การพัฒนาเอเจนท์สำหรับระบบตรวจจับผู้บุกรุก

(Intelligent Agent for Intrusion Detection System)

นางสาว นัทธิกา สกุลเอี่ยมไพบูลย์

นาย บรรดิษฐ์ แก้วสระแสน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

การพัฒนาเอเจนท์สำหรับระบบตรวจจับผู้บุกรุก

(Intelligent Agent for Intrusion Detection System)

โดย

นางสาว นัทธิกา สกุลเอี่ยมไพบูลย์

นาย บรรดิษฐ์ แก้วสระแสน

อาจารย์ที่ปรึกษา

รศ.ดร. เอื้อน ปิ่นเงิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเอเจนท์สำหรับระบบตรวจจับผู้บุกรุก

(Intelligent Agent for Intrusion Detection System)

คณะผู้จัดทำ นางสาว นัทธิกา สกุลเอี่ยมไพบูลย์ รหัส 42010169

นาย บรรดิษฐ์ แก้วสระแสน รหัส 42010176

..... อาจารย์ที่ปรึกษา

(รศ.ดร. เอื้อน ปิ่นเงิน)

การพัฒนาเอเจนท์สำหรับระบบตรวจจับผู้บุกรุก

นางสาว นัทธิกา สกุลเอี่ยมไพบูลย์ 42010169
นาย บรรดิษฐ์ แก้วสระแสน 42010176
รศ.ดร. เอื้อน ปิ่นเงิน อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

ปัจจุบันการแลกเปลี่ยนข้อมูลข่าวสารทางระบบเครือข่ายคอมพิวเตอร์เป็นไปอย่างกว้างขวาง ระบบรักษาความปลอดภัยก็มีความจำเป็นตามมาพร้อมกับเทคโนโลยีที่ก้าวไปข้างหน้าอย่างรวดเร็ว ดังนั้น ระบบตรวจจับผู้บุกรุกก็เป็นทางเลือกหนึ่งสำหรับผู้ใช้คอมพิวเตอร์ที่ต้องการป้องกันข้อมูลจากผู้บุกรุก

โครงงานนี้จัดทำขึ้นเพื่อสร้างระบบที่ทำการตรวจจับผู้บุกรุกบนระบบปฏิบัติการ Linux โดยระบบจะนำปัญญาประดิษฐ์มารวมพัฒนาเพื่อเพิ่มความสามารถในการเรียนรู้ให้แก่ระบบตรวจจับผู้บุกรุก และสามารถประเมินรูปแบบการบุกรุกที่มีความไม่ชัดเจนได้ด้วย โดยวิธีตรวจจับการบุกรุกตั้งอยู่บนสมมติฐานที่ว่าการกระทำใดๆ ที่เป็นการบุกรุกต้องมี การใช้งานระบบผิดไปจากเดิม (Anomaly-Detection) โดยทำการสังเกตลักษณะการใช้งานคำสั่งของผู้ใช้ในช่วงระยะเวลาปัจจุบันกับฐานข้อมูลการใช้งานในอดีต โดยผู้ใช้แต่ละคนจะมีข้อมูลแยกออกจากกัน หากตรวจพบการใช้งานที่ผิดปรกติก็แจ้งเตือนแก่ผู้ดูแลระบบ

Intelligent Agent for Intrusion Detection System

Miss Nattika Sakuliampaiboon 42010169
Mr. Bandit Kaewsasan 42010176
Assoc. Prof. Dr. Ouen Pinngern Advisor

ABSTACT

Nowadays, communication with computer is used spaciously to exchange data and information. Computer technology grows very fast so security system has to develop together. Intrusion detection is an alternative strategy for protecting computer.

The objective of this thesis is to develop an intelligent agent for Intrusion Detection System on Linux operating system. Artificial Intelligent technology is used to learn and assess flexibly of the possibility. The assumption is base on behavior of user that intruder have abnormal behavior. The system will observe process for each user to compare between present and past behavior. If abnormal behavior detected, it will report to system administrator.

กิตติกรรมประกาศ

โครงงานนี้จะไม่สามารถเสร็จสมบูรณ์ได้หากไม่ได้รับคำแนะนำ คำเตือน ควบคุม เอาใจใส่ ช่วยเหลือและดูแลการทำโครงงานนี้อย่างใกล้ชิดอย่างสูงจาก รศ.ดร.เอื้อน ปิ่นเงิน นอกจากนี้ท่านอาจารย์ยังเป็นกำลังใจและแรงผลักดันให้โครงงานนี้เสร็จทันตามกำหนด สามารถเผยแพร่สู่สายตาสาธารณชนได้อย่างภาคภูมิใจ คณะผู้จัดทำขอขอบพระคุณยิ่งสำหรับทุกสิ่งทุกอย่างเป็นอย่างสูงไว้ ณ ที่นี้

นอกจากนี้ต้องขอขอบพระคุณอาจารย์ทุกท่านในสถาบันนี้ที่ได้สอนสั่งคณะผู้จัดทำจนมีความรู้ความสามารถจนถึงทุกวันนี้ รวมทั้งคณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ทำให้คณะผู้จัดทำได้เป็นวิศวกรคอมพิวเตอร์อย่างเต็มภาคภูมิ ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์โดยเฉพาะห้องวิจัย Multimedia ที่ได้เอื้อเฟื้อสถานที่ให้คณะผู้จัดทำได้ทำการวิจัย และช่วยอำนวยความสะดวกต่างๆ ขอขอบคุณ อ.ธนัญชัย ตรีภาค ที่ให้ความช่วยเหลือคณะผู้จัดทำในการทำงานตลอดเวลา ให้คำปรึกษาในยามที่เกิดปัญหาขึ้น ให้ความรู้และสนับสนุนข้อมูลในการทำงานต่างๆ และขอบคุณเพื่อนๆ ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกคนที่คอยเป็นกำลังใจที่ดีมาโดยตลอด

ประการสุดท้ายที่สำคัญที่สุด คือ ขอขอบพระคุณบิดา มารดาที่ได้ให้กำเนิด อบรม สั่งสอน และสนับสนุนการศึกษาและเป็นกำลังใจให้คณะผู้จัดทำอย่างดีมาโดยตลอด นับเป็นพระคุณที่เปรียบมิได้ ทางคณะผู้จัดทำขอขอบพระคุณและจักรำลึกพระคุณของท่านไว้ตลอดไป

นางสาว นัทธิกา สกุลเอี่ยมไพบูลย์

นาย บรรดิษฐ์ แก้วสระแสน

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพประกอบ

สารบัญภาพประกอบ (ต่อ)

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ปัจจุบันมีการใช้งานระบบเครือเน็ตเวิร์คอย่างแพร่หลาย หลายหน่วยงานมีการนำคอมพิวเตอร์มาใช้เพื่อพัฒนาศักยภาพ การทำงานด้วยคอมพิวเตอร์นั้น จะมีการส่งถ่ายข้อมูลถึงกันได้โดยง่ายและรวดเร็ว แต่ปัญหาที่ตามมาคือ ปัญหาของการบุกรุกสร้างความเสียหายให้กลับระบบ หรืออาจเข้ามาขโมยข้อมูลที่มีความสำคัญหรือความลับก็ได้ ทำให้การทำงานของผู้ดูแลระบบมีภาระเพิ่มมากขึ้น อีกทั้งปัญหาบางปัญหาผู้ดูแลระบบไม่สามารถตรวจสอบได้ถี่ถ้วน แนวทางในการแก้ปัญหาคือ การหาผู้ช่วยมาทำหน้าที่ดูแลระบบตลอดเวลา สามารถมองเห็นในสิ่งที่ผู้ดูแลระบบมองไม่เห็น คอยแจ้งเตือนให้กับผู้ดูแลระบบเมื่อมีเหตุการณ์ผิดปกติใดๆ ผู้ช่วยที่สามารถแบ่งเบาภาระของผู้ดูแลระบบได้อย่างมากคือ ระบบตรวจจับผู้บุกรุก

โครงงานวิจัยนี้จัดทำโดยมีการนำความรู้ด้านปัญญาประดิษฐ์มาใช้งาน เพื่อเพิ่มประสิทธิภาพในการเรียนรู้ รวมถึงทำให้ระบบมีความยืดหยุ่นเพิ่มขึ้นด้วย โดยในการทำงานนั้นเราจะนำหลักการของฟัซซี่ลอจิก(Fuzzy logic) และนิวรอนเน็ตเวิร์ค (Neural Network) มาใช้ โดยในขั้นตอนแรกจะนำฟัซซี่ลอจิก มาใช้ในการระบุความถี่ของการใช้งานแต่ละคำสั่ง โดยแบ่งระดับความถี่ของการใช้งานออกเป็น 11 ช่วงตั้ง 0.0 – 1.0 โดยแต่ละช่วงจะต่างกันช่วงละ 0.1 โดยเลขน้อยจะเป็นลักษณะการใช้งานที่น้อย เลขมากความถี่ของการใช้งานก็มากตามไปด้วย

