self.table, &self.column[1..].join(", "), values).as_str())
            .fetch_all(&self.connection)
            .await.unwrap();
    } else if Self::reverse_name(files, self.content[self.content.len() - 2].to_string()) {
    // } else if time_function(|| Self::reverse_name(files, self.content[self.content.len() - 2].to_string()), "file_reverse_name") {
        sqlx::query(
            format!("UPDATE {} SET {} = {}, _get = NOW() WHERE {} = \"{}\"",
                self.table,
                self.column[self.column.len() - 1],
                self.content[self.content.len() - 1],
                self.column[self.column.len() - 2],
                self.content[self.content.len() - 2]
            ).as_str()
        )
            .execute(&self.connection)
            .await.unwrap();
    }
    self.move_dir().await;
    // time_function(|| self.move_dir(), "move_dir").await;
    Ok(name)
}

async fn move_dir(&mut self) {
    let files = read_dir(&self.directory).expect("Failed to read directory!");
    let source_path = match self.directory.chars().nth(self.directory.len() - 1).unwrap() {
        '/' => self.directory.clone(),
        _ => format!("{}/", self.directory),
    };
    for i in files {
        let filename: Vec<String> = i.unwrap().file_name().to_string_lossy()
            .split('@')
            .map(|s| s.to_string())
            .collect();
        if let Some(extension) = Path::new(format!("{}{}", source_path, filename.join("@")).as_str()).extension() {
            if let Some(extension_str) = extension.to_str() {
                if filename[filename.len() - 1] != ".DS_Store" && extension_str == "log" {
                    let destination_path = match self.log_path.chars().nth(self.log_path.len() - 1).unwrap() {
                        '/' => self.log_path.clone(),
```

```
                    _ => format!("{}/", self.log_path),
                };
                rename(
                    format!("{}{}", source_path, filename.join("@")).as_str(),
                    format!("{}{}", destination_path, filename.join("@")).as_str()
                ).unwrap();
            } else if filename[filename.len() - 1] != ".DS_Store" && vec!["csv", "xlsx", "xls"].contains(&extension_str) {
                let destination_path = match self.doc_path.chars().nth(self.doc_path.len() - 1).unwrap() {
                    '/' => self.doc_path.clone(),
                    _ => format!("{}/", self.doc_path),
                };
                rename(
                    format!("{}{}", source_path, filename.join("@")).as_str(),
                    format!("{}{}", destination_path, filename.join("@")).as_str()
                ).unwrap();
            }
        }
    }
  }
}
```

รายละเอียชุดข้อมูลโค้ดในส่วนตรวจสอบฐานข้อมูลของแม่ข่าย

```
#![allow(dead_code)]
use sqlx::{
    mysql::{MySqlPoolOptions, MySqlPool},
    Row
};
use serde::Serialize;
use oracle::pool::{PoolBuilder, Pool as OraclePool, CloseMode};

// use test
// use crate::module::test::*;

#[derive(Debug, Default, Serialize)]
struct Table {
    name: String,
    columns: Vec<String>,
}

#[derive(Debug)]
pub struct TestConnect {
    host: String,
    user: String,
```

```
    passwd: String,
    database: String,
}
impl TestConnect {
    pub fn new(host: String, user: String, passwd: String, database: String) -> TestConnect {
        TestConnect { host, user, passwd, database }
    }

    #[allow(unused_must_use)]
    pub async fn mysql(&self) -> Result<String, sqlx::Error> {
        let database_url = format!("mysq://{}:{}@{}:3306/{}", self.user, self.passwd, self.host, self.database);
        let pool = MySqlPoolOptions::new()
            .max_connections(10)
            .connect(&database_url)
            .await;
        match pool {
            Ok(pool) => {
                let tables: Vec<String> = sqlx::query("SHOW TABLES")
                    .fetch_all(&pool)
                    .await.unwrap()
                    .into_iter()
                    .map(|row| match row.try_get(format!("Tables_in_{}", self.database.to_lowercase()).as_str()) {
                        Ok(row) => row,
                        Err(_) => row.get(format!("Tables_in_{}", self.database).as_str()),
                    })
                    .collect();
                let mut mix: Vec<Table> = vec![];
                for i in tables {
                    let res: Vec<String> = sqlx::query(&format!("DESCRIBE {}", i))
                        .fetch_all(&pool)
                        .await.unwrap()
                        .into_iter()
                        .map(|row| row.get("Field"))
                        .collect();
                    mix.push(Table { name: i, columns: res });
                }
                pool.close();
                Ok(serde_json::to_string(&mix).unwrap())
            }
            Err(e) => Ok(e.to_string())
        }
    }

    async fn oracle_query(&self, query: &str, pool: OraclePool) -> Result<Vec<String>, oracle::Error> {
```

```
        let conn = pool.get()?;
        match conn.query(query, &[]) {
            Ok(query) => {
                let res = query.into_iter()
                    .map(|row| match row.expect("REASON").get(0) {
                        Ok(row) => row,
                        Err(e) => e.to_string(),
                    })
                    .collect::<Vec<String>>();
                conn.close()?;
                Ok(res)
            }
            Err(err) => Ok(vec![err.to_string()])
        }
    }

    #[allow(unused_must_use)]
    pub async fn oracle(&self) -> Result<String, oracle::Error> {
        //Set Oracle instant client.
        // std::env::set_var("LD_LIBRARY_PATH", "/Users/kuzan04/Desktop/instantclient_19_8/");
        let pool = PoolBuilder::new(self.user.as_str(), self.passwd.as_str(), format!("//{}:1521/{}", self.host, self.database).as_str())
            .max_connections(10)
            .build();

        match pool {
            Ok(pool) => {
                let mut mix: Vec<Table> = vec![];
                let tables = self.oracle_query("SELECT table_name FROM user_tables", pool.clone()).await?;
                for i in tables {
                    let column = self.oracle_query(format!("SELECT column_name FROM all_tab_columns WHERE table_name = '{}' GROUP BY
column_name, column_id ORDER BY column_id", i).as_str(), pool.clone()).await?;
                    // function on test only!!
                    // let format_cols = format!("SELECT column_name FROM all_tab_columns WHERE table_name = '{}' GROUP BY
column_name, column_id ORDER BY column_id", i);
                    // let format_name = format!("oracle_query#{}", i);
                    // let format_name_static: &'static str = Box::leak(format_name.into_boxed_str());
                    // let column = time_function(|| self.oracle_query(&format_cols, pool.clone()), format_name_static).await?;
                    mix.push(Table{ name: i, columns: column});
                }
                // Not sure, ## Can Force to disconnect. ##
                pool.close(&CloseMode::Default);
                Ok(serde_json::to_string(&mix).unwrap())
            }
            Err(e) => Ok(e.to_string())
```

```
        }
    }
}

#[derive(Debug)]
pub struct DatabaseCheck {
    connection: MySqlPool,
    from: String,
    table: String,
    columns: Vec<String>,
    content: Vec<String>
}

impl DatabaseCheck {
    pub fn new(connection: MySqlPool, from: String, table: String, columns: Vec<String>, content: Vec<String>) -> Self {
        Self { connection, from, table, columns, content }
    }
}
```

รายละเอียชุดข้อมูลโค้ดในส่วนหลักของลูกข่าย

```
extern crate dotenv;

// Package install.
use dotenv::dotenv;
use base64::{Engine as _, engine::general_purpose};
use sqlx::mysql::MySqlPoolOptions;

// Packge already exist after install Rust Language.
use std::fs::{File, metadata, remove_file};
use std::io::{self, Write, BufReader, BufRead};
use std::env;

mod module;
mod model;
use crate::module::handles::Handler;

// on test only!
// use crate::module::test::*;

fn set_env(input: Vec<&str>) {
    let details = vec!["TYPE", "STATUS", "NAME", "HOST", "PORT", "DETAILS", "TOKEN"]
        .iter()
        .zip(input.iter())
        .map(|(&x, &y)| format!("{}=\"{}\"\n", x, y))
```

```
        .collect::<Vec<_>>();
    let mut file = File::create(".env").unwrap();
    for i in details {
        file.write_all(i.as_bytes()).unwrap();
    }
    let line_count = BufReader::new(File::open(".env").unwrap()).lines().count();
    if line_count != 7 {
        remove_file(".env").unwrap();
        println!("[Error] Token incorrect!");
        std::process::exit(1);
    }
}

