เขียนโค้ดอย่างไรให้มั่นคงปลอดภัยตามมาตรฐานสากล ISO/IEC 27001:2022

How to Secure Coding for Application with ISO/IEC 27001:2022

เลอพงศ์ แก้วอินทร์[1] และ กรรณิกา ละมั่งทอง[1]

Lerpong Kaewin[1] and Kannika Lamungthong[1]

[1]ฝ่ายสารสนเทศ คณะแพทยศาสตร์ศิริราชพยาบาล มหาวิทยาลัยมหิดล 10700

[1]Siriraj Information Technology Department, Faculty of Medicine, Mahidol University, 10700

*Corresponding Author: lerpong.kaw@mahidol.ac.th

Received 28 มกราคม 2568; Revised 21 เมษายน 2568; Accepted 26 เมษายน 2568

บทคัดย่อ

มาตรฐานสากล ISO 27001 เป็นมาตรฐานสากลสำหรับระบบบริหารความมั่นคงปลอดภัยสารสนเทศ (Information Security Management System: ISMS) มุ่งเน้นให้องค์กรปกป้องข้อมูลสำคัญขององค์กรได้อย่างเป็นระบบซึ่งองค์กรสามารถนำกรอบการปฏิบัติงานตามหลักของ ISO 27001 มาใช้โดยการกำหนดให้องค์กรดำเนินการระบุความเสี่ยงตลอดจนการบริหารจัดการความเสี่ยงด้านความมั่นคงปลอดภัยสารสนเทศและมีประสิทธิภาพ

ปัจจุบันมาตรฐานสากล ISO 27001 ได้ปรับเปลี่ยนเวอร์ชันจาก ISO/IEC 27001:2013 มาเป็นเวอร์ชัน ISO/IEC 27001:2022 เพื่อให้ครอบคลุมในส่วนของความปลอดภัยทางไซเบอร์และการปกป้องความเป็นส่วนตัว โดยเพิ่ม 11 มาตรการควบคุมความมั่นคงปลอดภัยสารสนเทศให้ครอบคลุมมิติต่าง ๆ มากขึ้น ซึ่งบทความนี้ได้นำมาตรการควบคุมสำหรับการเขียนโค้ดให้มีความมั่นคงปลอดภัยมาอธิบายเพื่อให้ผู้อ่านและผู้ที่ประกอบอาชีพเกี่ยวกับการพัฒนาแอปพลิเคชันได้มีความเข้าใจเกี่ยวกับการเขียนโค้ดให้มีความมั่นคงปลอดภัยเพิ่มมากขึ้น โดยนำ 8 ประเด็นสำคัญมาใช้สำหรับเป็นแนวทางการเขียนโค้ดให้มีความมั่นคงปลอดภัยตั้งแต่เริ่มต้นจนสิ้นสุดกระบวนการพัฒนาแอปพลิเคชัน

คำหลัก: *การเขียนโค้ดให้มีความมั่นคงปลอดภัย; ระบบบริหารความมั่นคงปลอดภัยของสารสนเทศ; ข้อกำหนดมาตรฐาน ISO/IEC 27001:2022*

Abstract

The international standard ISO/IEC 27001 is a globally recognized framework for Information Security Management Systems (ISMS), aiming to help organizations systematically protect their critical information assets. Organizations can adopt the ISO/IEC 27001 framework by identifying and managing information security risks effectively.

Recently, ISO/IEC 27001 has been updated from the 2013 version to ISO/IEC 27001:2022 to better address cybersecurity and privacy protection. The revised standard introduces 11 new information security control measures that broaden the scope of protection across multiple dimensions. This article focuses on the security control related to secure coding practices. It aims to enhance the understanding of secure coding among readers and professionals in application development. The article presents eight key

principles as a guideline for writing secure code throughout the entire application development lifecycle—from initial design to deployment.

Keywords: *Secure Coding; Information Security Management System; ISO/IEC 27001:2022*

1. บทนำ

ในโลกยุคดิจิทัลแอปพลิเคชันคือหัวใจสำคัญของการแข่งขันทางธุรกิจช่วยให้องค์กรหรือหน่วยงานต่าง ๆ สามารถดำเนินการทางธุรกิจได้อย่างมีประสิทธิภาพ แต่สำหรับโลกยุคดิจิทัลนั้นการโจมตีและภัยคุกคามทางไซเบอร์เกิดขึ้นอยู่ตลอดเวลา อาทิ มัลแวร์ (Malware) ไวรัสคอมพิวเตอร์ (Computer Virus) บอท (Bot) และการโจมตีในรูปแบบอื่น ๆ อีกมากมายล้วนมีพัฒนาของการโจมตีอย่างต่อเนื่อง ระดับความรุนแรงเพิ่มขึ้น ภัยคุกคามทางไซเบอร์เหล่านี้สามารถเข้าไปแฝงตัวอยู่ในได้ทุกที่ทุกเวลา ทำให้ข้อมูลสำคัญทางธุรกิจที่องค์กรหรือหน่วยงานเก็บรักษาไว้สามารถถูกโจมตี สร้างความเสียหายได้เพียงชั่วข้ามคืน ภาพลักษณ์และชื่อเสียงที่สั่งสมมาอาจพังทลายภายในพริบตา

องค์กรหรือหน่วยงานจะทำอย่างไรจึงจะสามารถป้องกันภัยคุกคามทางไซเบอร์และสร้างมั่นใจให้กับผู้มีส่วนได้ส่วนเสีย (Stakeholder) ตั้งแต่จุดเริ่มต้นของการโจมตี คำตอบคือ “การเขียนโค้ดให้มีความมั่นคงปลอดภัย” เพื่อให้มีความทนทานต่อการถูกโจมตีจากผู้ไม่ประสงค์ดี ซึ่งการเขียนโค้ดให้มีความมั่นคงปลอดภัยได้รับการปรับปรุงและเพิ่มเติมเพื่อเป็นมาตรการควบคุมใหม่ 11 มาตรการ ในมาตรฐานความมั่นคงปลอดภัยสารสนเทศ