จากนั้นใช้หลักการของนิวรอนเน็ตเวิร์คมาใช้ในการจดจำรูปแบบการใช้งานของแต่ละผู้ใช้งานระบบ โดยจะจดจำความถี่ในการใช้งานแต่ละคำสั่งว่าควรจะเป็นเช่นได้ คือ ถ้ามีการมีใช้งานในคำสั่งนั้นเป็นความถี่ที่เท่ากันหลายๆ ครั้ง ค่าน้ำหนัก ในโหนด (Node) นั้นๆ ก็จะมีค่าเพิ่มขึ้น ในการใช้งานของนิวรอนเน็ตเวิร์ค นั้นจะนำมาใช้ในการเรียนรู้ เอาท์พุตจะเป็นหมายเลขรหัสของผู้ใช้แต่ละคน

เมื่อนำโปรแกรมไปใช้งานจริงนั้นจำเป็นต้องมีช่วงเวลาในการเรียนรู้ และตรวจสอบความถี่ในการใช้งาน หากค่าของความถี่ของการใช้งานแต่ละผู้ใช้งานระบบไม่ตรงกับพฤติกรรมในอดีต จะแสดงถึงการผิดปกติของการใช้งาน

1.2 วัตถุประสงค์และเป้าหมาย

1) เพื่อศึกษาและพัฒนาการนำระบบตรวจจับผู้บุกรุกมาใช้งาน
2) เพื่อพัฒนาแนวคิดในการพัฒนาโปรแกรม แก้ปัญหาของโปรแกรม อย่างมีหลักเกณฑ์
3) เพื่อพัฒนาแนวความคิดในการนำปัญญาประดิษฐ์มาใช้
4) เพื่อสร้าง Agent ที่ช่วยวิเคราะห์และตรวจจับผู้บุกรุก ให้มีประสิทธิภาพและมีความยืดหยุ่นมากขึ้น
5) เพื่อเป็นแนวทางในการพัฒนาระบบรักษาความปลอดภัยในอนาคต

1.3 ขอบเขตของงานวิจัย

1) การพัฒนาจะกระทำบนระบบปฏิบัติการ Linux Red Hat 7.3
2) ระบบที่พัฒนาจะทำการตรวจจับที่เครื่อง Server เพียงเครื่องเดียวเท่านั้น (Host-Base IDS)
3) ระบบนี้ตั้งอยู่สมมติฐานที่ว่า พฤติกรรมของผู้ใช้งานปกติจะไม่ต่างไปจากเดิมมากนัก หากพบพฤติกรรมที่ต่างไปจากเดิมระบบจะถือว่ามีแนวโน้มของการบุกรุกได้ (anomaly detection model)
4) ออกแบบการนำความรู้ด้านปัญญาประดิษฐ์มาใช้งาน คือ Neural และ Fuzzy Logic
5) ไม่สามารถทำงานแบบ Real_Time ได้
6) การป้องกันยังไม่ครอบคลุมถึงการบุกรุกทาง Network

1.4 ขั้นตอนการพัฒนา

1) ศึกษาทฤษฎีและแนวความคิดต่างๆ เกี่ยวกับระบบตรวจจับผู้บุกรุก
2) ศึกษาการทำงานของระบบปฏิบัติการลีนุกซ์
3) ศึกษาการนำปัญญาประดิษฐ์มาใช้รวมกับระบบตรวจจับผู้บุกรุก
4) ออกแบบโครงสร้าง ลักษณะการทำงานรวมถึงฟังก์ชันต่างๆ ของระบบ
5) พัฒนาระบบตรวจจับผู้บุกรุก
6) ทดลองและปรับปรุงระบบตรวจจับผู้บุกรุก

บทที่ 2
ทฤษฎีและหลักการเบื้องต้น

2.1 ระบบตรวจจับผู้บุกรุกคอมพิวเตอร์

2.1.1 ความหมายของระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

ระบบตรวจจับผู้บุกรุกคอมพิวเตอร์ คือ กระบวนการในการตรวจสอบเหตุการณ์ที่เกิดขึ้นในระบบคอมพิวเตอร์หรือระบบเน็ตเวิร์ค โดยรวบรวมข้อมูลจากหลายแหล่ง นำไปทำการวิเคราะห์เพื่อหาเหตุการณ์ที่ผิดปกติ จากนั้นตอบสนองต่อเหตุการณ์นั้นตามผลที่ได้วิเคราะห์มา

กล่าวโดยสรุปแล้วระบบตรวจจับผู้บุกรุกคอมพิวเตอร์ คือ ระบบที่ทำหน้าที่ติดตามดูการทำงานที่เกิดขึ้นบนระบบคอมพิวเตอร์ เพื่อค้นหาร่องรอยที่บอกว่ามีผู้พยายามบุกรุกระบบคอมพิวเตอร์ หรือค้นหาการกระทำที่เกินขอบเขตสิทธิ์ของผู้ใช้ระบบ

การบุกรุก คือ ทุกสถานะหรือเหตุการณ์ที่หลบซ่อนเข้ามาเพื่อทำความเสียหายแก่ระบบ ได้แก่ การเปิดเผยข้อมูล การทำลายข้อมูล การแก้ไขข้อมูล การปฏิเสธการเข้าถึงข้อมูล หรือกระบวนการของระบบ เข้าสู่ระบบที่ไม่อนุญาตให้ใช้

2.1.2 ความจำเป็นที่ต้องมีระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

เมื่อคำนึงถึงเรื่องความปลอดภัยของคอมพิวเตอร์มักเป็นการยากในการมองภาพที่ชัดเจนว่า อะไรที่จะบ่งบอกได้ว่าการใช้งานคอมพิวเตอร์มีความปลอดภัย เนื่องจากความปลอดภัยของคอมพิวเตอร์เป็นสิ่งที่จับต้องไม่ได้และยากต่อการวัด แต่อย่างไรก็ตามเราสามารถเปรียบเทียบความปลอดภัยของคอมพิวเตอร์กับการรักษาความปลอดภัยสถานที่ ในการรักษาความปลอดภัย (รปภ.) สถานที่นั้นนอกจากการ จัดบริเวณที่ต้อง รปภ. ให้มีรั้วรอบขอบชิด มีกุญแจที่ใช้ล็อกประตูหรือทางเข้าออก สิ่งหนึ่งที่จะขาดไม่ได้คือการจัดให้มีบุคคลหรืออุปกรณ์ที่คอยตรวจสอบ การละเมิดต่ออุปกรณ์หรือเครื่องกีดขวางที่จัดตั้งเพื่อความปลอดภัย ทั้งนี้เนื่องจากอาจมีผู้ไม่หวังดีพยายามบุกรุกโดยทำลายอุปกรณ์หรือเครื่องกีดขวางดังกล่าว ดังนั้นเราจึงต้องอาศัยระบบที่ใช้ตรวจสอบเมื่อมีการทำลายหรือล่วงล้ำต่ออุปกรณ์หรือเครื่องกีดขวางที่ได้ติดตั้งไว้อีกชั้นหนึ่ง ตัวอย่างอุปกรณ์ที่ใช้ตรวจสอบเช่น ระบบสัญญาณเตือนขโมยที่ใช้ควบคู่กับรั้วที่แข็งแรง

เป็นที่รู้กันดีว่าระบบคอมพิวเตอร์และเครือข่ายมีการออกแบบที่ไม่ค่อยปลอดภัยมากนักทำให้ผู้บุกรุกมีโอกาสที่จะบุกรุกระบบได้ง่ายแม้ว่ามีไฟร์วอลล์ (Firewall) อยู่แล้วก็ตามแต่ก็ไม่สามารถป้องกันการบุกรุก 100 เปอร์เซ็นต์ เนื่องจากไฟร์วอลล์เป็นการป้องกันระหว่างระบบคอมพิวเตอร์ภายในเครือข่าย (Internal networks) กับระบบภายนอก (Internet) หากผู้บุกรุกเป็นคนภายในไฟร์วอลล์เอง ก็ไม่สามารถป้องกันได้ ดังนั้นการมีไฟร์วอลล์ เปรียบเสมือนการมีรั้วกั้นรอบบ้าน และระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์เปรียบเสมือนการติดตั้งกล้องวีดีโอคอยตรวจสอบสิ่งผิดปกติทำให้การป้องกันระบบทำได้ดียิ่งขึ้น