#[tokio::main]
async fn main() {
    dotenv().ok();

    if metadata(".env").is_err() {
        loop {
            print!("Please enter the token you have: ");
            io::stdout().flush().unwrap();

            let mut input = String::new();
            io::stdin().read_line(&mut input).expect("Failed to read input!");

            match !input.trim().is_empty() && general_purpose::STANDARD.decode(input.trim()).is_ok() {
                true => {
                    if let Ok(decoded_bytes) = general_purpose::STANDARD.decode(input.trim()) {
                        let base = String::from_utf8_lossy(&decoded_bytes).to_string();
                        let mut mix = base.split("&&&").collect::<Vec<&str>>();
                        mix.push(input.trim());
                        set_env(mix.to_vec());
                        // function on test only!!
                        // time_function(|| set_env(mix.to_vec()), "set_env");

                        break
                    } else {
                        println!("Invalid input. Value must be base64!");
                    }
                },
                false => println!("Invalid input. Please enter the token on base64"),
            }
        }
    }
```

```
    env::set_var("DB_MAIN", "DOL_PDPA_LOCAL");
    let database_url = format!("mysql://root:P@ssw0rd@localhost:3306/{}", env::var("DB_MAIN").unwrap_or_else(|_| "DOL_PDPA".to_string()));
    let pool = match MySqlPoolOptions::new()
        .max_connections(10)
        .connect(&database_url)
        .await {
            Ok(pool) => {
                pool
            }
            Err(e) => {
                println!("Failed to connect the database: {:?}", e);
                std::process::exit(1);
            }
        };

    // Main process.
    Handler::new(
        pool,
        dotenv::var("HOST").unwrap_or_else(|_| "127.0.0.1".to_string()),
        dotenv::var("PORT").unwrap_or_else(|_| 5050.to_string()),
    ).task().await;

    // Option send details.
    // dotenv::vars().collect::<HashMap<String, String>>()
}
```

รายละเอียดชุดข้อมูลโค้ดในส่วนโมเดลของลูกข่าย

```
use serde::{Deserialize, Serialize};
use sqlx::{FromRow, mysql::MySqlRow, Row, MySqlPool};
use oracle::{Row as OracleRow, pool::Pool as OraclePool};
use chrono::{NaiveDate, NaiveDateTime};

#[derive(Debug, Deserialize, Serialize, Clone, PartialEq)]
#[allow(non_snake_case)]
pub struct FilterAgentManage {
    pub code: String,
    pub name: String,
    pub status: i32,
    pub details: String,
    pub token: String,
}

impl FilterAgentManage {
```

```
    pub fn new(code: String, name: String, status: i32, details: String, token: String) -> Self {
        Self { code, name, status, details, token }
    }
}

impl Default for FilterAgentManage {
    fn default() -> Self {
        Self { code: "NULL".to_string(), name: "NULL".to_string(), status: -1, details: "NULL".to_string() , token: "NULL".to_string() }
    }
}

impl FromRow<'_, MySqlRow> for FilterAgentManage {
  fn from_row(row: &MySqlRow) -> Result<Self, sqlx::Error> {
    let name: String = row.try_get("agm_name")?;
    let code: String = row.try_get("code")?;
    let status: i32 = row.try_get("agm_status")?;
    let details: String = row.try_get("config_detail")?;
    let token: String = row.try_get("agm_token")?;

    Ok(Self { name, code, status, details, token})
  }
}

#[derive(Debug, Deserialize, Serialize, Clone, PartialEq)]
#[allow(non_snake_case)]
pub struct LogStore {
    pub device_name: String,
    pub os_name: String,
    pub path: String,
    pub name_file: String,
    pub total_line: String,
}

#[allow(dead_code)]
impl LogStore {
    pub fn new(device_name: String, os_name: String, path: String, name_file: String, total_line: String) -> Self {
        Self { device_name, os_name, path, name_file, total_line }
    }
}

impl Default for LogStore {
    fn default() -> Self {
        Self { device_name: "NULL".to_string(), os_name: "NULL".to_string(), path: "NULL".to_string(), name_file: "NULL".to_string(), total_line:
0.to_string() }
```

```
    }
}

impl FromRow<'_, MySqlRow> for LogStore {
    fn from_row(row: &MySqlRow) -> Result<Self, sqlx::Error> {
        let device_name: String = row.try_get("device_name")?;
        let os_name: String = row.try_get("os_name")?;
        let path: String = row.try_get("path")?;
        let name_file: String = row.try_get("name_file")?;
        let total_line: String = row.try_get("total_line")?;

        Ok(Self { device_name, os_name, path, name_file, total_line })
    }
}

#[derive(Debug)]
pub enum DateOrDateTime {
    Date(NaiveDate),
    DateTime(NaiveDateTime),
}

impl ToString for DateOrDateTime {
    fn to_string(&self) -> String {
        match self {
            DateOrDateTime::Date(date) => date.to_string(),
            DateOrDateTime::DateTime(datetime) => datetime.to_string(),
        }
    }
}

#[derive(Debug)]
pub enum PoolRow {
    MyRow(MySqlRow),
    OrRow(OracleRow),
}

#[derive(Debug)]
pub enum PoolType {
    MyPool(MySqlPool),
    OrPool(OraclePool),
}

#[derive(Debug)]
#[allow(dead_code)]
```

```
pub struct SizeFile {
    name: String,
    source: String,
    destination: String,
}

#[allow(dead_code)]
impl SizeFile {
    pub fn new(name: String, source: String, destination: String) -> Self {
        Self { name, source, destination }
    }
}

#[derive(Debug, Clone, PartialEq)]
pub struct MyInterface {
    pub ip: String,
    pub netmask: String,
    pub broadcast: Option<String>,
}

impl Default for MyInterface {
    fn default() -> Self {
        Self { ip: "NULL".to_string(), netmask: "NULL".to_string(), broadcast: None }
    }
}
```

รายละเอียดชุดข้อมูลโค้ดในส่วนเชื่อมต่อแม่ข่ายของลูกข่าย

```
use tokio::net::TcpStream;
use tokio::io::AsyncWriteExt;
use sqlx::{FromRow, mysql::MySqlPool};
use sys_info::{hostname, os_release};

use std::{env, fs};
use std::process;
// use std::time::Duration;

use crate::model::FilterAgentManage;
use crate::module::{
    log0::LogHash,
    file::DirectoryFile,
    db::DatabaseCheck,
    sniffer::TaskSniffer,
};
```

```
// use on test
use crate::module::test::*;

#[derive(Debug)]
pub struct Handler {
    pub db: MySqlPool,
    pub host: String,
    pub port: String,
} impl Handler {
    pub fn new(db: MySqlPool, host: String, port: String) -> Handler {
        Handler { db, host, port }
    }

    fn check(fetch: Vec<FilterAgentManage>, token: String) -> FilterAgentManage {
        let mut result = FilterAgentManage::default();
        let mut i = 0;
        while i < fetch.len() {
            if fetch[i].token == token {
                result = fetch[i].clone();
            }
            i+=1
        }
        result
    }

    async fn set_status(fetch: FilterAgentManage, details: FilterAgentManage) -> bool {
        match details.status {
            0 | 1 => {
                let new_env = [
                    ("STATUS", fetch.status.to_string()),
                    ("TOKEN", fetch.token),
                ];
                let mut env_file_content = fs::read_to_string(".env").unwrap();
                for (key, new) in new_env {
                    let current_env = env::var(key).unwrap();
                    env::set_var(key, &new);
                    env_file_content = env_file_content.replace(&format!("{}=\"{}\"\n", key, current_env), &format!("{}=\"{}\"\n", key, new));
                }
                fs::write(".env", env_file_content).unwrap();
                match fetch.status {
                    1 => true,
                    0 => false,
                    _ => {
                        println!("[Error] Agent client encountering a problem from database!!!!");
```

```
                process::exit(1);
            }
            // Today 2023.06.4 Web alltra function update not have token.
            // env::set_var("TOKEN", fetch.status);
        }
    },
    _ => {
        println!("[Error] Value of status overdue!!!");
        process::exit(1);
    }
  }
}