บทความนี้นำเสนอหลักการและมาตรการควบคุมในส่วนของการเขียนโค้ดให้มีความมั่นคงปลอดภัยตามมาตรฐานสากล ISO/IEC 27001:2022 พร้อมรวบรวมวิธีการคิดและวิธีการเขียนโค้ดสำหรับผู้พัฒนาแอปพลิเคชันได้นำไปใช้เป็นแนวทางสำหรับการเขียนโค้ดให้มีความมั่นคงปลอดภัยและสามารถป้องกันการโจมตีทางไซเบอร์ได้

2. มาตรฐานสากล ISO/IEC 27001:2022

มาตรฐานสากล ISO/IEC 27001 ประกาศใช้ครั้งแรกเมื่อปี ค.ศ. 2005 และปรับเปลี่ยนมาตรฐานครั้งแรกเมื่อปี ค.ศ. 2013 และได้รับการปรับเปลี่ยนครั้งล่าสุดในปี ค.ศ. 2022 โดยมาตรฐานสากล ISO/IEC 27001 เป็นมาตรฐานสากลที่เน้นการจัดการความมั่นคงปลอดภัยสารสนเทศในองค์กร ซึ่งใช้กรอบการทำงานและแนวทางสำหรับการปรับปรุงความปลอดภัยของข้อมูลโดยการประเมินความเสี่ยงและการบริหารจัดการความเสี่ยงเพื่อป้องกัน ควบคุม ลดความเสี่ยงที่เกี่ยวข้องกับข้อมูลและความปลอดภัยของข้อมูลสารสนเทศ หากบุคลากรขององค์กรหรือหน่วยงานมีความตระหนักและปฏิบัติงานได้สอดคล้องตามแนวทางที่กำหนดไว้ของมาตรฐาน ISO/IEC 27001:2022 ก็จะเป็นส่วนเสริมความแข็งแกร่งให้กับมาตรการรักษาความมั่นคงปลอดภัยสารสนเทศ สร้างความมั่นใจและความน่าเชื่อถือให้กับลูกค้า พนักงาน คู่ค้า และผู้มีส่วนได้ส่วนเสียขององค์กรหรือหน่วยงานได้อีกทางหนึ่งด้วย [1]

มาตรฐานสากล ISO/IEC 27001 เป็นข้อกำหนดในกลุ่มมาตรฐานความมั่นคงปลอดภัยสารสนเทศที่มุ่งเน้นการปกป้องข้อมูลสารสนเทศ 3 ประการ ได้แก่ การรักษาความลับ (Confidentiality) ความถูกต้องสมบูรณ์ (Integrity) และความพร้อมใช้งาน (Availability) [2]

มาตรฐานสากล ISO/IEC 27001 มีมาตรฐานที่เกี่ยวข้องอยู่หลายฉบับ อาทิ ISO/IEC 27000:2018 กล่าวถึงนิยามและคำศัพท์, ISO/IEC 27001:2022 กล่าวถึงข้อกำหนดการบริหารจัดการความมั่นคงปลอดภัยสารสนเทศ และ ISO/IEC 27002:2022 กล่าวถึงหลักปฏิบัติสำหรับการควบคุมความปลอดภัยของข้อมูล โดยการเปลี่ยนแปลงของมาตรฐานสากล ISO/IEC 27001:2013 มาเป็นมาตรฐานสากล ISO/IEC 27001:2022 นั้น มีการปรับเปลี่ยนเล็กน้อยในส่วนของข้อกำหนด 4.2 ความเข้าใจความจำเป็นความคาดหวังของผู้มีส่วนได้ส่วนเสีย, ข้อกำหนด 6.2 วัตถุประสงค์ความมั่นคงปลอดภัยสารสนเทศและแผนงานเพื่อทำให้บรรลุ, ข้อกำหนด 6.3 การวางแผนการเปลี่ยนแปลง, ข้อกำหนด 8.1 การวางแผนและควบคุมการดำเนินการ, ข้อกำหนด 9.2 แยกข้อกำหนดออกเป็น 9.2.1 และ 9.2.2 เนื้อหายังคงเหมือนเดิม และข้อกำหนด 9.3 แยกออกเป็น 3 ส่วนย่อย คือ 9.3.1 บททั่วไป, 9.3.2 ข้อมูลนำเข้าการทบทวน และ 9.3.3 ผลลัพธ์การทบทวน (ซึ่งในส่วนนี้

เพิ่มเติม 9.3.2 c) คือ การเปลี่ยนแปลงความต้องการและความคาดหวังของผู้มีส่วนได้เสียที่เกี่ยวข้องกับระบบบริหารความมั่นคงปลอดภัยสารสนเทศซึ่งต้องมีการทบทวนโดยผู้บริหาร

ในส่วนของการเปลี่ยนแปลงหลักที่เกี่ยวข้องโดยองค์กรหรือหน่วยงานต้องตระหนัก คือ รายละเอียดมาตรการควบคุมที่อ้างอิงใน Annex A โดยมีการปรับเปลี่ยนโครงสร้าง รวมไปถึงมีการปรับปรุงให้สอดคล้องกับมาตรฐาน ISO/IEC 27002:2022 ซึ่งเป็นการปรับเปลี่ยนและรวมตัวควบคุม (Control) ที่เกี่ยวข้องหรือเชื่อมโยงถึงกันไว้ด้วยกัน จากเดิมมาตรการควบคุมของมาตรฐานสากล ISO/IEC 27001:2013 มีจำนวน 114 รายการ ในส่วนของมาตรฐานสากล ISO/IEC 27001:2022 มาตรการควบคุมจะเหลือเพียง 93 รายการ [3]