2.1.3 หน้าที่ของระบบตรวจจับผู้บุกรุก

1) ตรวจจับและวิเคราะห์กิจกรรมของผู้ใช้และระบบ
2) ตรวจสอบรูปแบบของระบบ และ ความอ่อนแอของระบบ
3) กำหนดความมั่งคงของระบบที่สำคัญและไฟล์ข้อมูล
4) จดจำกิจกรรมต่างที่เข้ามาโจมตี
5) วิเคราะห์ที่เป็นมาตรฐานสำหรับรูปแบบพฤติกรรมที่ผิดปกติ
6) ตรวจสอบการทำงานของระบบด้วยการจำกิจกรรมของผู้ใช้ที่สะท้อนถึงนโยบายการบุกรุก

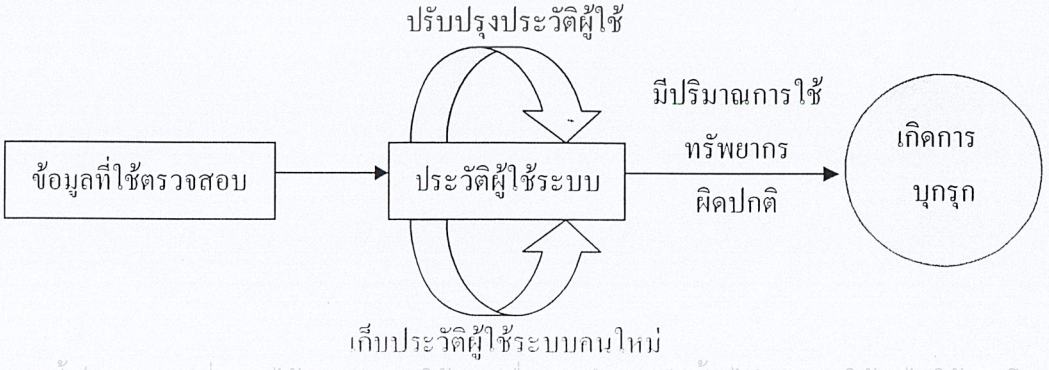
2.1.4 ชนิดของระบบตรวจจับผู้บุกรุก

1) **ระบบตรวจจับผู้บุกรุกในโฮสต์ (Host-base IDS)** ถูกออกแบบมาสำหรับการตรวจสอบโฮสต์ เพียงโฮสต์เดียว จะใช้ข้อมูลการใช้งานระบบเป็นอินพุท สำหรับระบบตรวจจับผู้บุกรุก
2) **ระบบตรวจจับผู้บุกรุกในโฮสต์แบบกระจาย (Distributed Host-Based IDS)** คือการใช้ระบบตรวจจับผู้บุกรุกรวมกันจากหลายๆโฮสต์ จะใช้ข้อมูลการใช้งานระบบจากโฮสต์ หรือข้อมูลจากหลายๆโฮสต์ ซึ่งข้อมูลจะดำเนินการบนเครื่องสูนย์กลาง
3) **ระบบตรวจจับผู้บุกรุกทางเครือข่าย (Network-based IDS)** วิเคราะห์การส่งจ่ายข้อมูลบนแลน เพื่อที่จะทำการตรวจจับพฤติกรรม

2.1.5 ประเภทการตรวจจับ แบ่งได้เป็น 3 แบบ คือ

2.1.5.1 วิธีตรวจสอบการใช้งานระบบที่ผิดปกติ (Anomaly Intrusion Detection Model)

วิธีตรวจสอบการใช้งานทรัพยากรระบบที่ผิดปกติตั้งอยู่บนสมมติฐานที่ว่าการกระทำใดๆ ที่เป็นการบุกรุกจะต้องมีการใช้งานระบบอย่างผิดปกติ โดยจะตรวจจับผู้บุกรุกโดยการตรวจสอบจากประวัติการทำงานที่เคยทำมาแล้วเปรียบเทียบกับการทำงานปัจจุบัน ถ้าการทำงานแตกต่างกันมาก หรือเกิดการใช้งานระบบอย่างผิดปกติ แสดงว่าเกิดการบุกรุก หมายความว่าในการตรวจจับวิธีนี้ต้องมีการเก็บพฤติกรรมปกติของผู้ใช้ระบบไว้เพื่อเปรียบเทียบว่าพฤติกรรมใดที่ผิดปกติ ดังแสดงในรูป



รูปที่ 2-1 การทำงานของการตรวจจับผู้บุกรุกโดยวิธีตรวจสอบการใช้งานระบบที่ผิดปกติ

จากรูป 2-1 ข้อมูลที่ใช้ตรวจสอบ (Audit data) ซึ่งเก็บบันทึกโดยระบบ จะถูกนำมาปรับปรุงไฟล์ประวัติ (Profile) ของผู้ใช้ พร้อมกันนั้นนำข้อมูลนี้มาเปรียบเทียบกับสถิติการใช้งานของผู้ใช้แต่ละคน หากพบว่ามีค่าผิดไปจากค่าปกติก็ระบุว่าอยู่ในสถานะที่มีการบุกรุกเกิดขึ้น และถ้ามีการตรวจสอบโดยใช้ข้อมูลใหม่ๆ ก็สามารถเพิ่มในไฟล์ประวัติได้ ทำให้ระบบสามารถตรวจจับการบุกรุกใหม่ๆ ได้

ข้อดีของการตรวจจับโดยวิธีตรวจสอบการใช้งานระบบที่ผิดปกติ คือ

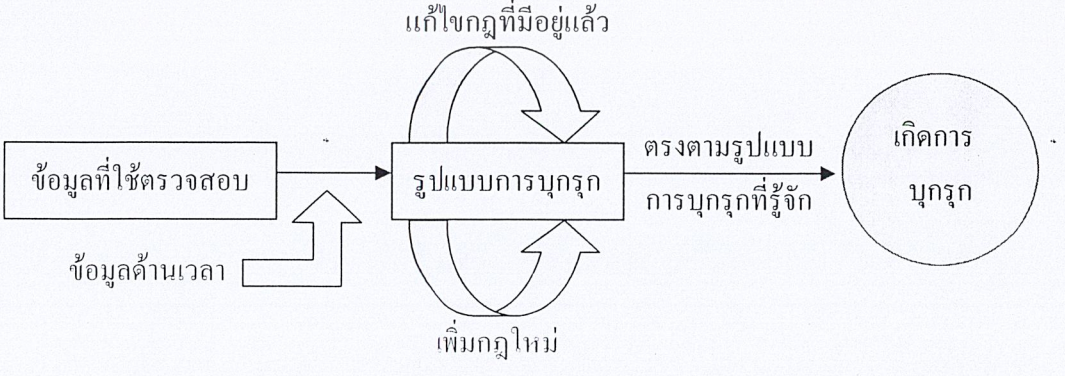
- สามารถตรวจจับบุกรุกแบบที่ไม่เคยเจอมาก่อนได้ โดยที่ไม่ต้องรู้รายละเอียดของการบุกรุกนั้นๆ
- สามารถสร้างข้อมูลเพื่อที่จะระบุพฤติกรรมส่งต่อให้การเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จักได้ (Misuse)

ข้อเสียของการตรวจจับโดยวิธีตรวจสอบการใช้งานระบบที่ผิดปกติ คือ

- อาจมีพฤติกรรมการใช้งานทรัพยากรระบบของผู้ใช้ที่ผิดปกติเกิดขึ้นแต่ไม่ได้เป็นการบุกรุกระบบ ทำให้ระบุสถานะผิดพลาด
- ตรวจไม่พบการบุกรุกระบบเนื่องจากการบุกรุกนั้นไม่ได้ใช้ทรัพยากรระบบอย่างผิดปกติ
- หากผู้บุกรุกค่อยๆ เปลี่ยนพฤติกรรมการใช้งานไปทีละเล็กละน้อยระบบจะไม่สามารถตรวจจับความผิดปกติได้

2.1.5.2 วิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก (Misuse Intrusion Detection Model)

วิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จักประกอบด้วยการเก็บบันทึกและการระบุรูปแบบของการบุกรุกซึ่งอาจบุกรุกจากจุดอ่อนของระบบหรือการละเมิดกฎรักษาความปลอดภัย โดยมีตัวตรวจจับคอยดูแลกิจกรรมต่างๆ ที่กระทำในปัจจุบันว่าเหมือนกับพฤติกรรมการบุกรุกที่เคยเกิดขึ้นหรือได้รับรายงานว่าเป็นการบุกรุกหรือไม่ ซึ่งหัวใจสำคัญของการตรวจจับวิธีนี้คือต้องระบุถึงสัญญาณการเกิดการบุกรุกทั้งหมดที่เป็นไปได้ ในบางระบบมีการใช้กฎ (Rule-based expert system) โดยตั้งกฎขึ้นจากพฤติกรรมที่น่าสงสัย เช่น การ login ล้มเหลวเกินกว่า 3 ครั้งต่อเนื่องกัน ในเวลา 5 นาที ถือว่าพยายามบุกรุก