#[allow(unused_must_use)]
async fn main_task(&mut self) -> Vec<String> {
  match env::var("TYPE") {
    Ok(code) => {
        // Set varriable to send to listener.
        let hostname = match hostname() {
            Ok(name) => name,
            Err(_) => "unknow".to_string(),
        };
        let platform = env::consts::OS;
        let release = match os_release() {
            Ok(os) => os,
            Err(_) => "unknow".to_string(),
        };
        match code.as_str() {
            "AG1" => {
                let details = env::var("DETAILS").unwrap().split(',').map(|s| s.to_string()).collect::<Vec<String>>();
                // match LogHash::new(
                //     hostname,
                //     format!("{} {}", platform, release),
                //     details,
                //     self.db.clone()
                // ).build().await {
                //         Ok(result) => result,
                //         Err(err) => vec![format!("[Failed] {}", err)],
                //     }
                // function on test only!!
                let mut start = LogHash::new(hostname, format!("{} {}", platform, release), details, self.db.clone());
                match time_function(|| start.build(), "log0_start").await {
                    Ok(result) => result,
                    Err(err) => vec![format!("[Failed] {}", err)],
```

```
    }
},
"AG2" => {
    let details = env::var("DETAILS").unwrap().split(',').map(|s| s.to_string()).collect::<Vec<String>>();
    let mix = vec![env::var("TYPE").unwrap(), env::var("NAME").unwrap(), hostname, format!("{}-{}", platform, release)];
    // match DirectoryFile::new(
    //     mix,
    //     details,
    //     env::var("HOST").unwrap(),
    //     21,
    //     "ftpuser".to_string(),
    //     "ftpuser".to_string(),
    // ).build().await {
    //         Ok(result) => result,
    //         Err(err) => vec![format!("[Failed] {}", err)]
    //     }
    // function on test only!!
    let start = DirectoryFile::new(mix, details, env::var("HOST").unwrap(), 21, "ftpuser".to_string(), "ftpuser".to_string());
    match time_function(|| start.build(), "file_start").await {
        Ok(result) => result,
        Err(err) => vec![format!("[Failed] {}", err)]
    }
},
"AG3" => {
    let mut details = env::var("DETAILS").unwrap().split('&').map(|s| s.to_string()).collect::<Vec<String>>();
    let db_type = details.remove(0).parse::<i32>().unwrap_or(-1);
    // match DatabaseCheck::new(
    //     self.db.clone(),
    //     db_type,
    //     details,
    // ).build().await {
    //         Ok(result) => result,
    //         Err(err) => vec![format!("[Failed] {}", err)],
    //     }
    //  function on test only!!
    let start = DatabaseCheck::new(self.db.clone(), db_type, details);
    match time_function(|| start.build(), "database_start").await {
        Ok(result) => result,
        Err(err) => vec![format!("[Failed] {}", err)],
    }
},
"AG4" => {
    let selected = "en0".to_string();
    let details = env::var("DETAILS").unwrap().split(',').map(|s| s.to_string()).collect::<Vec<String>>();
```

```
                    TaskSniffer::new(self.db.clone(), details, selected).run().await;
                    vec!["Skip listener".to_string()]
                },
                _ => {
                    println!("[Error] Type of agent client overdue!!!");
                    process::exit(1);
                }
            }
        },
        Err(err) => {
            println!("[Error] {}", err);
            process::exit(1);
        }
    }
}

pub async fn task(&mut self) {
    // loop {
        let query = "SELECT pas.code, am.agm_name, am.config_detail, am.agm_status, am.agm_token FROM
TB_TR_PDPA_AGENT_MANAGE as am JOIN TB_TR_PDPA_AGENT_STORE as pas ON am.ags_id = pas.ags_id";
        let result_manager: Vec<FilterAgentManage>= sqlx::query(query)
            .fetch_all(&self.db)
            .await.unwrap()
            .into_iter()
            .map(|row| FilterAgentManage::from_row(&row).unwrap())
            .collect();

        let result_filter = Self::check(result_manager, env::var("TOKEN").unwrap_or_else(|_| "unknow".to_string()));
        // function on test only!!
        // let result_filter = time_function(|| Self::check(result_manager, env::var("TOKEN").unwrap_or_else(|_| "unknow".to_string())), "check");
        let status = match env::var("STATUS").unwrap_or_else(|_| "-1".to_string()).parse::<i32>() {
            Ok(s) => s,
            Err(err) => {
                let err = format!("[Error] Value of status incorrect!!\n{:?}", err.to_string());
                println!("{}", err);
                process::exit(1);
            }
        };

        let reverse_details = FilterAgentManage::new(
            env::var("TYPE").unwrap_or_else(|_| "unknow".to_string()),
            env::var("NAME").unwrap_or_else(|_| "unknow".to_string()),
            status,
            env::var("DETAILS").unwrap_or_else(|_| "unknow".to_string()),
```

```
    env::var("TOKEN").unwrap_or_else(|_| "unknow".to_string()),
);

// Main task if statement check 2 condition is struct (Object) not default
// and status client start run.
if result_filter != FilterAgentManage::default() && Self::set_status(result_filter, reverse_details).await {
// function on test only!!
// if result_filter != FilterAgentManage::default() && time_function(|| Self::set_status(result_filter, reverse_details), "set_status").await {
    let message: Vec<String> = self.main_task().await
    // let message: Vec<String> = time_function(|| self.main_task(), "main_task").await
        .into_iter()
        .map(|msg| {
            format!("{}#{}|||{}",
                env::var("TYPE").unwrap(),
                env::var("NAME").unwrap(),
                msg
            )
        })
        .collect();
    // function on test only!!
    // let (cpu, ram, disk_read, disk_write) = benchmark_env_usage(Duration::from_secs(1));
    // println!(
    //     "Average CPU Usage: {:.2}%, RAM Usage: {:.2} Mb, Disk Read: {:.2} Mb, Disk Write: {:.2} Mb.",
    //     cpu,
    //     ram,
    //     disk_read,
    //     disk_write
    // );
    if env::var("TYPE").unwrap() != "AG4" {
        for i in message {
            // println!("{}", i.as_bytes().len());
            let mut stream = TcpStream::connect(format!("{}:{}", &self.host, &self.port)).await.expect("Failed to connect to server");
            if let Err(error) = stream.write_all(i.as_bytes()).await {
                println!("Failed to write to stream: {}", error);
            }
            std::thread::sleep(std::time::Duration::from_secs(1));
            drop(stream);
        }
        // let _ = &self.db.close();
    } else {
        // break;
    }
} else {
    // continue;
```

```
            }
        }
        // drop(stream);
        // let mut buffer = [0; 1024];
        // match stream.read(&mut buffer).await {
        //     Ok(_) => continue,
        //     Err(_) => continue,
        // }
    // }
}
```

รายละเอียดชุดข้อมูลโค้ดในส่วนจัดตรวจสอบข้อมูลจราจรของลูกข่าย

```
#![allow(dead_code)]
use sqlx::{MySqlPool, FromRow};
use tokio::time::sleep;
use sha256::try_digest;
use md5::compute;
use sha1::{Sha1, Digest};
use std::fs::{File, read_dir};
use std::path::{Path, PathBuf};
use std::io::{self, Read, BufReader, BufRead};
use std::time::Duration;

use crate::model::LogStore;

// use test
use crate::module::test::*;

#[derive(Debug)]
pub struct LogHash {
    device_name: String,
    os_name: String,
    directory: Vec<String>,
    connection: MySqlPool,
}

impl LogHash {
    pub fn new(device_name: String, os_name: String, directory: Vec<String>, connection: MySqlPool) -> Self {
        Self { device_name, os_name, directory, connection }
    }

    async fn sha256sum(file: PathBuf) -> String {
        match try_digest(file.as_path()) {
            Ok(hash) => hash,
```

```
            Err(err) => format!("[Failed] {}", err)
        }
    }

    async fn md5sum(file: PathBuf) -> String {
        let mut buffer = Vec::new();
        File::open(file.as_path()).expect("Failed to open file").read_to_end(&mut buffer).expect("Failed to read file");
        format!("{:x}", compute(&buffer))
    }

    async fn sha1sum(file: PathBuf) -> String {
        let mut buffer = Vec::new();
        File::open(file.as_path()).expect("Failed to open file").read_to_end(&mut buffer).expect("Failed to read file");
        let mut sha1 = Sha1::new();
        sha1.update(&buffer);
        format!("{:x}", sha1.finalize())
    }