มาตรการควบคุมใน Annex A ที่เพิ่มขึ้นสำหรับมาตรฐานสากล ISO/IEC 27001:2022 มีจำนวนทั้งสิ้น 11 รายการ ดังนี้

- A5.7 การวิเคราะห์เชิงลึกด้านภัยคุกคาม (Threat Intelligence)
- A5.23 การรักษาความปลอดภัยข้อมูลในการใช้บริการคลาวด์ (Information Security for Use of Cloud Services)
- A5.30 ความพร้อมด้านเทคโนโลยีสารสนเทศและการสื่อสาร (ICT Readiness for Business Continuity)
- A7.4 การเฝ้าระวังความมั่นคงปลอดภัยทางกายภาพ (Physical Security Monitoring)
- A8.9 การจัดการการตั้งค่า (Configuration Management)
- A8.10 การลบข้อมูล (Information Deletion)
- A8.11 การซ่อนข้อมูล (Data Masking)
- A8.12 การป้องกันข้อมูลรั่วไหล (Data Leakage Prevention)
- A8.16 การเฝ้าติดตามกิจกรรม (Monitoring Activities)
- A8.23 การกรองเว็บ (Web Filtering)
- A8.28 การเขียนโค้ดให้มีความมั่นคงปลอดภัย (Secure Coding)

3. การเขียนโค้ดให้มีความมั่นคงปลอดภัย (Secure Coding) คืออะไร

การเขียนโค้ดให้มีความมั่นคงปลอดภัย (Secure Coding) คือ แนวทางการเขียนโค้ดให้มีความทนทานต่อการถูกโจมตีจากผู้ไม่ประสงค์ดี โดยแอปพลิเคชันนั้นจะต้องได้รับการออกแบบและพัฒนาตลอดจนได้รับการตรวจสอบแล้วว่าจะไม่สามารถถูกโจมตีหรือถูกเจาะด้วยเครื่องมือหรือภัยคุกคามที่เป็นที่รู้จักได้ง่าย สำหรับแอปพลิเคชันที่ไม่มีความมั่นคงปลอดภัยสามารถถูกเจาะได้ด้วยหลากหลายวิธีการ ซึ่งวิธีการต่าง ๆ มุ่งเน้นเพื่อเข้าถึงและควบคุมเครื่องคอมพิวเตอร์แม่ข่าย (Server) หรือเครื่องคอมพิวเตอร์ลูกข่าย (Client) ผ่านทางช่องโหว่ของแอปพลิเคชันต่าง ๆ อาทิ การปลอมตัวเป็นผู้ใช้งานแล้วปล่อยมัลแวร์เพื่อเจาะเข้าระบบ การปฏิเสธการให้บริการ (Denial of Service) การขโมยข้อมูลสำคัญขององค์กร

ดังนั้นการพัฒนาแอปพลิเคชันให้มีความปลอดภัยไม่ใช่เพียงแค่การเพิ่มคุณสมบัติ (Feature) ด้านความปลอดภัย (Security) เท่านั้น แต่ต้องพิจารณาตั้งแต่กระบวนการเริ่มต้นด้วยการรวบรวมมิติต่าง ๆ ที่เกี่ยวกับความมั่นคงปลอดภัยสารสนเทศ (Information Security) เทคนิค วิธีการและเทคโนโลยีที่เลือกใช้ การควบคุมในทุกขั้นตอนการสร้างและดูแลบำรุงรักษาแอปพลิเคชัน หรืออาจกล่าวได้ว่า การเขียนโค้ดให้มีความมั่นคงปลอดภัยเป็นควบคุมตั้งแต่กระบวนออกแบบจนไปสิ้นสุดถึงกระบวนการนำแอปพลิเคชันไปใช้งาน โดยปัจจุบันมีมาตรฐานการเขียนโค้ดที่ปลอดภัยและคู่มือความปลอดภัยในการเข้ารหัสหลายฉบับที่ใช้กันอย่างแพร่หลาย อาทิ แนวทางการเข้ารหัสที่ปลอดภัยของ OWASP และมาตรฐานการเข้ารหัส SEI CERT สำหรับบทความนี้ได้รวบรวมแนวทางสำหรับการเขียนโค้ดให้มีความมั่นคงปลอดภัย 8 แนวทาง [4] เพื่อใช้สำหรับควบคุมการเขียนโค้ดให้สอดคล้องกับ Annex A8.28 การเขียนโค้ดให้มีความมั่นคงปลอดภัย ดังนี้

3.1. การออกแบบแอปพลิเคชันให้มีความปลอดภัย (Security Design)

สิ่งสำคัญที่สุดสำหรับองค์กรหรือหน่วยงานที่ต้องการนำแอปพลิเคชันต่าง ๆ เข้าไปช่วยในการดำเนินกิจการขององค์กร คือ การออกแบบแอปพลิเคชันให้มีความปลอดภัย ซึ่งแนวทางนี้จะให้ผลตอบแทนในระยะยาว โดยเป็นการ

บรรเทาความเสี่ยงและลดต้นทุนทางเทคนิคที่จะเกิดขึ้นในอนาคตหากถูกเจาะระบบหรือถูกโจมตีทางไซเบอร์ สำหรับการออกแบบแอปพลิเคชันให้มีความปลอดภัย ควรทำการวิเคราะห์การเขียนโค้ดต้นฉบับตลอดวงจรชีวิตการพัฒนาระบบ (Software Development Life Cycle: SDLC) และควรนำระบบอัตโนมัติด้านความปลอดภัยมาพิจารณาเพื่อใช้ประกอบการเขียนโค้ด [5] สำหรับแนวทางการออกแบบแอปพลิเคชันให้มีความปลอดภัย จะต้องคำนึงถึงชุดข้อมูล (Sessions) การเข้าถึงทางเดียว (Single Access Point) บทบาทของผู้ใช้ (Role) การจำกัดการมองเห็น (Limited View) และการกำหนดสิทธิ์เข้าถึงให้น้อยที่สุด (Least Privileged Access)