รูปที่ 2-2 การทำงานของวิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก

จากรูป 2-2 ข้อมูลที่ใช้ตรวจสอบจะถูกนำมาเปรียบเทียบกับกฎ (Rules) ที่มีอยู่สังเกตว่ามีการนำข้อมูลทางเวลาเข้ามาพิจารณาด้วย การตรวจจับการบุกรุกโดยวิธีนี้สามารถมีการแก้ไขกฎหรือเพิ่มกฎได้ในระบบที่เป็นปัญญาประดิษฐ์ (Artificial Intelligence) ระบบอาจทำการแก้ไขกฎหรือเพิ่มกฎได้ด้วยตัวเอง

ข้อดีของวิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก คือ

- การตรวจจับมีประสิทธิภาพที่น่าเชื่อถือ มีความผิดพลาดน้อย
- การตรวจจับทำได้อย่างรวดเร็ว ทำให้สามารถป้องกันระบบได้อย่างทันท่วงที

ข้อเสียของวิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก คือ

- ประสิทธิภาพของระบบการตรวจจับชนิดนี้ขึ้นอยู่กับรูปแบบการบุกรุกที่ระบบรู้จัก
- มีข้อจำกัดในเรื่องจำนวนของรูปแบบในการบุกรุก ซึ่งหากเป็นการบุกรุกที่ระบบไม่รู้จักมาก่อนจะทำให้ไม่สามารถตรวจจับการบุกรุกได้

2.1.5.3 วิธีการแบบผสม (Hybrid detection model)

เป็นการนำ วิธีตรวจสอบการใช้งานระบบที่ผิดปกติ และวิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จักมารวมกัน

2.2 ข้อดีและข้อเสียของระบบตรวจจับผู้บุกรุก

2.2.1 ข้อดีของระบบตรวจจับผู้บุกรุก

1) การตอบสนองทันทีทันใด

จริงๆ แล้วการวิเคราะห์การบุกรุกนั้น หากเป็นผู้เชี่ยวชาญที่มีความรู้ความเข้าใจด้านเน็ตเวิร์คและโพรโตคอลเป็นอย่างดีก็จะวิเคราะห์ได้โดยอาศัยเครื่องมือเพียงเล็กน้อยเท่านั้น คือ ใช้เครื่องมือทำการจัดเก็บบันทึกข้อมูลทั้งหมดที่มีการสื่อสารกันบนเน็ตเวิร์ค แล้วนำข้อมูลที่ได้เหล่านั้นมาวิเคราะห์โดยพฤติกรรมและความสัมพันธ์ ก็จะสามารถหาสิ่งผิดปกติที่เกิดขึ้นได้ แต่การวิเคราะห์ในลักษณะดังกล่าวจะกระทำได้ก็ต่อเมื่อได้เกิดเหตุการณ์ไปแล้ว เนื่องจากการวิเคราะห์จะเป็นไปในลักษณะการวิเคราะห์ข้อมูลย้อนหลัง ไม่สามารถจะกระทำได้ในทันที ซึ่งระบบตรวจจับผู้บุกรุก จะช่วยแก้ไขข้อบกพร่องในส่วนนี้ เพราะ ระบบตรวจจับผู้บุกรุก สามารถตรวจจับได้ทันทีที่มีความผิดปกติเกิดขึ้น และช่วยให้ทำการแก้ไขได้ทันท่วงที การทำงานพื้นฐานของ ระบบตรวจจับผู้บุกรุก จะเหมือนกับการที่ทำโดยคน เพียงแต่ระบบตรวจจับผู้บุกรุกนั้นทำงานโดยอัตโนมัติและการทำงานอยู่ตลอดเวลาไม่มีหยุด จึงสามารถตอบสนองต่อสิ่งผิดปกติได้รวดเร็วกว่า ซึ่งถ้าให้คนมานั่งตรวจจับก็ไม่สามารถทำได้ตลอดเวลา

2) การมีฐานความรู้ของการวิเคราะห์

จากที่ได้กล่าวมาข้างต้น การที่จะตรวจจับสิ่งผิดปกติและแยกแยะกิจกรรมเหล่านั้นออกจากการสื่อสารข้อมูลตามปกติได้นั้นจะต้องอาศัยความชำนาญ และเข้าใจในรูปแบบของการสื่อสารข้อมูลและการบุกรุกเป็นอย่างดี นั่นคือทักษะที่จำเป็นของนักวิเคราะห์การบุกรุก (Intrusion Analyst) ซึ่งผู้เชี่ยวชาญในระดับที่จะทำงานเช่นนี้ได้มีไม่มากนัก ประกอบกับเทคนิคและกลวิธีในการบุกรุกหรือก่อกวนนั้นได้พัฒนาขึ้นทุกวัน วิธีการตรวจจับและวิเคราะห์จำเป็นต้องพัฒนาตามให้สอดคล้องกันจึงจะตรวจจับได้อย่างมีประสิทธิผล ซึ่งในส่วนนี้ผู้เชี่ยวชาญเองก็อาจจะทำได้ไม่ดีเท่า

ระบบตรวจจับผู้บุกรุก สามารถช่วยแบ่งเบาภาระของนักวิเคราะห์ลงได้มาก โดยหากรู้รูปแบบพฤติกรรมแน่ชัดว่าเป็นการมุ่งร้ายก็ให้จัดเก็บข้อมูลรูปแบบเหล่านี้ในระบบตรวจจับผู้บุกรุกเสีย เมื่อมีกิจกรรมดังกล่าวเกิดขึ้นในเน็ตเวิร์ค ระบบตรวจจับผู้บุกรุก ก็สามารถตรวจพบได้ทันที และเมื่อค้นพบรูปแบบใหม่ก็จัดเก็บลงในระบบตรวจจับผู้บุกรุก อีกทั้งทำให้ระบบตรวจจับผู้บุกรุกเสมือนมีฐานความรู้ในการวิเคราะห์การบุกรุกได้ดีในระดับหนึ่ง และขีดความสามารถก็จะเพิ่มขึ้นเรื่อยๆ ตามปริมาณของรูปแบบที่เก็บอยู่ในฐานความรู้นั่นเอง หากมีการบำรุงรักษาฐานความรู้ในตัวระบบตรวจจับผู้บุกรุกได้ดีและนำระบบตรวจจับผู้บุกรุกไปใช้ในจุดที่เหมาะสมแล้ว การบุกรุกที่ไม่ใช่เทคนิคใหม่ล่าสุดจริงๆ ก็แทบจะไม่สามารถเล็ดรอดสายตาระบบตรวจจับผู้บุกรุกไปได้ ถึงขั้นนี้แล้วแม้ว่าจะเป็นระบบตรวจจับผู้บุกรุกแบบธรรมดาๆ ก็มีความสามารถมากกว่าผู้บริหารระบบทั่วไปเสียอีก

สำหรับนักวิเคราะห์แล้วเมื่อมีระบบตรวจจับผู้บุกรุกจะทำให้ไม่ต้องห่วงหน้าพะวงหลัง เพราะการบุกรุกที่สามารถตรวจจับได้ง่ายๆ ก็สามารถตรวจพบได้โดยระบบตรวจจับผู้บุกรุก อย่างน้อยระบบตรวจจับผู้บุกรุกก็ช่วยกลั่นกรองข้อมูลเบื้องต้นได้ในระดับหนึ่งและแบ่งเบาภาระได้พอสมควร

3) การช่วยตรวจสอบข้อบกพร่องของระบบป้องกันอื่น ๆ

เน็ตเวิร์คของผู้ใช้อาจมีการป้องกันการบุกรุกอยู่แล้วโดยใช้ไฟร์วอลล์ (Firewall) อย่างไรก็ตามไฟร์วอลล์มิใช่เครื่องมือที่จะป้องกันการบุกรุกได้โดยอัตโนมัติ จะต้องอาศัยผู้ที่บริหารระบบกำหนดกฎให้เหมาะสมกับการใช้งาน อีกประการหนึ่ง ถึงแม้จะมีการตั้งกฎที่เหมาะสมแล้วก็ตาม แต่กฎเหล่านั้นอาจไม่สามารถป้องกันการบุกรุกได้ การบริหารไฟร์วอลล์ที่ดีก็ควรจะมีการตรวจสอบย้อนหลัง (Audit) และการทดสอบการเจาะระบบ (Penetration Test) เพื่อเป็นการสอบทานระบบอีกครั้งหนึ่ง

ระบบตรวจจับผู้บุกรุก สามารถช่วยได้มาก โดยติดตั้งระบบตรวจจับผู้บุกรุก ไว้หลังไฟร์วอลล์และทำการทดสอบเจาะระบบด้วยวิธีต่างๆ เพื่อดูว่าจะมีเทคนิคใดที่สามารถเจาะผ่านไฟร์วอลล์ได้บ้างและหากมีแพ็กเกจใดผ่านเข้าไปได้ระบบตรวจจับผู้บุกรุก ก็จะตรวจพบทำให้ผู้บริหารระบบสามารถปรับปรุงกฎให้รัดกุมมาก