    // check have for check run client after first time.
    async fn check(&self, dir: String, file: String) -> i32 {
        let query1 = "SELECT device_name, os_name, path, name_file, total_line FROM (SELECT * FROM TB_TR_PDPA_AGENT_LOG0_HASH
ORDER BY id DESC) as log0";
        let query2 = format!(
            " WHERE log0.device_name = \"{}\" AND log0.os_name = \"{}\" AND log0.path = \"{}\" AND log0.name_file = \"{}\" LIMIT 1",
            self.device_name,
            self.os_name,
            dir,
            file
        );
        let mut store: Vec<LogStore> = sqlx::query(format!("{}{}",query1,query2).as_str())
            .fetch_all(&self.connection)
            .await.unwrap()
            .into_iter()
            .map(|row| LogStore::from_row(&row).unwrap())
            .collect();

        // Return Here!
        match store.len() {
            0 => -1,
            _ => {
                let selected = store.pop().unwrap();
                selected.total_line.parse::<i32>().unwrap()
            }
        }
```

```
}

async fn calculate_hash(&mut self, dir:String, file: PathBuf) -> String {
    let line_count = BufReader::new(File::open(file.clone()).unwrap()).lines().count();
    // let res_256= Self::sha256sum(file.clone()).await;
    // let res_md5 = Self::md5sum(file.clone()).await;
    // let res_sha1 = Self::sha1sum(file.clone()).await;
    let filename = file.file_name().unwrap().to_str().unwrap();
    // function on test only!!
    let res_256 = time_function(|| Self::sha256sum(file.clone()), "log0_sha256").await;
    let res_md5 = time_function(|| Self::md5sum(file.clone()), "log0_md5").await;
    let res_sha1 = time_function(|| Self::sha1sum(file.clone()), "log0_sha1").await;
    // Example result :=
    // _host_|||_plaform_ _release_|||_.log_|||_number_|||_sha256_|||_md5_|||_sha1_
    format!("{}|||{}|||{}|||{}|||{}|||{}|||{}|||{}|||{}",
        self.device_name,
        self.os_name,
        dir,
        filename,
        line_count,
        res_256,
        res_md5,
        res_sha1
    )
}

pub async fn build(&mut self) -> Result<Vec<String>, io::Error> {
    let mut message: Vec<String> = vec![];
    for i in self.directory.clone() {
        let dir_path = Path::new(&i);
        let mut entries: Vec<String> = read_dir(dir_path).unwrap()
            .into_iter()
            .map(|s| {
                let name = s.unwrap().file_name().to_string_lossy().to_string();
                if let Some(extension) = dir_path.join(name.clone()).extension().unwrap().to_str() {
                    if extension == "log" || extension == "evtx" {
                        name
                    } else {
                        "Other".to_string()
                    }
                } else {
                    "None".to_string()
                }
            })
```

```
    .collect();
// Retain with get true value.
entries.retain(|x| x != "Other" && x != "None");

// Read directory first time.
if entries.is_empty() {
    loop {
        // Delay wait file in directory.
        sleep(Duration::from_secs(3)).await;
        // Call function create 3 type hash.
        entries = read_dir(dir_path).unwrap()
            .into_iter()
            .map(|s| {
                let name = s.unwrap().file_name().to_string_lossy().to_string();
                if let Some(extension) = dir_path.join(name.clone()).extension().unwrap().to_str() {
                    if extension == "log" || extension == "evtx" {
                        name
                    } else {
                        "Other".to_string()
                    }
                } else {
                    "None".to_string()
                }
            })
            .collect();
        match entries.len() {
            0 => continue,
            _ => match entries[0].as_str() {
                "Other" | "None" => continue,
                _ => break,
            },
        }
    }
    // message.push(self.calculate_hash(i.to_string(), dir_path.join(entries[0].as_str())).await);
    // function on test only!!
    message.push(time_function(|| self.calculate_hash(i.to_string(), dir_path.join(entries[0].as_str())), "log0_calculate_hash#1").await);
} else {
    for j in &entries {
        // Setup get lines from file.
        let read_first = BufReader::new(File::open(dir_path.join(j.clone())).unwrap()).lines().count();

        // Process check file.
        // if self.check(i.to_string(), j.to_string()).await == -1 {
        if time_function(|| self.check(i.to_string(), j.to_string()), "log0_check#1").await == -1 {
```

```
                    // message.push(self.calculate_hash(i.to_string(), dir_path.join(j)).await);
                    // function on test only!!
                    message.push(time_function(|| self.calculate_hash(i.to_string(), dir_path.join(entries[0].as_str())),
"log0_calculate_hash#2").await);
                } else {
                    // let backup = self.check(i.to_string(), j.to_string()).await;
                    // function on test only!!
                    let backup = time_function(|| self.check(i.to_string(), j.to_string()), "log0_check#2").await;
                    match backup as usize != read_first {
                        true => {
                            // message.push(self.calculate_hash(i.to_string(), dir_path.join(j)).await);
                            // function on test only!!
                            message.push(time_function(|| self.calculate_hash(i.to_string(), dir_path.join(j)), "log0_calculate_hash#3").await);
                        },
                        false => {
                            sleep(Duration::from_secs(2)).await;
                            let read_second = BufReader::new(File::open(dir_path.join(j.clone())).unwrap()).lines().count();
                            if read_first < read_second {
                                message.push(self.calculate_hash(i.to_string(), dir_path.join(j)).await);
                                // function on test only!!
                                message.push(time_function(|| self.calculate_hash(i.to_string(), dir_path.join(j)), "log0_calculate_hash#4").await);
                            } else {
                                continue;
                            }
                        }
                    }
                }
            }
        }
        Ok(message)
    }
}
```

รายละเอียชุดข้อมูลโค้ดในส่วนจัดการไฟล์ของลูกข่าย

```
use async_ftp::FtpStream;
use ssh2::Session;

use std::{io, io::Read};
use std::path::Path;
use std::fs::{File, read_dir, metadata};

// use test
use crate::module::test::*;
```

```
#[derive(Debug)]
#[allow(dead_code)]
pub struct DirectoryFile {
    details: Vec<String>,
    directory: Vec<String>,
    ftp_host: String,
    ftp_port: i32,
    ftp_user: String,
    ftp_passwd: String,
}

impl DirectoryFile {
    pub fn new(details: Vec<String>, directory: Vec<String>, ftp_host: String, ftp_port: i32, ftp_user: String, ftp_passwd: String) -> Self {
        Self { details, directory, ftp_host, ftp_port, ftp_user, ftp_passwd }
    }

    #[allow(unused_must_use)]
    async fn ftp_handle(&self, name: String, file: String) {
        match self.ftp_port {
            21 => {
                // Set varriable to use put file ftp.
                let mut buffer = Vec::new();
                File::open(file).expect("Failed to open file").read_to_end(&mut buffer).expect("Failed to read file");
                let mut content = io::Cursor::new(buffer);

                let mut ftp_stream = FtpStream::connect(format!("{}:{}", self.ftp_host, "1021")).await.unwrap();
                ftp_stream.login(&self.ftp_user, &self.ftp_passwd).await;

                let set_name: Vec<&str> = match name.contains('\n') {
                    true => name.split('\n').enumerate().filter(|&(i, _)| i == 1).map(|(_, e)| e).collect::<Vec<&str>>(),
                    false => vec![name.as_str()]
                };
                // Store (PUT) a file from client to server on root directory.
                ftp_stream.put(
                    set_name[set_name.len() - 1],
                    &mut content
                ).await.unwrap();

                // Disconnect FTP server.
                ftp_stream.quit().await;
            },
            22 => {
                // Establish an SSH session.
```

```
            let tcp = std::net::TcpStream::connect(format!("{}:{}", self.ftp_host, self.ftp_port)).unwrap();
            let mut sess = Session::new().unwrap();
            sess.set_tcp_stream(tcp);
            sess.handshake().unwrap();

            // Open as SFTP channel.
            let channel = sess.sftp().unwrap();

            // Write a local file to the remote server.
            let mut local_file = File::open(file).unwrap();
            let mut remote_file = channel.create(Path::new(&name)).unwrap();
            io::copy(&mut local_file, &mut remote_file).unwrap();