นอกจากนี้ยังต้องทบทวนหลักการต่าง ๆ ที่สามารถนำมาใช้ในการออกแบบแอปพลิเคชันให้มีความปลอดภัย [6] ได้แก่

- การออกแบบแอปพลิเคชันให้มีความปลอดภัยให้เรียบง่ายและมีขนาดไม่ใหญ่จนเกินไป (Economy of Mechanism)
- การออกแบบให้แอปพลิเคชันตัดสินใจเรื่องการควบคุมการเข้าถึง (Fail-safe Default) โดยใช้หลัก Whitelist และ Blacklist ซึ่งหลัก Whitelist ได้แก่ ให้ใครทำอะไรบ้าง เข้าถึงแหล่งข้อมูลอะไร และเข้าใช้งานด้วยวิธีใด หากไม่ตรงตามหลักนี้ แอปพลิเคชันจะไม่อนุญาตให้ผู้ใช้ใช้งานได้ สำหรับการใช้หลัก Blacklist เป็นการกำหนดไม่ให้ใครทำอะไรบ้าง ในระยะยาวการดูแลบำรุงรักษาแอปพลิเคชันจะทำได้ลำบากและมีความยุ่งยากในการบริหารจัดการ เนื่องจากมีหลากหลายวิธีที่จะหลีกเลี่ยงหลักการนี้ ซึ่งผู้ดูแลระบบต้องทบทวนและเพิ่มเติมเข้าไปใน Blacklist อย่างสม่ำเสมอ
- การตรวจสอบสิทธิ์ก่อนการเข้าถึง (Complete Mediation) เป็นการตรวจสอบสิทธิ์ของผู้ใช้แอปพลิเคชันก่อนที่จำเข้าถึงฐานข้อมูล (Database), ตาราง (Table) และรายการ (Record) ของข้อมูลที่ต้องการ
- การให้สิทธิ์ระดับต่ำสุด (Least privilege) เป็นการออกแบบโดยกำหนดให้ผู้ใช้แอปพลิเคชันได้รับสิทธิ์ในระดับต่ำสุดเท่าที่จำเป็นต่อการทำงาน เพื่อลดความเสี่ยงในหลากหลายรูปแบบและช่วยลดระดับความรุนแรงของความเสี่ยงได้อีกด้วย
- การจำกัดการใช้งานร่วมกัน (Least Common Mechanism) เป็นการออกแบบแอปพลิเคชันให้จำกัดจำนวนการใช้งาน การเข้าถึงทรัพยากรต่าง ๆ พร้อม ๆ กันของผู้ใช้ เพื่อป้องกันกรณีที่หากเกิดปัญหาขึ้นกับทรัพยากรดังกล่าว ก็จะกระทบกับผู้ใช้ในวงที่จำกัด
- การออกแบบแอปพลิเคชันให้ง่ายต่อการใช้งาน (Psychological Acceptability) ซึ่งจะไม่เป็นอุปสรรคต่อการตั้งค่าการใช้งาน ซึ่งหากออกแบบแอปพลิเคชันให้มีความซับซ้อน จะส่งผลกระทบและเป็นอุปสรรคต่อการตั้งค่าการใช้งาน และผู้ใช้ก็จะไม่ยอมใช้งานแอปพลิเคชันนั้น
- การจัดเก็บบันทึกการเข้าถึงแอปพลิเคชัน (Compromise Recording) เป็นการออกแบบแอปพลิเคชันให้มีการจัดเก็บบันทึกการเข้าถึงหรือการใช้งานแอปพลิเคชัน อาทิ การเข้าใช้งานระบบ (Login), การเข้าถึงข้อมูล (Access), การแก้ไขข้อมูล (Modify) เพื่อตรวจสอบกิจกรรมที่ไม่พึงประสงค์ และสามารถค้นหาร่องรอยย้อนหลังได้

3.2. การจัดการรหัสผ่าน (Password Management)

รหัสผ่านเป็นจุดอ่อนของแอปพลิเคชัน ซึ่งหากไม่บังคับข้อกำหนดของรหัสผ่านให้มีความยาวและความซับซ้อนมากพอ หรือไม่ปิดใช้งานการป้อนรหัสผ่านหลังจากพยายามเข้าสู่ระบบไม่ถูกต้องหลายครั้ง จะส่งผลให้สามารถคาดเดารหัสผ่านนั้น ๆ ได้ ปัจจุบันแอปพลิเคชันได้นำการยืนยันตัวตนโดยใช้หลายปัจจัย (Multi Factor Authentication: MFA) ซึ่งเป็นกระบวนการเข้าสู่แอปพลิเคชันแบบหลายขั้นตอน [7] โดยจะกำหนดให้ผู้ใช้ป้อนข้อมูลเพิ่มเติมนอกเหนือจากรหัสผ่าน อาทิ แอปพลิเคชันขอให้ผู้ใช้ป้อนรหัสที่ส่งไปยังอีเมล ตอบคำถามลับ หรือสแกนลายนิ้วมือร่วมกับการป้อนรหัสผ่าน ซึ่งการยืนยันตัวตนโดยใช้หลายปัจจัยช่วยป้องกันการเข้าถึงแอปพลิเคชันได้ในกรณีที่รหัสผ่านของผู้ใช้หลุดรอดออกไป วิธีการยืนยันตัวตนโดยใช้หลายปัจจัยจะพิจารณาในประเด็นต่าง ๆ คือ ปัจจัยสิ่งที่รู้ ปัจจัยสิ่งที่ครอบครอง และปัจจัยสิ่งที่เป็น