2.2.2 ข้อเสียของระบบตรวจจับผู้บุกรุก

ถึงแม้ระบบตรวจจับผู้บุกรุก จะมีประโยชน์ค่อนข้างมากในการช่วยรักษาความปลอดภัยและการเตือนภัยล่วงหน้า แต่ก็มีข้อเสียอยู่หลายประการซึ่งผู้ที่จะนำไปใช้จะต้องระหนักไว้

1) การละเมิดความเป็นส่วนบุคคล

เนื่องจากระบบตรวจจับผู้บุกรุก มีพื้นฐานจากการนำข้อมูลทั้งหมดที่สื่อสารกันมาทำการวิเคราะห์ ซึ่งข้อมูลเหล่านั้นจะต้องครอบคลุมถึงข้อมูลทั่วไปที่มีการสื่อสารกันตามปกติ และการที่จะทราบว่ามีความผิดปกติหรือไม่นั้นก็จะต้องอ่านข้อมูลทั้งหมดด้วย ดังนั้นไม่ว่าจะมีกิจกรรมใดๆ ที่เกิดขึ้นในเน็ตเวิร์คไม่ว่าจะเป็นการท่องเว็บ การดาวน์โหลดข้อมูล การแชทคุยกัน ไอซีคิว อี-เมล์ และกิจกรรมอื่นๆ ที่สื่อสารข้อมูลผ่านเน็ตเวิร์ค ก็จะสามารถถูกเปิดอ่านได้จากระบบตรวจจับผู้บุกรุก นั่นหมายความว่าระบบตรวจจับผู้บุกรุก สามารถนำไปใช้ในทางที่ผิดเพื่อละเมิดสิทธิส่วนบุคคลได้ การทำงานของระบบตรวจจับผู้บุกรุก เปรียบเสมือนการที่ตำรวจต้องการตรวจสอบและดักจับผู้ไม่หวังดีที่คอยโทรศัพท์ก่อกวนชาวบ้านในหมู่บ้าน และเพื่อการนี้ตำรวจจึงต้องทำการดักฟังโทรศัพท์ของทุกคนที่อยู่ในหมู่บ้านนั้น ซึ่งอาจจะมีเพียงหนึ่งในพันที่เป็นผู้ร้าย แต่ตำรวจผู้ทำหน้าที่ดักฟังก็จะรู้ความลับของคนทุกคน บางทีการที่มีคนดักฟังความลับของทุกคนอาจเป็นอันตรายกว่าการโดนผู้ร้ายก่อกวนก็เป็นได้

ดังนั้นการนำ ระบบตรวจจับผู้บุกรุก มาติดตั้งในเน็ตเวิร์คจะต้องได้รับการอนุมัติจากหน่วยงานอย่างถูกต้องแล้วเท่านั้น และผู้ทำหน้าที่ในด้านนี้จะต้องเป็นผู้ที่ได้รับความไว้วางใจและมีความรับผิดชอบสูงในอันที่จะไม่ละเมิดสิทธิส่วนบุคคลของคนอื่น และหากเห็นข้อมูลใดๆ ก็จะต้องไม่เปิดเผยข้อมูลเหล่านั้นแก่บุคคลนั้น โดยทั่วไปแล้วการติดตั้งอุปกรณ์ที่สามารถอ่านข้อมูลของผู้อื่นบนเน็ตเวิร์คได้นั้นจะเป็นข้อห้ามอันดับต้นๆ ในนโยบายรักษาความปลอดภัยเลยทีเดียว สิ่งที่เป็นข้อสังเกต คือ การกระทำในลักษณะนี้ยากต่อการป้องกันในทางเทคนิค ดังนั้นหน่วยงานโดยทั่วไปจึงต้องกำหนดเป็นข้อห้ามในนโยบายความปลอดภัย และมีบทลงโทษสำหรับผู้ที่ละเมิดในขั้นรุนแรง

2) การตอบโต้อัตโนมัติ

ระบบตรวจจับผู้บุกรุก ที่มีจำหน่ายอยู่ในท้องตลาดจะมีส่วนหนึ่งที่ให้ผู้ใช้สามารถกำหนดการดำเนินการอย่างหนึ่งอย่างใดเมื่อตรวจพบการบุกรุกเกิดขึ้น เช่น ส่งจดหมายเตือนผู้ดูแลระบบ เรียกวิทยุติดตามตัว ส่งคำสั่งไปยังไฟร์วอลล์เพื่อจำกัดการเข้าออกของข้อมูล และสิ่งที่สำคัญที่สุดซึ่งอาจจะส่งผลเสียหายใหญ่หลวงต่อเจ้าของได้ก็คือ การโจมตีกลับไปยังต้นกำเนิดของการบุกรุก (Counter attack) โดยที่ระบบตรวจจับผู้บุกรุกเองก็จะรู้จักวิธีการโจมตีแบบต่างๆ ดีอยู่แล้ว จึงมิใช่เรื่องยากเย็นแต่อย่างใดที่จะทำการโจมตีผู้อื่น ผู้ผลิตจึงมักเพิ่มเติมส่วนนี้ให้แก่ระบบตรวจจับผู้บุกรุกเสมือนหนึ่งการติดอาวุธๆ ไว้ให้ต่อกรกับแฮ็กเกอร์เลยทีเดียว

ผู้ดูแลระบบบางส่วนอาจรู้สึกสะใจและคิดว่าเหมาะสมแล้วกับการโจมตีกลับไปยังแฮ็กเกอร์เหล่านั้นให้หลาบจำจะได้ไม่พยายามมาข้องแวะอีก เป็นนโยบายการรักษาความปลอดภัยแบบตาต่อตาฟันต่อฟัน และเชื่อว่าหากกำหนดให้การโจมตีกลับเป็นไปอย่างอัตโนมัติแล้วน่าจะทำให้ปลอดภัยมากขึ้น ในค่ำคืนที่เงียบสงบใครจะไปรู้ว่าระบบตรวจจับผู้บุกรุก อาจจะกำลังต่อกรอยู่กับแฮ็กเกอร์ที่พยายามแอบเข้ามาในระบบอย่างสุดกำลัง และสู้ยิบตาเพื่อรักษามิให้แฮ็กเกอร์บุกรุกเข้ามาในเน็ตเวิร์คได้ ในโลกแห่งความเป็นจริงแล้วการตัดสินใจว่าผู้ใดเป็นแฮ็กเกอร์อย่างชัดเจนมิได้ทำได้โดยง่าย และในเวลาอันรวดเร็ว การที่กำหนดให้ระบบตรวจจับผู้บุกรุก ทำการตอบโต้กลับไปในทันทีโดยมีข้อมูลเพียงผิวเผินนั้น นอกจากจะ

ไม่ช่วยให้เน็ตเวิร์คของเราปลอดภัยแล้ว ยังจะทำให้เรากลายไปเป็นแฮ็กเกอร์ ที่คอยโจมตีผู้อื่นเสียเอง ยกตัวอย่างความเสียหาย เช่น

- การวิเคราะห์ผิดพลาดเข้าใจว่ากิจกรรมที่เกิดขึ้นเป็นการบุกรุก และระบบตรวจจับผู้บุกรุก ก็ดำเนินการโจมตีกลับไปในทันที กรณีนี้ผู้บริสุทธิ์ก็จะถูกโจมตีจากระบบตรวจจับผู้บุกรุก ของเราโดยที่ไม่รู้เรื่องใดๆ
- การวิเคราะห์ถูกต้องแต่แอดเดรสของต้นทางเป็นแอดเดรสปลอม กรณีนี้หาก ระบบตรวจจับผู้บุกรุก ไม่มีกลไกในการตรวจสอบแอดเดรสที่มีประสิทธิภาพ อาจไม่สามารถแยกแยะได้ว่าต้นทางของการโจมตีแท้จริงนั้นเป็นที่ไหน และเมื่อทำการโจมตีกลับไปก็อาจจะมิใช่ตัวการที่แท้จริง และเหตุการณ์จะเลวร้ายยิ่งขึ้นหากแอดเดรสที่ปลอมมานั้นเป็นของหน่วยงานทางความมั่นคง หรือหน่วยงานทางทหาร และเมื่อนั้นผู้ดูแลระบบอาจจะตระหนักได้ว่า ระบบตรวจจับผู้บุกรุกตัวเดียวอาจจะทำให้เขาต้องเข้าไปนอนในคุกหลายคืน เทคนิคการปลอมแอดเดรสลักษณะนี้อาจเป็นการยืมมือ ระบบตรวจจับผู้บุกรุกของเราไปโจมตีผู้อื่นอีกทอดหนึ่งได้เป็นอย่างดี
- การวิเคราะห์แอดเดรสที่ถูกต้อง และการโจมตีกลับไปก็ตรงไปยังแฮ็กเกอร์อย่างถูกต้องตามที่ต้องการ แต่ผลที่ได้ก็เพียงอาจจะทำให้แฮ็กเกอร์หยุดความพยายามไปชั่วขณะเท่านั้น อีกไม่นานก็จะหาวิธีกลับมาใหม่ และไม่เกิดผลใดๆ เลยนอกจากจะเป็นการยั่วยุให้มีความรุนแรงมากขึ้นเท่านั้น