            // Close the SFTP channel and SSH session.
            drop(remote_file);
            drop(channel);
            sess.disconnect(None, "Bye bye", Some("0")).unwrap();
        },
        _ => println!("[Error] unknow type of ftp or sftp"),
    }
}

pub async fn build(&self) -> Result<Vec<String>, io::Error> {
    let digits = vec!["0".to_string(); 3];
    let mut message: Vec<String> = vec![];
    for i in 0..self.directory.len() {
        let inform = format!(
            "{}@{}{}@{}@",
            self.details[0], // TYPE of client.
            digits[0..(digits.len() - (i+1).to_string().chars().count())].join(""),
            (i+1),
            self.details[1].split_whitespace().map(|s| s.to_string()).collect::<Vec<String>>().join("_") // NAME of client.
        );
        let mut file: Vec<String> = read_dir(Path::new(&self.directory[i])).unwrap()
            .into_iter()
            .map(|f| match f {
                Ok(entry) => entry.file_name().to_string_lossy().to_string(),
                Err(_) => "Err".to_string(),
            })
            .collect();
        file.retain(|x| {
            let ext = Path::new(x).extension().and_then(|ext| ext.to_str()).unwrap_or("");
            x != ".DS_Store" && vec!["log", "xls", "xlsx", "csv", "evtx"].contains(&ext)
        });
```

```
            // Process FTP.
            for j in file {
                // Set varriable to use put file ftp.
                let _set_name = format!("{}{}", inform, j.clone());
                let full_file = Path::new(&self.directory[i]).join(j.clone()).to_str().unwrap().to_string();
                let size = match metadata(full_file.clone()) {
                    Ok(l) => l.len(),
                    Err(_) => 0,
                };

                // Result
                // self.ftp_handle(set_name, full_file).await;
                // self.ftp_handle(j.clone(), full_file).await;
                // function on test only!!
                // time_function(|| self.ftp_handle(set_name, full_file), "file_ftp_handle").await;
                time_function(|| self.ftp_handle(j.clone(), full_file), "file_ftp_handle").await;
                message.push(
                    format!("{}|||{}|||{}|||{}|||{}",
                        self.details[2], // Device_name
                        self.details[3], // OS-Release
                        self.directory[i],
                        j,
                        size
                    )
                );
            }
        }
        Ok(message)
    }
}
```

รายละเอียชุดข้อมูลโค้ดในส่วนตรวจสอบฐานข้อมูลของลูกข่าย

```
use sqlx::{MySqlPool, Row};
use oracle::{
    pool::{PoolBuilder, CloseMode}, Error as OracleError
};
use chrono::{NaiveDate, NaiveDateTime};
use rust_decimal::prelude::*;

use std::env;

use crate::model::{DateOrDateTime, PoolRow, PoolType};
```

```
// use test
// use crate::module::test::*;

#[derive(Debug)]
pub struct DatabaseCheck {
    connection: MySqlPool,
    _type: i32,
    host: String,
    user: String,
    passwd: String,
    db_name: String,
    tables: Vec<String>,
}

impl DatabaseCheck {
    pub fn new(connection: MySqlPool, _type: i32, mut details: Vec<String>) -> Self {
        let host: String = details.remove(0);
        let user: String = details.remove(0);
        let passwd: String = details.remove(0);
        let db_name: String = details.remove(0);
        let tables: Vec<String> = details[details.len()-1..].to_vec();
        Self { connection, _type, host, user, passwd, db_name, tables }
    }

    fn set_query(_type: i32, column: Vec<String>, query: PoolRow) -> Vec<String> {
        let mut result = Vec::new();
        match query {
            PoolRow::MyRow(query) => {
                for i in column {
                    let value = query.try_get(i.trim());
                    match value {
                        Ok(Some(val)) => result.push(val),
                        Ok(None) => result.push("NULL".to_string()),
                        Err(e) => {
                            let err: Vec<String> = e.to_string().split(`'`).into_iter().map(|s| s.to_string()).collect();
                            match err[err.len() - 2].as_str() {
                                "INT" | "BIGINT" | "TINYINT" => {
                                    let new_value: i64 = query.get(i.trim()); // int = i32, bigint = i64
                                    result.push(new_value.to_string())
                                },
                                "BOOLEAN" | "BOOL" => {
                                    let new_value: bool = query.get(i.trim());
                                    result.push(new_value.to_string())
                                },
```

```
                "FLOAT" | "DOUBLE" | "DOUBLE PRECISION" | "DEC" => {
                    let new_value: f64 = query.get(i.trim());
                    result.push(new_value.to_string())
                },
                "DECIMAL" | "BIGDECIMAL" => {
                    let new_value: Decimal = query.get(i.trim());
                    result.push(new_value.to_string())
                },
                "DATE" | "DATETIME" | "DATE_TIME" | "TIMESTAMP" => {
                    let new_value = match query.try_get::<NaiveDate, _>(i.trim()) {
                        Ok(date) => DateOrDateTime::Date(date),
                        Err(_) => DateOrDateTime::DateTime(query.try_get::<NaiveDateTime, _>(i.trim()).unwrap()),
                    };
                    result.push(new_value.to_string())
                },
                _ => result.push("Unknow".to_string()),
            }
        },
    }
}
},
PoolRow::OrRow(query) => {
    for i in 0..column.len() {
        let value: Result<Option<String>, OracleError> = query.get(i);
        match value {
            Ok(Some(val)) => result.push(val),
            Ok(None) => result.push("NULL".to_string()),
            Err(e) => {
                let err: Vec<String> = e.to_string().split('`').into_iter().map(|s| s.to_string()).collect();
                match err[err.len() - 2].as_str() {
                    "INT" | "BIGINT" | "TINYINT" => {
                        let new_value: i64 = query.get(i).unwrap(); // int = i32, bigint = i64
                        result.push(new_value.to_string())
                    },
                    "BOOLEAN" | "BOOL" => {
                        let new_value: bool = query.get(i).unwrap();
                        result.push(new_value.to_string())
                    }
                    "FLOAT" | "DOUBLE" | "DOUBLE PRECISION" | "DECIMAL" | "BIGDECIMAL" | "DEC" => { // not sure
                        let new_value: f64 = query.get(i).unwrap();
                        result.push(new_value.to_string())
                    }
                    "DATE" | "DATETIME" | "DATE_TIME" | "TIMESTAMP" => {
                        let new_value = match query.get(i) {
```

```
                                Ok(date) => DateOrDateTime::Date(date),
                                Err(_) => DateOrDateTime::DateTime(query.get(i).unwrap()),
                            };
                            result.push(new_value.to_string())
                        },
                        _ => result.push("Unknow".to_string()),
                    }
                }
            }
        }
    }
    result
}

async fn check_row(old: Vec<String>, current: Vec<String>) -> Option<(Vec<usize>, i32)> {
    // Process check.
    let first_check = old
        .iter()
        .zip(current.iter())
        .map(|(x, y)| {
            x == y
        })
        .collect::<Vec<bool>>();
    match (old.len(), current.len()) {
        (v1, v2) if v1 == v2 => Some((first_check.iter().enumerate().filter(|(_, &value)| !value).map(|(index, _)| index).collect::<Vec<usize>>(),
1)),
        (v1, v2) if v1 > v2 => Some((first_check.iter().enumerate().filter(|(_, &value)| !value).map(|(index, _)| index).collect::<Vec<usize>>(), -
1)),
        (v1, v2) if v1 < v2 => Some((first_check.iter().enumerate().filter(|(_, &value)| !value).map(|(index, _)| index).collect::<Vec<usize>>(), 0)),
        _ => None,
    }
}

async fn statement(&self, main: Vec<Vec<String>>, old: Vec<String>, current: Vec<String>, columns: Vec<String>, from: String, name:
String) -> String {
    // Process check
    let result_check = Self::check_row(old.clone(), current.clone()).await;
    // function on test only!!
    // let result_check = time_function(|| Self::check_row(old.clone(), current.clone()), "database_check_row").await;
    match result_check {
        Some((res, i)) => {
            // Check value in row not equal.
            match res.is_empty() {
```

```
                true => {
                    self.last(i, main, current, old, columns, vec![from, name]).await
                    // function on test only!!
                    // time_function(|| self.last(i, main, current, old, columns, vec![from, name]), "database_last#1").await
                },
                false => {
                    // Set Row to Update.
                    let result_id = res.clone()
                        .into_iter()
                        .map(|j| main[j][0].clone())
                        .collect::<Vec<String>>();
                    let result_update = res
                        .into_iter()
                        .map(|j| current[j].clone())
                        .collect::<Vec<String>>();