- ปัจจัยสิ่งที่รู้ ผู้ใช้แอปพลิเคชันต้องพิสูจน์ตัวตนของตนเองด้วยการเปิดเผยข้อมูลที่ไม่มีคนอื่นรู้

ตัวอย่างเช่น คำถามลับพร้อมคำตอบที่มีเพียงผู้ใช้ที่รู้ ชื่อของสัตว์เลี้ยงตัวแรก หรือพินสี่หลักสำหรับการเข้าถึงแอปพลิเคชัน

- ปัจจัยสิ่งที่ครอบครอง ผู้ใช้แอปพลิเคชันต้องระบุโดยใช้บางสิ่งบางอย่างที่มีไม่เหมือนใคร อาทิ โทรศัพท์มือถือ โทเค็นความปลอดภัย การ์ดแสดงผล กุญแจล็อกฮาร์ดแวร์ และกุญแจรักษาความปลอดภัย และอีเมล
- ปัจจัยสิ่งที่จำเป็น ผู้ใช้แอปพลิเคชันต้องพิสูจน์ตัวตนโดยใช้ข้อมูลที่ผู้ใช้มีอยู่ในตัว อาทิ ลายนิ้วมือ ม่านตา เสียง ใบหน้า หรือไบโอเมตริกเชิงพฤติกรรม เช่น ลักษณะของจังหวะการพิมพ์

3.3. การควบคุมการเข้าถึง (Access Control)

การควบคุมการเข้าถึง เป็นกระบวนการที่ช่วยควบคุมสิทธิ์การเข้าใช้แอปพลิเคชัน เพื่อป้องกันไม่ให้บุคคลที่ไม่ได้รับอนุญาตใช้แอปพลิเคชันในการลบ แก้ไข หรือขโมยข้อมูล จนส่งผลให้เกิดความเสียหายต่อองค์กร โดยแนวทางเบื้องต้นสำหรับการควบคุมการเข้าถึง คือ "ปฏิเสธโดยค่าเริ่มต้น" สำหรับข้อมูลที่ละเอียดอ่อน จำกัดสิทธิ์และจำกัดการเข้าถึงข้อมูลที่ปลอดภัยเฉพาะผู้ใช้ที่จำเป็นเท่านั้น นอกจากนี้การปฏิเสธการเข้าถึงสำหรับผู้ใช้ที่ไม่สามารถแสดงการอนุญาตได้ ตรวจสอบให้แน่ใจว่ามีการตรวจสอบคำขอข้อมูลที่ละเอียดอ่อนเพื่อยืนยันว่าผู้ใช้ได้รับอนุญาตให้เข้าถึงข้อมูลดังกล่าว โดยการควบคุมเข้าถึงแบ่งออกเป็น 4 ประเภท [8] ได้แก่

- การควบคุมการเข้าถึงทรัพยากรที่เจ้าของเป็นผู้ควบคุมอย่างสมบูรณ์ (Discretionary Access Control: DAC) คือ ผู้ดูแลแอปพลิเคชันสามารถกำหนดสิทธิ์การเข้าถึงของผู้ใช้อื่น ๆ ได้ อาทิ การอนุญาตให้เข้าไปอ่าน หรือแก้ไขข้อมูล เมื่อมีการร้องขอขณะที่เข้าใช้แอปพลิเคชัน โดยผู้ดูแลแอปพลิเคชันสามารถพิจารณาอนุญาตหรือไม่อนุญาตตามการร้องขอนั้น ๆ นอกจากนี้ยังสามารถโอนสิทธิ์ความเป็นเจ้าของหรือผู้ดูแลแอปพลิเคชันให้กับผู้อื่นได้อีกด้วย ซึ่งเป็นวิธีการควบคุมการเข้าถึงที่มีความยืดหยุ่น และจะไม่เหมาะสมหากนำไปใช้กับการปกป้องข้อมูลที่เป็นความลับ หรือทรัพยากรที่ต้องการความปลอดภัยในระดับสูง
- การควบคุมแบบส่วนกลาง (Mandatory Access Control: MAC) เป็นการกำหนดสิทธิ์การเข้าใช้งานแอปพลิเคชันตามนโยบาย หรือตามระดับชั้นความปลอดภัยที่กำหนดไว้ ซึ่งเป็นการควบคุมโดยระบบไม่ใช่การควบคุมโดยผู้ดูแลระบบ ส่งผลให้ผู้ใช้ไม่สามารถเปลี่ยนแปลงนโยบายต่าง ๆ เหล่านั้นได้ การควบคุมการเข้าถึงประเภทนี้มักใช้กับแอปพลิเคชันที่มีความอ่อนไหวสูง
- การจัดการสิทธิ์ในการเข้าถึงระบบ (Role-based Access Control: RBAC) เป็นการกำหนดบทบาทให้ผู้ใช้สามารถเข้าถึงหน้าจอหรือข้อมูลส่วนใดได้บ้าง อาทิ กำหนดให้ผู้ใช้ ก สามารถเข้าไปลบแก้ไขข้อมูล และติดตั้งแอปพลิเคชันใหม่ได้ ในขณะที่ผู้ใช้ ข ได้รับอนุญาตให้ลบและแก้ไขข้อมูลได้เท่านั้น ไม่สามารถติดตั้งแอปพลิเคชันใหม่ได้ ผู้ใช้อื่น ๆ อาจได้รับสิทธิ์เพียงแค่อ่านข้อมูลเท่านั้น นอกจากนี้ผู้งานหนึ่งคนยังสามารถมีได้หลายบทบาท เช่น ผู้ใช้ ก ได้บทบาทผู้ดูแลระบบ (System Administrator) ของระบบ A แต่เป็นเพียงผู้ใช้ (User) ของระบบ B โดยการควบคุมการเข้าถึงประเภทนี้มีการนำไปใช้อย่างกว้างขวาง และสามารถนำไปใช้ร่วมกับการควบคุมทั้งแบบ DAC และ MAC ได้อีกด้วย
- การควบคุมการเข้าถึงโดยการคัดกรองจากคุณสมบัติของผู้ใช้ (Attribute-based Access Control: ABAC) เป็นการควบคุมการเข้าถึงทรัพยากรโดยคัดกรองจากคุณสมบัติของผู้ใช้บางประการ เช่น หากผู้ใช้เป็นพนักงานขององค์กรและปฏิบัติงานด้าน Security แผนกไอที จะสามารถเข้าถึงอุปกรณ์ Firewall ได้ หรือกำหนดให้ผู้จัดการมีสิทธิ์เข้าไปแก้ไขข้อมูลที่มีความอ่อนไหวได้ เป็นต้น