สิ่งสำคัญที่ผู้ทำหน้าที่ด้านความปลอดภัยและผู้บริหารระบบควรจะตระหนักไว้ให้จงหนักคือ ท่านไม่มีสิทธิพิเศษที่จะไปตอบโต้ผู้บุกรุกโดยการโจมตีกลับไม่ว่าในกรณีใด สิ่งที่ท่านจะทำได้ดีที่สุด คือ ทำระบบให้แข็งแรงมั่นคงและปลอดภัยที่สุดเท่านั้น นั่นคือปิดประตูบ้านให้แน่น ตรวจตราอย่างรัดกุม และใช้งานเท่าที่จำเป็น ส่วนผู้ที่กระทำผิดเหล่านั้นควรจะปล่อยให้เป็นไปตามกฎหมายและกระบวนการยุติธรรมจะดีที่สุด เพราะการตอบโต้การกระทำที่ผิดกฎหมายด้วยวิธีที่ผิดกฎหมายจะทำให้เรากลายเป็นจำเลยไปด้วยในที่สุด

3) การเตือนภัยที่ผิดพลาด

ข้อนี้อาจไม่ใช่ข้อเสียที่สำคัญของการใช้ระบบตรวจจับผู้บุกรุก หากผู้ใช้มีความรู้ในการใช้งานที่ดีพอและเข้าใจหลักการวิเคราะห์การบุกรุกของระบบตรวจจับผู้บุกรุกได้ดี อย่างที่ได้กล่าวแล้วข้างต้น ก็คือ อาจจะมีกิจกรรมปกติหลายอย่างที่มีลักษณะใกล้เคียงหรือบางครั้งเหมือนกับการพยายามบุกรุก ซึ่งแน่นอนว่าหากระบบตรวจจับผู้บุกรุก ได้ถูกกำหนดให้ตรวจจับกิจกรรมประเภทดังกล่าวแล้วก็จะมีการเตือนในทันทีที่ตรวจพบและเป็นหน้าที่ของนักวิเคราะห์ที่จะทำการสืบค้นข้อมูลด้านอื่นๆ มาประกอบการวินิจฉัยอีกครั้งหนึ่งว่าพฤติกรรมดังกล่าวที่ตรวจพบนั้นเป็นการบุกรุกหรือไม่อย่างไร ระบบตรวจจับผู้บุกรุกที่ถูกกำหนดให้มีความไวเป็นพิเศษมักจะสามารถตรวจจับพฤติกรรมที่ก้ำกึ่งนั้นได้มากเป็นพิเศษ ตัวอย่างเช่น ระบบตรวจจับผู้บุกรุก ได้ถูกกำหนดไว้ว่า เมื่อได้รับ Ping Packet จากแอดเดรสเดิมติดต่อกัน 10 แพ็กเกจภายใน 30 วินาที ให้เตือนว่าเป็นการพยายามโจมตีโดยเทคนิค Ping Flood เป็นต้น

หากเน็ตเวิร์คดังกล่าวเป็นเน็ตเวิร์คที่ใช้งานโดยวิศวกรระบบ และมีการทดสอบการ Ping บ่อยๆ ก็อาจจะทำให้ระบบตรวจจับผู้บุกรุก เตือนอยู่แทบตลอดเวลาโดยไม่ได้มีการบุกรุกที่แท้จริง

การเตือนโดยมิได้มีการบุกรุกจริงนั้น อาจจะดูเหมือนว่าไม่ส่งผลเสียหายประการใดและน่าจะเกิดประโยชน์เสียด้วยซ้ำ เพราะจะทำให้ผู้ดูแลระบบมีความตื่นตัวตลอดเวลา แต่ในความเป็นจริงแล้วธรรมชาติของมนุษย์มีแนวโน้มจะละเลยต่อสิ่งเหล่านี้ หากมีการเตือนแล้วไม่มีการบุกรุกจริงบ่อยครั้งเข้าความน่าเชื่อถือของ ระบบตรวจจับผู้บุกรุก ก็จะลดลงตามลำดับ และเมื่อมีความพยายามบุกรุกจริงก็จะไม่ได้ให้ความสนใจเท่าที่ควรและไม่ได้หาทางป้องกันอย่างเหมาะสม นั่นคือ ระบบตรวจจับผู้บุกรุก จะกลายเป็นเด็กเลี้ยงแกะที่เวลาหมาป่าเข้ามาจริงก็ไม่มีผู้ใดรับฟัง ดูเผินๆ อาจจะเหมือนว่ายังดีกว่าการไม่มีระบบตรวจจับผู้บุกรุก เสียเลยแต่การมีระบบตรวจจับผู้บุกรุกอยู่ในระบบโดยไม่ได้นำมาปรับแต่งอย่างเหมาะสม และเชื่อมั่นว่าระบบตรวจจับผู้บุกรุก สามารถจะคอยระแวดระวังและเก็บหลักฐานต่างๆ ไว้ให้นั้นจะทำให้ผู้บริหารระบบนิ่งนอนใจและคลายความเคร่งครัดในการปฏิบัติงานลง อาจจะถึงขั้นหย่อนยานกว่าการป้องกันในระดับปกติที่ไม่มี ระบบตรวจจับผู้บุกรุกได้ นอกจากนี้การปล่อยให้ ระบบตรวจจับผู้บุกรุก มีการเตือนอย่างไม่เหมาะสมจะทำให้เกิดข้อมูลในลักษณะที่เป็นการบุกรุกจริงและการเตือนผิดพลาดผสมกันอยู่ อาจทำให้การเตือนที่เป็นของจริงถูกกลบไปและยากต่อการสังเกต อย่าลืมว่าแฮ็กเกอร์ที่มีความสามารถจะทิ้งร่องรอยของการบุกรุกไว้เพียงเล็กน้อยอาจจะมีเพียง 2-3 ร่องรอยเท่านั้นที่ ระบบตรวจจับผู้บุกรุก สามารถตรวจพบได้หากร่องรอยเหล่านี้ถูกนำไปผสมปนเปกับการตรวจจับอื่นๆ อีกนับพัน ย่อมมีโอกาสสูงที่จะถูกมองเลยไปโดยไม่มีผู้ใดให้ความสนใจ

2.3 ชนิดของการบุกรุก

2.3.1 การบุกรุกจากภายนอก (Outside Intruders)

การบุกรุกจากภายนอกคือ การบุกรุกจากผู้ที่ไม่ได้รับอนุญาตในการเข้าระบบผู้ที่เข้ามาโจมตีจะทำการขโมย หรือทำลาย Password ทำให้ระบบเกิดข้อผิดพลาด หรือเปลี่ยนแปลงรูปแบบ เพื่อเข้าสู่ระบบ

2.3.2 การบุกรุกจากภายใน (Inside Intruders)

การบุกรุกจากภายใน คือ การบุกรุกจากผู้ใช้ที่ได้รับการอนุญาตให้เข้า แต่ทำการเข้าถึงระบบนอกเหนือจากขอบเขตที่ได้รับ เพื่อที่จะใช้งาน หรือ ทำความเสียหายแก่ระบบ การบุกรุกประเภทนี้ได้สร้างความเสียหายแก่ระบบมากกว่าการบุกรุกจากภายนอก จากรายงานของ FBI ผลออกมาการบุกรุกจากภายในมีปริมาณสูงกว่าการบุกรุกจากภายนอกถึง 80% โดยแบ่งเป็น 3 ประเภท ได้แก่

- ผู้ใช้ตัวปลอมโดยใช้สิทธิในการเข้าระบบของผู้ใช้คนอื่นที่มีสิทธิถูกต้อง
- ผู้ใช้ที่หลบหนีเข้ามาในระบบ คือผู้ที่เข้ามาในระบบได้สำเร็จโดยหนีพ้นจากการตรวจจับ
- ผู้ใช้ที่มีสิทธิพิเศษ คือ ผู้ใช้ที่มีสิทธิพิเศษในการเข้าระบบ

2.4 พฤติกรรมโดยทั่วไปของผู้บุกรุก

โดยประกอบด้วยขั้นตอนสำคัญ 3 ขั้นซึ่งจำเป็นและเป็นพื้นฐานทั่วไปของการบุกรุก

2.4.1 การแกะรอย (Footprinting)