                    // first update
                    for i in 0..result_id.len() {
                        let convert_value: Vec<String> = result_update[i].replace(['(', ')'], "").split(',').map(|s| s.to_string()).collect();
                        let set_query = convert_value.iter().zip(columns[1..].iter()).map(|(x, y)| format!("{} = {}", y, x)).collect::<Vec<String>>().join(",
");
                        sqlx::query(&format!("UPDATE TB_TR_PDPA_AGENT_DATABASE_CHECK SET {} WHERE {} = {}", set_query, columns[0],
result_id[i]))
                            .execute(&self.connection)
                            .await.unwrap();
                    }

                    self.last(i, main, current, old, columns, vec![from, name]).await
                    // function on test only!!
                    // time_function(|| self.last(i, main, current, old, columns, vec![from, name]), "database_last#2").await
                },
            }
        },
        None => "None".to_string()
    }
}

async fn last(&self, status: i32, main: Vec<Vec<String>>, current: Vec<String>, old: Vec<String>, columns: Vec<String>, from_name:
Vec<String>) -> String {
    let from = &from_name[0];
    let name = &from_name[1];
    if status == 1 { // Total old equal Total current.
        current.len().to_string()
    } else if status == -1 { // Total old more than row new.
```

```
            let select_old = main[current.len()..].to_vec();
            for j in select_old {
                let query_delete = format!("DELETE FROM TB_TR_PDPA_AGENT_DATABASE_CHECK WHERE {} = {}",
                    columns[0], // id
                    j[0], // on select id.
                );
                sqlx::query(&query_delete)
                    .execute(&self.connection)
                    .await.unwrap();
            }
            current.len().to_string()
        } else if status == 0 { // Total old less total new.
            let set_value = current[old.len()..].
                iter()
                .map(|v| {
                    let mut seprate = v.replace(['(', ')'], "");
                    seprate.push_str(&format!(", \"{}\"", name));
                    format!("({})", seprate)
                })
                .collect::<Vec<String>>()
                .join(", ");
            sqlx::query(&format!("INSERT INTO TB_TR_PDPA_AGENT_DATABASE_CHECK ({}, {}) VALUES {}", columns[1..].join(","), from, set_value))
                .execute(&self.connection)
                .await.unwrap();
            current.len().to_string()
        } else { // Other case.
            "Other".to_string()
        }
    }

    // Mix type db and check pass enum.
    #[allow(clippy::needless_collect, unused_must_use)]
    async fn manage(&self, pool: PoolType, main_id: String, from: String, table: String, columns: Vec<String>) -> String {
        let mut all_columns = columns.clone();
        all_columns.insert(0, main_id);

        // Delete all not client.
        sqlx::query(&format!("DELETE FROM TB_TR_PDPA_AGENT_DATABASE_CHECK WHERE {} IS NULL",
from.clone())).execute(&self.connection).await.unwrap();

        // From main.
        let query = format!("SELECT {} FROM TB_TR_PDPA_AGENT_DATABASE_CHECK WHERE {} = '{}' ORDER BY {} ASC",
            all_columns.join(","),
            from,
```

```
            table,
            all_columns[0],
        );
        let result_main: Vec<Vec<String>> = sqlx::query(&query).fetch_all(&self.connection).await.unwrap()
            .into_iter()
            .map(|row| {
                let mut result: Vec<String> = vec![];
                for i in &all_columns {
                    result.push(match row.try_get::<String, _>(i.as_str()) {
                        Ok(val) => val,
                        Err(_) => match row.try_get::<i32, _>(i.as_str()) {
                            Ok(val1) => val1.to_string(),
                            Err(_) => "NULL".to_string()
                        },
                    });
                }
                result
            })
            .collect();

        // of client.
        let mut use_table = table.split(':').collect::<Vec<&str>>();
        let use_column = use_table.pop().unwrap().split(',').map(|s| s.to_string()).collect::<Vec<String>>();

        let query_client = format!("SELECT {} FROM {} ORDER BY {} ASC", use_column.join(","), use_table[0], use_column[0]);

        let result_client: Vec<Vec<String>>;
        match pool {
            PoolType::MyPool(pool) => {
                result_client = sqlx::query(&query_client).fetch_all(&pool).await.unwrap()
                    .into_iter()
                    .map(|row| Self::set_query(self._type, use_column.clone(), PoolRow::MyRow(row)))
                    // function on test only!!
                    // .map(|row| time_function(|| Self::set_query(self._type, use_column.clone(), PoolRow::MyRow(row)),
"database_set_query#mysql"))
                    .collect();
            },
            PoolType::OrPool(pool) => {
                match pool.get() {
                    Ok(conn) => {
                        result_client = conn.query(&query_client, &[]).unwrap()
                            .into_iter()
                            .map(|row| Self::set_query(self._type, use_column.clone(), PoolRow::OrRow(row.unwrap())))
                            // function on test only!!
```

```
                        // .map(|row| time_function(|| Self::set_query(self._type, use_column.clone(), PoolRow::OrRow(row.unwrap())),
"database_set_query#oracle"))
                            .collect();
                        conn.close().unwrap();
                    },
                    Err(err) => {
                        println!("[Failed] {}", err);
                        std::process::exit(1);
                    }
                }
            }
        }

        // Main process.
        match result_main.len() {
            0 => {
                // Convert to tuple.
                let value_all: Vec<String> = result_client.clone()
                    .into_iter()
                    .map(|q| {
                        format!("({}, \"{}\")", q.iter().map(|v| format!("\"{}\"", v)).collect::<Vec<String>>().join(","), table)
                    })
                    .collect();
                let query_insert = format!("INSERT INTO TB_TR_PDPA_AGENT_DATABASE_CHECK ({},{}) VALUES {}", columns.join(","), from,
value_all.join(", "));
                sqlx::query(&query_insert).execute(&self.connection).await;
                result_client.len().to_string()
            },
            _ => {
                let old_value: Vec<String> = result_main.clone().into_iter().map(|o| {
                    format!("({})", o[1..].iter().map(|v| format!("\'{}\'", v)).collect::<Vec<String>>().join(","))
                }).collect();
                let new_value: Vec<String> = result_client.clone().into_iter().map(|o| {
                    format!("({})", o.iter().map(|v| format!("\'{}\'", v)).collect::<Vec<String>>().join(","))
                }).collect();
                self.statement(result_main, old_value, new_value, all_columns, from, table).await
                // function on test only!!
                // time_function(|| self.statement(result_main, old_value, new_value, all_columns, from, table), "database_statement").await
            }
        }
    }

    #[allow(unused_must_use)]
    pub async fn build(&self) -> Result<Vec<String>, std::io::Error> {
```

```
        let mut message = vec![];
        let query = format!(
            "SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'TB_TR_PDPA_AGENT_DATABASE_CHECK'
AND TABLE_SCHEMA = '{}' ORDER BY ORDINAL_POSITION",
            env::var("DB_MAIN").unwrap_or_else(|_| "DOL_PDPA".to_string())
        );
        let mut all_field: Vec<String> = sqlx::query(&query)
            .fetch_all(&self.connection)
            .await.unwrap()
            .into_iter()
            .map(|row| row.get::<String, _>("COLUMN_NAME"))
            .collect();

        // Not sure.
        all_field.pop();
        let column_id = all_field.remove(0);
        let from_client = all_field.pop().unwrap();

        for i in &self.tables {
            let size_field = i.split(',').count();
            let select_field = &all_field[0..size_field];
            match self._type {
                1 => {
                    match MySqlPool::connect(&format!("mysql://{}:{}@{}:{}/{}", self.user, self.passwd, self.host, 3306, self.db_name)).await {
                        Ok(pool) => {
                            let result = format!("{}|||{}",
                                i,
                                self.manage(PoolType::MyPool(
                                    pool.clone()),
                                    column_id.to_owned(),
                                    from_client.to_owned(),
                                    i.to_string(),
                                    select_field.to_vec()
                                ).await
                            );
                            // function on test only!!
                            // let result = format!("{}|||{}",
                            //     i,
                            //     time_function(||
                            //         self.manage(PoolType::MyPool(
                            //             pool.clone()),
                            //             column_id.to_owned(),
                            //             from_client.to_owned(),
                            //             i.to_string(),
```