3.4. การจัดการและการบันทึกข้อผิดพลาด (Error Handling and Logging)

การจัดการและการบันทึกข้อผิดพลาดในกรณีเมื่อเกิดข้อผิดพลาด (Error) ของแอปพลิเคชันเป็นการเขียนโค้ดเพื่อแสดงสาเหตุของข้อผิดพลาดนั้นและทำให้แอปพลิเคชันสามารถทำงานต่อไปได้ ข้อผิดพลาดของแอปพลิเคชันมักบ่งชี้ถึงข้อบกพร่อง ซึ่งหลาย ๆ อย่างทำให้เกิดช่องโหว่ การ

จัดการข้อผิดพลาดและการบันทึกเป็นสองเทคนิคที่มีประโยชน์มากที่สุดในการลดผลกระทบ การจัดการข้อผิดพลาดพยายามตรวจจับข้อผิดพลาดในการเขียนโค้ดก่อนที่จะส่งผลให้เกิดความล้มเหลวร้ายแรง การบันทึกเอกสารข้อผิดพลาดจะช่วยให้นักพัฒนาสามารถวินิจฉัยและบรรเทาสาเหตุได้ ควรมีการนำเอกสารและการบันทึกความล้มเหลว ข้อยกเว้น และข้อผิดพลาดทั้งหมดไปใช้กับระบบที่เชื่อถือได้เพื่อให้เป็นไปตามมาตรฐานการเขียนโค้ดที่ปลอดภัย สำหรับตัวอย่างเทคนิคที่นำไปใช้ในการจัดการและบันทึกข้อผิดพลาด ได้แก่ เทคนิคการใช้ Middleware เพื่อจัดการข้อผิดพลาด เทคนิคการใช้ Try...Catch และเทคนิคการเขียน Custom Error

3.5. การกำหนดค่าระบบ (System Configuration)

การกำหนดค่าระบบ คือ การปรับปรุง (Update) แอปพลิเคชันรวมถึงการแพตช์ (Patch) ของแอปพลิเคชันให้เป็นรุ่น (Version) ปัจจุบันอย่างสม่ำเสมอ เนื่องจากแอปพลิเคชันที่ล้าสมัยเป็นแหล่งรวมช่องโหว่และการละเมิดความปลอดภัย ดังนั้นการกำหนดค่าระบบเป็นแนวทางสำหรับการแก้ไขช่องโหว่และลดความเสี่ยงของการถูกโจมตีได้ โดยการกำหนดค่าระบบเป็นการดำเนินการเพื่อให้ผู้ใช้มั่นใจได้ว่าแอปพลิเคชันที่ใช้งานได้รับการดูแลและบำรุงรักษา 4 ประเด็น ได้แก่

- ความปลอดภัย (Security) ผู้ใช้มั่นใจได้ว่าแอปพลิเคชันที่ใช้งานได้รับการปกป้องจากการละเมิดตามช่องโหว่ต่าง ๆ หลีกเลี่ยงความเสี่ยง ความเสียหาย และการสูญหายของข้อมูล
- นวัตกรรม (Innovation) ปัจจุบันการกำหนดค่าระบบไม่ได้ถูกสร้างมาเพื่อแก้ไขข้อบกพร่องของแอปพลิเคชันเท่านั้น บางแอปพลิเคชันได้มีการนำฟังก์ชันใหม่ ๆ เพิ่มเข้าไป หรือทดแทนฟังก์ชันการทำงานเดิมของแอปพลิเคชัน
- ประสิทธิภาพการทำงาน (Efficiency) สำหรับแอปพลิเคชันที่ล้าสมัย (Outdate) อาจทำให้อุปกรณ์ไม่สามารถทำงานได้อย่างถูกต้องตามวิถีปกติ ส่งผลให้ประสิทธิภาพในการทำงานของผู้ใช้ลดลง หรือทำงานได้ไม่เต็มที่ โดยการกำหนดค่าระบบให้เป็นปัจจุบันสามารถเพิ่มประสิทธิภาพการทำงานของแอปพลิเคชันและอุปกรณ์ที่เกี่ยวข้องได้อีกด้วย
- การตรวจสอบ (Audit) เป็นที่ประจักษ์กันดีในปัจจุบันว่า การทำ IT Audit มักมีข้อกำหนดให้ผู้ใช้และผู้ดูแลแอปพลิเคชันดำเนินการกำหนดค่าระบบให้เป็นปัจจุบัน หากไม่ปฏิบัติตามอาจส่งผลกระทบกับแอปพลิเคชันและเกิดบทลงโทษที่รุนแรงอันเนื่องจากการไม่ปฏิบัติตามข้อกำหนดด้านความมั่นคงปลอดภัยสารสนเทศ (IT Security)

3.6. การสร้างแบบจำลองภัยคุกคาม (Threat Modeling)