เป็นการรวบรวมข้อมูลของเครื่องเป้าหมายที่ต้องการให้ได้มากที่สุดโดยเฉพาะช่องโหว่ที่มีอยู่บนเน็ตเวิร์ค ทำให้ทราบโพรไฟล์ (Profile) ของเครื่องเป้าหมายที่เชื่อมต่ออยู่กับอินเตอร์เน็ต (Internet) ทั้งในส่วนของเน็ตเวิร์คภายในหรืออินทราเน็ต (Intranet) และเอ็กซ์ทราเน็ต (Extranet) รวมทั้งการให้บริการการเชื่อมต่อจากระยะไกล (remote access) ประกอบด้วย 3 ขั้นตอนย่อย ดังนี้

1) กำหนดขอบเขตของการแกะรอย เป็นการพิจารณาว่าต้องการแกะรอยเน็ตเวิร์คทั้งองค์กรหรือสนใจเฉพาะบางส่วนโดยค้นหาข้อมูลจากแหล่งข้อมูลที่เปิดเผยได้ (Open Source Search) ทำให้ทราบข้อมูลบางอย่างที่น่าสนใจได้ เช่น นโยบายด้านการรักษาความปลอดภัยซึ่งบ่งบอกให้ทราบถึงกลไกการรักษาความปลอดภัยที่ใช้งานอยู่ปัจจุบัน ชื่อผู้ติดต่อและอี-เมล์แอดเดรส หรือตำแหน่งที่ตั้ง
2) การรวบรวมรายละเอียดต่างๆ ของเน็ตเวิร์คเป้าหมาย โดยหาชื่อโดเมนและเน็ตเวิร์คที่เกี่ยวโยงกับเครื่องเป้าหมาย แล้วเรียกใช้โปรแกรม เช่น whois (เป็นโปรแกรมที่ใช้หาข้อมูลว่ามีใครใช้งานอยู่ในระบบบ้าง) ดูโดเมนเนมของระบบ และค้นหาค่ากำหนดของเครื่องที่ทำงานอยู่ในเน็ตเวิร์ค เช่น ชื่อเครื่องรุ่นของระบบปฏิบัติการ ชื่อผู้ใช้ทั้งหมดในระบบ
3) การสำรวจเน็ตเวิร์ค โดยพยายามสำรวจเส้นทางที่แพ็กเกจไอพี เริ่มส่งจากเครื่องต้นทางไปถึงไปทาง

2.4.2 การสแกนเพื่อตรวจสอบ

เปรียบเสมือนการเคาะกำแพงเพื่อสำรวจหาประตูบ้านและหน้าต่าง (ช่องโหว่) โดยการแกะรอยจะทำให้ได้ไอพีแอดเดรส และข้อมูลเกี่ยวกับเน็ตเวิร์คของเครื่องเป้าหมายผ่านการทางสอบถามจากฐานข้อมูล whois ส่วนต่อมาคือ ทำการตรวจสอบว่าเครื่องคอมพิวเตอร์ปลายทางใดบ้างที่เปิดอยู่และสามารถเข้าถึงได้โดยตรงผ่านทางอินเตอร์เน็ตและถ้าเป็นไปได้ควรทราบด้วยว่า มีหมายเลขพอร์ตใดเปิดอยู่บ้าง โดยการใช้เครื่องมือและเทคนิคต่างๆ เช่น ping sweeps , port scans และ automated discovery tools

2.4.3 การค้นหาและรวบรวมรายละเอียด (Enumeration)

เป็นการค้นหาบัญชีผู้ใช้หรือค้นหาทรัพยากรที่แชร์ไว้ ซึ่งข้อแตกต่างสำคัญระหว่างเทคนิคการรวบรวมข้อมูลและเบาะแสกับการค้นหาและรวบรวมรายละเอียดต่างๆ คือ ระดับหรือความร้ายแรงของการบุกรุก หมายความว่า มีการเปิดคอนเน็กชันไปยังเครื่องปลายทางโดยตรงและมีการส่งคำถามถามไปด้วย เมื่อแฮ็กเกอร์ทราบชื่อแอคเคาน์ของผู้ใช้ที่ถูกต้องหรือทราบชื่อแชร์โฟลเดอร์ ผู้บุกรุกจะพยายามทำการคาดเดารหัสผ่านหรือค้นหาจุดบกพร่องของสิทธิที่เซตไว้ที่แชร์โฟลเดอร์นั้นๆ เมื่อผู้บุกรุกสามารถเข้าไปเป็นผู้ใช้ทั่วไปแล้วก็สามารถหาช่องทางขยายสิทธิ์ให้ได้เป็นผู้ดูแลระบบ

ประเภทของข้อมูลที่แฮ็กเกอร์ ต้องการรวบรวม สามารถแบ่งออกเป็น 3 กลุ่มใหญ่ คือ

- รายชื่อทรัพยากรในเน็ตเวิร์ค เช่น ชื่อแชร์โฟลเดอร์ เป็นต้น
- รายชื่อแอคเคานต์ของผู้ใช้และรายชื่อกลุ่ม
- รายชื่อแอพพลิเคชัน

2.5 แนวทางการปฏิบัติเมื่อมีการบุกรุกระบบ

- จัดตั้งกลุ่มที่รับผิดชอบในการจัดการได้ทันทีที่ทราบว่าถูกบุกรุก
- กำหนดแนวทางในการรับมือ เช่น ควรให้ความสำคัญกับการจัดการให้เน็ตเวิร์คทำงานได้ปกติในสภาพเดิมเร็วที่สุด หรือให้ความสำคัญในการหาตัว ผู้บุกรุกผู้ดูแลระบบสามารถถอดสายเน็ตเวิร์คโดยทันทีหรือไม่หากพบว่ามีผู้บุกรุกกำลังทำความเสียหายกับระบบอยู่เมื่อพบผู้บุกรุกแล้วจะคอยดูพฤติกรรมต่อไปหรือจัดการทันที ให้ลองคิดปัญหาที่เกิดขึ้นแล้วหาแนวทางการดำเนินการก่อนที่จะเกิดขึ้นจริง เพราะเมื่อเกิดเหตุการณ์ขึ้นแล้วจะไม่มีเวลาคิด
- ให้กำหนดแนวทางในการแจ้งปัญหา เช่น หากเกิดปัญหาแล้วจะแจ้งผู้บังคับบัญชาในลำดับสูงขึ้นไปก่อน หรือแจ้งหน่วยงานที่เกี่ยวข้องเลย ต้องแจ้งเหตุการณ์การบุกรุกที่เกิดขึ้นกับหน่วยงานที่คอยให้คำปรึกษาและช่วยเหลือใด (ในอเมริกามีหลายหน่วยงาน เช่น FIRST หรือ CERT) ต้องแจ้งตำรวจหรือไม่ จะแจ้งเหตุการณ์นี้กับหุ้นส่วนทางธุรกิจหรือไม่ จะปิดข่าวนี้กับหนังสือพิมพ์หรือไม่
- ให้จัดระบบและขั้นตอนในการดำเนินงานในการเก็บล็อก วิเคราะห์ และเฝ้าติดตามข้อมูลทันทีประเด็นสำคัญในการหาร่องรอยของผู้บุกรุก คือ ต้องมีการเก็บล็อกอย่างเพียงพอสำหรับการวิเคราะห์
- ให้คำแนะนำกับผู้ใช้ทุกคนให้ทราบเกี่ยวกับการป้องกันการบุกรุก

บทที่ 3
ระบบปฏิบัติการลินุกส์และโพรเซส

3.1 ระบบปฏิบัติการลีนุกซ์ (Linux)

3.1.1 ประวัติความเป็นมาของลินุกส์

ลีนุกซ์ถือกำเนิดขึ้นในประเทศฟินแลนด์เมื่อปี ค.ศ. 1980 โดยนายลีนุส โทรวัสด์ส (Linus Trovalds) นักศึกษาภาควิชาวิทยาการคอมพิวเตอร์ ในมหาวิทยาลัยเฮลซิงกิ มีความคิดว่าระบบปฏิบัติการยูนิกซ์ (UNIX) บนพีCในขณะนั้นยังมีความสามารถไม่เพียงพอกับความต้องการของผู้ใช้ ดังนั้นจึงเริ่มสร้างระบบปฏิบัติการยูนิกซ์ของตนเองขึ้น