```
                    //              select_field.to_vec()
                    //          ),
                    //          "database_manage#mysql"
                    //      ).await
                    // );
                    pool.close();
                    message.push(result);
                },
                Err(err) => {
                    println!("[Failed] {}", err);
                    std::process::exit(1);
                },
            };
        },
        0 => {
            // match PoolBuilder::new(&self.user, &self.passwd, &format!("//{}:{}/{}", self.host, 1521,
self.db_name)).max_connections(10).build() {
            match PoolBuilder::new(&self.user, &self.passwd, &format!("//{}:{}/{}", self.host, 1521, self.db_name)).build() {
                Ok(pool) => {
                    let result = format!("{}|||{}",
                        i,
                        self.manage(PoolType::OrPool(
                            pool.clone()),
                            column_id.to_owned(),
                            from_client.to_owned(),
                            i.to_string(),
                            select_field.to_vec()
                        ).await
                    );
                    // function on test only!!
                    // let result = format!("{}|||{}",
                    //     i,
                    //     time_function(|| self.manage(PoolType::OrPool(
                    //         pool.clone()),
                    //         column_id.to_owned(),
                    //         from_client.to_owned(),
                    //         i.to_string(),
                    //         select_field.to_vec()
                    //         ),
                    //         "database_manage#oracle"
                    //     ).await
                    // );
                    pool.close(&CloseMode::Default);
                    message.push(result);
```

```
                    },
                    Err(err) => {
                        println!("[Failed] {}", err);
                        std::process::exit(1);
                    },
                }
            },
            _ => {
                println!("[Error] Type database not support");
                std::process::exit(1);
            }
        }
    }
    Ok(message)
  }
}
```

รายละเอียชุดข้อมูลโค้ดในส่วนตัวข้อมูลจราจรคอมพิวเตอร์ทั้ง 2 รูปแบบของลูกข่าย

```
use get_if_addrs::{get_if_addrs, IfAddr::V6, IfAddr::V4};
use sqlx::{MySqlPool, Row};
use chrono::Local; use async_ftp::FtpStream;
use syslog::{Facility, Formatter3164};

use std::env;
use std::fs::{
    read_to_string,
    remove_file,
    write,
    metadata,
    create_dir,
    read_dir,
    OpenOptions,
    File,
    DirEntry,
};
use std::io::{
    Read,
    BufReader,
    BufRead,
    BufWriter,
    Write,
    Cursor,
};
use std::process::{
```

```
    Command,
    exit,
    Stdio,
};

use crate::model::MyInterface;

// use test
// use crate::module::test::*;

#[derive(Debug)]
#[allow(dead_code)]
pub struct TaskSniffer {
    connection: MySqlPool,
    interface: String,
    host: String,
    user: String,
    passwd: String,
    mode: i32,
    directory: String,
    status: i32,
}

impl TaskSniffer {
    pub fn new(connection: MySqlPool, details: Vec<String>, interface: String) -> Self {
        let user: String = "ftpuser".to_string();
        let passwd: String = "ftpuser".to_string();
        let mode: i32 = details[0].parse::<i32>().unwrap_or(-1);
        let host: String = match mode {
            0 => format!("{}:{}", env::var("HOST").unwrap_or_else(|_| "127.0.0.1".to_string()), 21),
            1 => format!("{}:{}", env::var("HOST").unwrap_or_else(|_| "127.0.0.1".to_string()), 514), // Tcp: 601, Udp 514.
            _ => {
                println!("[Failed] Unknow mode of sniffer");
                exit(1);
            }
        };
        let directory: String = details[1].to_owned();
        let status = -1;
        Self { connection, interface, host, user, passwd, mode, directory, status }
    }

    async fn set_interface(name: String) -> MyInterface {
        let mut ip = MyInterface::default();
        if let Ok(interfaces) = get_if_addrs() {
```

```
            for interface in interfaces {
                if !interface.is_loopback() && interface.name == name && interface.ip().is_ipv4() {
                    ip = match interface.addr {
                        V4(ref value) => MyInterface {
                            ip: value.ip.to_string(),
                            netmask: value.netmask.to_string(),
                            broadcast: value.broadcast.map(|b| b.to_string()),
                        },
                        V6(ref value) => MyInterface {
                            ip: value.ip.to_string(),
                            netmask: value.netmask.to_string(),
                            broadcast: value.broadcast.map(|b| b.to_string()),
                        },
                    };
                    break
                }
            }
        }
        ip
    }

    async fn status(&self) -> bool {
        let query = format!("SELECT * FROM TB_TR_PDPA_AGENT_MANAGE WHERE agm_token = \"{}\"", env::var("TOKEN").unwrap());
        let result = sqlx::query(&query).fetch_one(&self.connection).await.unwrap();
        if result.is_empty() {
            remove_file(".env").unwrap();
            println!("[Error] Token expired or has bee deleted");
            exit(1);
        } else {
            match result.try_get::<i32, _>("agm_status") {
                Ok(value) => {
                    let mut env_file_content = read_to_string(".env").unwrap();
                    let new_env = ("STATUS", value.to_string());
                    let current_env = env::var("STATUS").unwrap();
                    env::set_var(new_env.0, new_env.1.clone());
                    env_file_content = env_file_content.replace(&format!("{}=\"{}\"\n", "STATUS", current_env), &format!("{}=\"{}\"\n", new_env.0,
new_env.1));
                    write(".env", env_file_content).unwrap();
                    match value {
                        1 => true,
                        0 => false,
                        _ => {
                            println!("[Error] Unknow");
                            exit(1);
```

```
                }
            }
        },
        Err(err) => {
            println!("[Error] Can't get status: {}", err);
            exit(1);
        }
    }
  }
}

fn sort(dir_path: &str) -> Result<Vec<DirEntry>, Box<dyn std::error::Error>> {
    let mut list: Vec<_> = read_dir(dir_path).unwrap()
        .filter_map(|entry| entry.ok())
        .collect();
    list.sort_by(|a, b| {
        let a_time = a.metadata().unwrap().created().unwrap();
        let b_time = b.metadata().unwrap().created().unwrap();
        a_time.cmp(&b_time)
    });
    Ok(list)
}

#[allow(unused_must_use)]
async fn create_file(ip: String, dir: String, buffer: String) -> String {
    // Setup to create.
    let now = Local::now().format("%H.0,%Y-%m-%d").to_string();
    let set_directory = match dir.chars().nth(dir.chars().count() - 1) {
        Some('/') => dir.clone(),
        _ => format!("{}/", dir.clone()),
    };
    let name = format!("{},{}.snf", ip, now);
    let dir_file = format!("{}{}", set_directory, name);

    // Check directory.
    if !metadata(dir.clone()).unwrap().is_dir() {
        create_dir(dir).unwrap();
    }

    // Insert to file.
    if metadata(dir_file.clone()).is_ok() {
        // let mut file = OpenOptions::new().append(true).open(dir_file).unwrap();
        let mut file = OpenOptions::new().write(true).append(true).open(dir_file).unwrap();
        // let mut writer = BufWriter::new(file);
```

```
            // writer.write_all(buffer.as_bytes()).unwrap();
            writeln!(file, "{}", buffer).unwrap();
        }else{
            let file = File::create(dir_file).unwrap();
            let mut writer = BufWriter::new(file);
            writer.write_all(buffer.as_bytes()).unwrap();
        }

        // List in path and remove old file because new file.
        let mut list_in = Self::sort(&set_directory).unwrap().iter().map(|s| s.file_name().to_string_lossy().to_string()).collect::<Vec<String>>();
        // function on test only!!
        // let mut list_in = time_function(|| Self::sort(&set_directory).unwrap().iter().map(|s|
s.file_name().to_string_lossy().to_string()).collect::<Vec<String>>(), "sniffer_sort");
        list_in.retain(|s| !s.contains(".DS_Store"));

        if list_in.len() > 1 {
            for i in list_in.iter().take(list_in.len() - 1) {
                remove_file(format!("{}{}", set_directory, i)).unwrap();
            }
        }

        // Return.
        name
    }

    // Mode 0 (FTP).
    #[allow(unused_must_use)]
    async fn send_ftp(&self, name: String) -> Result<(), Box<dyn std::error::Error>> {
        // Setup varriable to use.
        // local file by default have one file.
        let mut local_file = read_dir(self.directory.clone()).unwrap().map(|s|
s.unwrap().file_name().to_string_lossy().to_string()).collect::<Vec<String>>();
        local_file.retain(|s| !s.contains(".DS_Store"));
        let set_dir = &name.clone().split(',').map(|s| s.to_string()).collect::<Vec<String>>()[0];

        // Start connect and process another.
        let mut ftp_stream = FtpStream::connect(&self.host).await?;
        ftp_stream.login(&self.user, &self.passwd).await?;