การสร้างแบบจำลองภัยคุกคามเป็นการวิเคราะห์และสร้างแบบจำลองของภัยคุกคามเพื่อหาวิธีและเครื่องมือในการป้องกันที่เหมาะสม ซึ่งการสร้างแบบจำลองภัยคุกคามของแอปพลิเคชันจะทำให้สามารถเข้าแอปพลิเคชันมีส่วนใดที่สำคัญ แอปพลิเคชันมีช่องทางในการเข้าใช้งานและการติดต่อสื่อสารกับระบบอะไรบ้าง แอปพลิเคชันมีความเสี่ยงจากการโจมตีที่อาจขึ้นจากช่องทางการติดต่อสื่อสารกับระบบอื่น ๆ อย่างไร และสามารถหาวิธีป้องกันหรือลดช่องโหว่ของภัยคุกคามที่อาจเกิดขึ้นกับแอปพลิเคชันได้ การสร้างแบบจำลองภัยคุกคามสามารถทำได้ 4 ขั้นตอน [9] คือ

- ระบุอุปกรณ์ จำนวน และจัดลำดับความสำคัญของอุปกรณ์ที่ใช้งานแอปพลิเคชัน (Identify Assets)
- ระบุช่องทางการสื่อสารหรือเชื่อมต่อกับระบบอื่น ๆ ของแอปพลิเคชัน (Review Architecture)
- ระบุรูปแบบภัยคุกคามที่มีแนวโน้มจะโจมตีแอปพลิเคชัน (Identify Threats & Vulnerabilities)
- สรุปการโจมตีทั้งหมดที่อาจเกิดขึ้นกับแอปพลิเคชันเพื่อหาแนวทางการป้องกัน และเฝ้าระวังจากเส้นทางที่ถูกตรวจพบ (Document the Threats & Vulnerabilities)

3.7. การเข้ารหัสข้อมูล (Cryptographic Practices)

การเข้ารหัสข้อมูลเป็นแนวทางการปฏิบัติในการปกป้องข้อมูลโดยการแปรรูปข้อมูลอิเล็กทรอนิกส์ธรรมดาให้อยู่ในรูปที่บุคคลทั่วไปไม่สามารถอ่านเข้าใจได้ โดยทั่วไปการเข้ารหัสจะกระทำก่อนการจัดเก็บข้อมูลหรือก่อนการส่งข้อมูล โดยการนำข้อมูลอิเล็กทรอนิกส์กับกุญแจ (Key) ซึ่งเป็นตัวเลขแบบสุ่มใด ๆ ไปผ่านกระบวนการทางคณิตศาสตร์ ผลที่ได้คือข้อมูลที่เข้ารหัส (Encryption) และเมื่อต้องการ

อ่านข้อมูลก็นำข้อมูลนั้นผ่านกระบวนการทางคณิตศาสตร์อีกครั้ง ผลลัพธ์ที่ได้คือข้อมูลดั้งเดิม ซึ่งเรียกว่าการถอดรหัส (Decryption) การเข้ารหัสข้อมูลแบ่งตามวิธีการใช้กุญแจได้ 2 วิธี [10] คือ

- การเข้ารหัสแบบกุญแจสมมาตร (Symmetric-key Cryptography) คือ การเข้ารหัสข้อมูลด้วยกุญแจเดี่ยว ทั้งผู้ส่งและผู้รับจะตกลงกันว่าจะใช้รูปแบบใดในการเข้ารหัสข้อมูล
- การเข้ารหัสแบบกุญแจอสมมาตร (Asymmetric-key Cryptography) คือ การเข้ารหัสและถอดรหัสด้วยกุญแจคู่ โดยกุญแจคู่ประกอบด้วยกุญแจส่วนตัว (Private Key) และกุญแจสาธารณะ (Public Key) โดยใช้หลักการหากใช้กุญแจใดเข้ารหัส ก็ต้องใช้กุญแจอีกลูกในการถอดรหัส

3.8. การตรวจสอบข้อมูลนำเข้าและการเข้ารหัสข้อมูลที่ส่งออกมา (Input Validation and Output Encoding)

การตรวจสอบข้อมูลนำเข้าเป็นกระบวนการเพื่อทำให้มั่นใจว่าข้อมูลที่นำเข้าสู่แอปพลิเคชัน สามารถทำให้แอปพลิเคชันทำงานได้ตรงตามที่คาดหวังหรือกำหนดไว้ โดยส่วนใหญ่การตรวจสอบข้อมูลนำเข้าจะตรวจสอบชนิดของข้อมูล รูปแบบของข้อมูล และขนาดของข้อมูล ซึ่งสิ่งต่าง ๆ เหล่านี้จะส่งผลกระทบต่อการประมวลผลของแอปพลิเคชันทั้งสิ้น นอกจากนี้การตรวจสอบข้อมูลนำเข้ายังสามารถช่วยลดช่องโหว่ที่อาจสร้างความเสียหายต่อแอปพลิเคชัน หรือข้อมูลสำคัญได้ ตัวอย่างเช่น ช่องโหว่แบบ SQL Injection เป็นช่องโหว่ที่สามารถทำให้ผู้ไม่ประสงค์ดีทำการ Bypass Login หรือดึงข้อมูลจากฐานข้อมูลได้โดยตรง ในส่วนของการป้องกันไม่ให้เกิดช่องโหว่ประเภทนี้สามารถใช้ Prepare Statement หรือใช้ Library ที่มีอยู่ในเครื่องมือสำหรับการพัฒนาแอปพลิเคชันสำหรับการตรวจสอบข้อมูลนำเข้าด้วยการได้