        // List directory in remote and check same client.
        let mut remote_dir = ftp_stream.list(Some("/")).await?
            .iter()
            .map(|entry| {
                let en_len: usize = entry.clone().split_whitespace().map(|e| e.to_string()).collect::<Vec<String>>().len();
```

```
            &entry.split_whitespace().map(|e| e.to_string()).collect::<Vec<String>>()[en_len - 1].to_string() == set_dir
        })
        .collect::<Vec<bool>>();
    remote_dir.retain(|&b| b);

    if remote_dir.is_empty() {
        ftp_stream.mkdir(set_dir).await?;
    }

    ftp_stream.cwd(set_dir).await?;

    // (PUT) a file from local to the current working directory of the server.
    // Today (2023-06-10) not check length of file 2 way.
    let mut buffer = Vec::new();
    let full_file = format!("{}{}", self.directory, local_file[0]);
    File::open(full_file).expect("Failed to open file").read_to_end(&mut buffer).expect("Failed to read file");
    let mut content = Cursor::new(buffer);

    ftp_stream.put(
        &name,
        &mut content
    ).await?;

    // Disconnect FTP server.
    ftp_stream.quit().await;

    // Return.
    Ok(())
}

// Mode 1 (Syslog).
async fn send_syslog(&self, event: String) -> Result<(), Box<dyn std::error::Error>> {
    // Setup format log.
    let formatter = Formatter3164 {
        facility: Facility::LOG_USER,
        hostname: None,
        process: "agent_sniffer".into(),
        pid: 0,
    };

    // Not sure.
    match syslog::tcp(formatter, &format!("{}:{}", self.host, "514")) {
        // syslog::udp(formatter, "127.0.0.1:5051", &self.host)
        Ok(mut writer) => {
```

```
                writer.info(&event)?;
            },
            Err(e) => {
                println!("Failed to connect to syslog server: {}", e);
                tokio::time::sleep(std::time::Duration::from_secs(5)).await;
            }
        };

        Ok(())
    }

    pub async fn run(&self) {
        let found = Self::set_interface(self.interface.clone()).await;
        // function on test only!!
        // let found = time_function(|| Self::set_interface(self.interface.clone()), "sniffer_set_interface").await;
        match found != MyInterface::default() {
            true => {
                loop {
                    // Core.
                    let mut dump = Command::new("tcpdump")
                        .arg("-i")
                        .arg(&self.interface)
                        .arg("-l")
                        .stdout(Stdio::piped())
                        .spawn()
                        .expect("Failed to execute command");

                    // Status.
                    let status = self.status().await;
                    // function on test only!!
                    // let status = time_function(|| self.status(), "sniffer_status").await;
                    match status {
                        true => {
                            // Output.
                            let stdout = dump.stdout.take().unwrap();
                            let reader = BufReader::new(stdout);
                            for line in reader.lines().flatten() {
                                // function on test only!!
                                match self.mode {
                                    0 => {
                                        let result = Self::create_file(found.ip.clone(), self.directory.to_owned(), line.clone()).await;
                                        // let result = time_function(|| Self::create_file(found.ip.clone(), self.directory.to_owned(), line.clone()),
"sniffer_create_file").await;
                                        // function on test only!!
```

```
                                // match time_function(|| self.send_ftp(result), "sniffer_send_ftp").await {
                                match self.send_ftp(result).await {
                                    Ok(_) => self.set_history().await,
                                    // function on test only!!
                                    // Ok(_) => {
                                    //     time_function(|| self.set_history(), "sniffer_set_history").await
                                    // },
                                    Err(err) => println!("{:?}", err),
                                }
                            },
                            1 => match self.send_syslog(line).await {
                            // function on test only!!
                            // 1 => match time_function(|| self.send_syslog(line), "sniffer_send_syslog").await {
                                Ok(_) => self.set_history().await,
                                // function on test only!!
                                // Ok(_) => time_function(|| self.set_history(), "sniffer_set_history").await,
                                Err(err) => println!("{:?}", err),
                            },
                            _ => {
                                println!("[Error] Not found mode its");
                                exit(1);
                            }
                        }
                    }
                },
                false => dump.kill().unwrap(),
            }
        }
    },
    false => {
        println!("[Error] Can't interface {} to capture event", self.interface);
        exit(1);
    }
  }
}

#[allow(unused_must_use)]
async fn set_history(&self) {
    let query = format!("SELECT agm_id FROM TB_TR_PDPA_AGENT_MANAGE WHERE agm_token = \"{}\"", env::var("TOKEN").unwrap());
    let selected: i32 = sqlx::query(&query)
        .fetch_one(&self.connection)
        .await.unwrap().get(0);

    let query_select = format!("SELECT * FROM TB_TR_PDPA_AGENT_LISTEN_HISTORY WHERE agm_id = \"{}\"", selected.clone());
```

```
        let result_history = match sqlx::query(&query_select)
            .fetch_one(&self.connection)
            .await {
                Ok(res) => res.get::<i64, usize>(1).to_string(),
                Err(_) => "".to_string(),
            };

        match result_history.is_empty() {
            true => sqlx::query(&format!("INSERT INTO TB_TR_PDPA_AGENT_LISTEN_HISTORY (agm_id) VALUE ({})",
selected)).execute(&self.connection).await.unwrap(),
            false => sqlx::query(&format!("UPDATE TB_TR_PDPA_AGENT_LISTEN_HISTORY SET _get_ = NOW() WHERE agm_id = {}",
selected)).execute(&self.connection).await.unwrap(),
        };
    }
}
```

รายละเอียดชุดข้อมูลโค้ดในส่วนการทดสอบ

```
use std::time::{SystemTime, Duration, Instant};
use sysinfo::{CpuExt, ProcessExt, System, SystemExt};

// use crate::model::{AgentStore, AgentManage, AgentHistory};

#[allow(dead_code)]
fn get_fn_name<F>(_: F) -> &'static str
where
    F: Fn(),
{
    std::any::type_name::<F>()
}

pub fn time_function<F: FnOnce() -> T, T>(func: F, name: &'static str) -> T {
    let start_time = SystemTime::now();
    let result = func();
    let end_time = SystemTime::now();
    let duration = end_time.duration_since(start_time).unwrap();
    println!("Time Fn #{}# : {:?} miliseconds", name, duration.as_millis());
    result
}

pub fn benchmark_env_usage(intervel: Duration) -> (f32, f64, f64, f64) {
    let mut system = System::new_all();

    system.refresh_all();
```

```
    let start_cpu_usage = system.global_cpu_info().cpu_usage();
    let start_ram_usage = system.used_memory();

    let start_time = Instant::now();

    while Instant::now() - start_time < intervel {
        std::thread::sleep(Duration::from_millis(100));
        system.refresh_all();
    }

    let end_cpu_usage = system.global_cpu_info().cpu_usage();
    let elapsed_time = start_time.elapsed().as_secs_f32();

    let end_ram_usage = system.used_memory();

    let cpu_usage = ((end_cpu_usage - start_cpu_usage) / elapsed_time) * -1.0 / 100.0;
    let ram_usage = (end_ram_usage - start_ram_usage) as f64 * 1e-6;
    let find_disk_usage: Vec<(bool, u64, u64)> = system.processes()
        .iter()
        .map(|(_, process)|
            match process.name() == "agent_client" {
                true => {
                    let dsk = process.disk_usage();
                    (true, dsk.total_read_bytes, dsk.total_written_bytes)
                },
                false => (false, 0, 0)
            })
        .collect();
    let (read_usage, write_usage) = find_disk_usage.iter().find(|&&(b, _, _)| b).map(|(_, r, w)| (*r as f64 * 1e-6, *w as f64 * 1e-6)).unwrap();

    (cpu_usage, ram_usage, read_usage, write_usage)
}
```

ประวัติผู้วิจัย

ชื่อ-สกุล: นายกิตติภูมิ จันพุดซา

วันเดือนปีเกิด: 15 กันยายน 2543

ที่อยู่ปัจจุบัน: บ้านเลขที่ 50 หมู่ 4 ตำบล ป่าตึง อำเภอ แม่จัน จังหวัด เชียงราย รหัสไปรษณีย์ 57110

เบอร์โทร: 099-387-6320

อีเมล: kittiphum1230@hotmail.co.th

ประวัติการศึกษา: สำเร็จการศึกษาระดับมัธยมตอนปลาย(เทียบเท่า)ศึกษาจากวิทยาลัยอาชีวศึกษาเชียงราย เมื่อ พ.ศ. 2561