การเข้ารหัสข้อมูลที่ส่งออกมาเป็นกระบวนการที่ใช้สำหรับการแปลงข้อมูลให้อยู่ในรูปแบบอื่น ๆ โดยที่ไม่มีการเข้ารหัสข้อมูล เช่น ASCII, UTF-8 เป็นต้น หรืออาจจะมีกระบวนการแปลงข้อมูลเป็นรูปแบบอื่นก็ได้ เช่น Base64 ซึ่งการ Encoding จะไม่เปลี่ยนความหมายหรือคุณสมบัติของข้อมูลเดิม แต่จะทำให้ข้อมูลมีรูปแบบที่แตกต่างไปจากรูปแบบเดิมเท่านั้น และสามารถแปลงกลับเป็นข้อมูลเดิมได้ (Decoding)

4. ข้อดีและข้อเสียของการเขียนโค้ดอย่างมั่นคงปลอดภัย

ข้อดี

- ลดความเสี่ยงจากการถูกโจมตีทางไซเบอร์
- สร้างความเชื่อมั่นให้กับผู้ใช้งานและลูกค้า
- ปฏิบัติตามมาตรฐานสากลที่ยอมรับ

ข้อเสีย

- ใช้เวลามากขึ้นในกระบวนการพัฒนา
- อาจมีค่าใช้จ่ายเพิ่มเติมในการใช้เครื่องมือและเทคโนโลยีที่สนับสนุน
- อาจต้องเปลี่ยนกระบวนการพัฒนาซอฟต์แวร์เดิมหรือเพิ่มความซับซ้อนของการเขียนโค้ด

5. บทสรุป

การเขียนโค้ดให้มีความมั่นคงปลอดภัยเป็นมาตรการควบคุมที่กำหนดไว้ในมาตรฐานสากล ISO/IEC 27001:2022 ซึ่งมุ่งเน้นการเขียนโค้ดสำหรับแอปพลิเคชันให้มีความทนทานต่อการถูกโจมตีจากผู้ไม่ประสงค์ดี โดยการเขียนโค้ดให้มีความมั่นคงปลอดภัยจำเป็นต้องพิจารณาทุกขั้นตอนของกระบวนการพัฒนาระบบ (SDLC) เพื่อให้มั่นใจว่า แอปพลิเคชันตลอดจนข้อมูลที่จัดเก็บจากแอปพลิเคชันได้รับความคุ้มครองให้มีความมั่นคงปลอดภัยและสามารถป้องกันการโจมตีทางไซเบอร์ได้ สำหรับบทความนี้ได้นำเสนอแนวทางสำหรับการเขียนโค้ดให้มีความมั่นคงปลอดภัย 8 แนวทาง แต่ละแนวทางมีวิธีการที่หลากหลาย และนอกจากแนวทางเหล่านี้ ยังมีแนวทางอื่น ๆ และวิธีการอื่น ๆ อีกมากมายที่สามารถนำมาใช้การเขียนโค้ดให้มีความมั่นคงปลอดภัยได้เช่นกัน

เอกสารอ้างอิง

[1] บรรจง หะรังสี. “มาตรฐาน ISO/IEC 27001:2022 ฉบับภาษาไทย,” [ออนไลน์] https: / / www. tnetsecurity. com/ download/ มาตรฐาน-iso-iec-270012022. (เข้าถึงเมื่อ: 18 เมษายน 2568).

[2] UPSKILL UX. “Designing for Security (การออกแบบเพื่อความปลอดภัย) คืออะไร?,” [ออนไลน์]. https://medium.com/upskill-ux/designing-for-

security-การออกแบบเพื่อความปลอดภัย-774e2da2f99c. (เข้าถึงเมื่อ: 15 ธันวาคม 2567).

[3] Equal Assurance (Thailand) Ltd. "ISO/IEC27001:2022 อัพเดทใหม่มีอะไรเปลี่ยนแปลงบ้าง," [ออนไลน์]. https://eqathaitraining.com/archives/2208. (เข้าถึงเมื่อ: 6 มกราคม 2568).

[4] Tori Thurmond. (2023). "8 Best Secure Coding Practices," BLOG [Online]. https://kirkpatrickprice.com/blog/secure-coding-best-practices/. Accessed: Jan. 9, 2025).

[5] Nattaphon Bunsuwan. "10 วิธีเพิ่มความปลอดภัยในการพัฒนาซอฟต์แวร์," [ออนไลน์]. https://sennalabs.com/blog/10-ways-to-enhance-security-in-software-development. (เข้าถึงเมื่อ: 20 ธันวาคม 2567).

[6] Pornsook Kornkitichai. "หลักแห่งการออกแบบระบบอย่างมั่นคงปลอดภัย (Secure Design Principles," [ออนไลน์]. https://medium.com/incognitolab/หลักแห่งการออกแบบระบบอย่างมั่นคงปลอดภัย-secure-design-principles-64a5ba0c6142. (เข้าถึงเมื่อ 21 เมษายน 2568).

[7] AWS Ltd. "การยืนยันตัวตนโดยใช้หลายปัจจัย (MFA) คืออะไร," [ออนไลน์]. https://aws.amazon.com/th/what-is/mfa/. (เข้าถึงเมื่อ: 10 มกราคม 2568).

[8] NT cyfernce. "Access Control จุดเริ่มต้นของการปกป้องข้อมูลให้ปลอดภัย," [ออนไลน์]. https://www.cyfence.com/article/access-control-are-starting-protect-for-data-secure/. (เข้าถึงเมื่อ: 10 มกราคม 2568).

[9] MindPHP Ltd. "Cryptographic (เครบโทกราฟฟิก) คืออะไร," [ออนไลน์]. https://www.mindphp.com/บทความ/244-security/5376-cryptographic-security-antivirus-spyware.html. (เข้าถึงเมื่อ: 10 มกราคม 2568).

[10] Ditto Ltd. "ทำความเข้าใจเรื่อง Data Encryption การเข้ารหัสเพื่อความปลอดภัยของข้อมูลองค์กร," [ออนไลน์]. https://www.dittothailand.com/dittonews/what-is-data-encryption. (เข้าถึงเมื่อ: 10 มกราคม 2568).