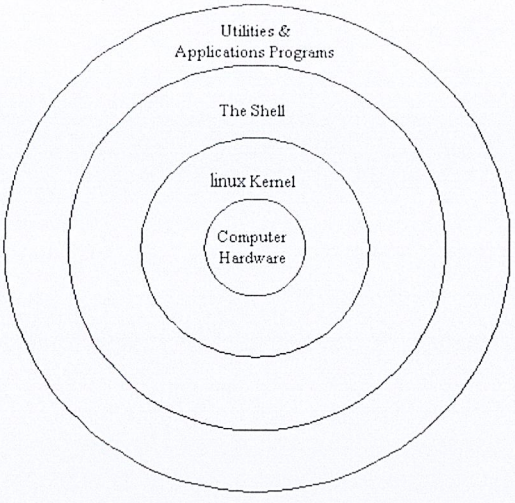
ลักษณะการพัฒนาลีนุกซ์ได้ใช้อินเทอร์เน็ตช่วยโดยอาศัยความร่วมมือของนักพัฒนาจากสถานที่ต่างๆ จึงสังเกตได้ว่าการพัฒนาของระบบปฏิบัติการลีนุกซ์เป็นไปอย่างรวดเร็ว

3.1.2 ลีนุกซ์คืออะไร

ลีนุกซ์เป็นระบบปฏิบัติการแบบ 32 บิตอย่างแท้จริงที่สนับสนุนการใช้งานแบบหลายงาน (Multitasking) และแบบหลายผู้ใช้ (Multi-user) โดยมีระบบการทำงานคล้ายระบบยูนิกซ์ (UNIX Clone) มีระบบเอ็กซ์วินโดว์ (X-windows) ซึ่งเป็นระบบการติดต่อกับผู้ใช้แบบกราฟิกที่ไม่ขึ้นระบบปฏิบัติการ (Operating system) หรือฮาร์ดแวร์ใดๆ และมีโพรโตคอล TCP/IP ซึ่งเริ่มแรกได้ถูกพัฒนาขึ้นมาเพื่อใช้งานกับระบบเน็ตเวิร์กของยูนิกซ์ที่ใช้งานกันอย่างกว้างขวางในการเชื่อมกันของเครือข่ายอินเตอร์เน็ต ระบบเน็ตเวิร์กบนลีนุกซ์ถือได้ว่ามีความสมบูรณ์ต่อการใช้งานจริง เช่น สามารถที่จะเชื่อมต่อเป็นเครือข่ายผ่านระบบแลนทั่วไป หรือการเชื่อมต่อในระบบรีโมตเทอร์มินัลผ่านโมเด็ม

ลีนุกซ์มีความเข้ากันได้ (Compatible) กับมาตรฐานโพสซิก (POSIX) ซึ่งเป็นมาตรฐานอินเทอร์เฟซที่ระบบยูนิกซ์ส่วนใหญ่ต้องมี และมีรูปแบบบางส่วนที่คล้ายกับระบบปฏิบัติการยูนิกซ์จากค่ายเบิร์กเลย์ (Berkeley) และซิสเต็มวี (System V)

โดยเทคนิคแล้วลีนุกซ์เป็นเพียงเคอร์เนล (Kernel) ของระบบปฏิบัติการ มีหน้าที่สำคัญ คือ การจัดการกับโพรเซสและการจัดการกับไฟล์ รวมทั้งการจัดการกับอุปกรณ์ต่างๆ เป็นส่วนที่จะทำหน้าที่ในการบริหารรวมทั้งควบคุมระบบทั้งหมด มีโครงสร้างการทำงาน ดังรูป



รูปที่ 3-1 โมเดลของระบบปฏิบัติการลินุกซ์

นอกจากนี้แล้วลีนุกซ์ยังมีลักษณะพิเศษในการจัดการกับหน่วยความจำและไฟล์ คือไฟล์เอ็กซิคิวต์เทเบิล (Executable file) สามารถแชร์พื้นที่ในหน่วยความจำได้ ลักษณะในการทำงานแบบนี้เรียกว่า copy on write page ซึ่งเป็นการเพิ่มประสิทธิภาพในการใช้งานหน่วยความจำ สามารถทำงานในระบบดีมานเพจจิง (Demand paging) ซึ่งเป็นการนำเอาโปรแกรมที่ใช้งานบ่อยมาเก็บไว้ในหน่วยความจำ และถ้ามีการใช้งานระบบมากขึ้น ความจุดิสก์แคชจะถูกปรับอัตโนมัติ เมื่อต้องการที่จะใช้หน่วยความจำเพิ่มขึ้น ลีนุกซ์สามารถที่จะนำพื้นที่บนดิสก์มาทำหน้าที่เสมือนหน่วยความจำได้ (swap space) แต่ก็มีข้อเสียอันเนื่องมาจากการแอ็กเซสข้อมูลที่ช้าของดิสก์เมื่อเทียบกับหน่วยความจำจริง

นอกจากนี้แล้ว ถ้ามีแอพพลิเคชันใดใช้งานไลบรารีเหมือนกัน ลีนุกซ์สามารถที่จะจัดการเพื่อให้แอพพลิเคชันเหล่านั้นใช้งานไลบรารีร่วมกัน เป็นการลดขนาดแอพพลิเคชันลง (Dynamic shared libraries)

นอกจากนี้ในปัจจุบัน ลีนุกซ์ยังสามารถใช้งานบนแพลตฟอร์ม (Platform) ต่างๆ ได้ ตัวอย่างเช่น DEC Alpha, Motorola Power-PC เป็นต้น

3.1.3 สาเหตุที่เลือกใช้ลีนุกซ์ในโครงงาน

สาเหตุที่เลือกให้ลีนุกซ์ในโครงงานนี้ คือ ลีนุกซ์เป็นระบบปฏิบัติการที่แจกฟรี สามารถดาวน์โหลดจากอินเทอร์เน็ตโดยไม่ผิดกฎหมายและมีผู้นิยมใช้มาก มีผู้นำลีนุกซ์ไปแก้ไขให้สามารถใช้งานได้บนตัวประมวลผลหลากหลาย นอกจากนี้ยังมีผู้พัฒนาแอพพลิเคชันสำหรับลีนุกซ์ออกมาเรื่อยๆ

เนื่องจากลีนุกซ์เป็นระบบปฏิบัติการ 32 บิตแบบแท้จริง ทำให้ดึงความสามารถดึงการทำงานของคอมพิวเตอร์ออกมาได้เต็มกำลัง การทำงานของลีนุกซ์จึงมีประสิทธิภาพและลีนุกซ์มีคุณลักษณะของยูนิกซ์เต็มรูปแบบ ทำให้เป็นระบบหลายผู้ใช้และหลายงาน ซึ่งสนับสนุนโพรโตคอลแบบ TCP/IP, SLIP, PPP, UUCP และอื่นๆ โดยเวอร์ชันที่เลือกนำมาใช้เป็น Linux mandrake 8.0 และใช้เคอร์เนล 2.4.3

สำหรับการใช้ประโยชน์ของลีนุกซ์ในด้านอื่นๆ สามารถใช้ลีนุกซ์ในการศึกษาระบบปฏิบัติการยูนิกซ์ และแพ็กเกจของลีนุกซ์ได้รวบรวมโปรแกรมที่ใช้จำลองสภาพการทำงานของดอส เพื่อให้โปรแกรมที่ทำงานบนดอส สามารถทำงานบนลีนุกซ์ สำหรับวินโดวส์มีซอฟต์แวร์ที่สามารถทำให้ลีนุกซ์รันแอพพลิเคชันของวินโดวส์บนระบบ X ชื่อ WABI (Windows Applications Binary Interface)

3.1.4 ระบบไดเรกทอรีบนลินุกซ์ (Linux Directory System)

ไดเรกทอรีเป็นกลุ่มของไฟล์ ข้อมูลภายในไดเรกทอรีจะประกอบด้วยไฟล์ หรือซับไดเรกทอรี ถ้ามีไดเรกทอรี A แล้วภายในไดเรกทอรี A มีไดเรกทอรีอื่นอยู่ด้วยชื่อไดเรกทอรี B จะเรียกไดเรกทอรี A ว่าเป็นแพเรนท์ไดเรกทอรี (Parent Directory) ส่วนไดเรกทอรี B จะเรียกว่าเป็นซับไดเรกทอรี

จำนวนไฟล์หรือซับไดเรกทอรีสำหรับลีนุกซ์นั้นมีได้ไม่จำกัดขึ้นอยู่กับฮาร์ดดิสก์ว่ามีพื้นที่ในการเก็บข้อมูลมากเท่าใด แต่โดยทั่วไปแล้วโครงสร้างของไฟล์จะประกอบไปด้วย root (Root Directory) ซึ่งเป็นแพเรนท์ไดเรกทอรีทั้งหมดบนระบบ ซึ่งจะแสดงด้วยตัวอักษร “/” หมายถึงว่า ตอนนี้อยู่ที่ไดเรกทอรีสูงสุดแล้ว ภายใต้แพเรนท์ไดเรกทอรีจะประกอบไปด้วยไดเรกทอรีที่สำคัญอยู่จำนวนหนึ่งเป็นไดเรกทอรีหลักของระบบ

- "/bin" เก็บโปรแกรมที่สำคัญของระบบเช่น “ls” (ใช้แสดงรายชื่อไฟล์), “cp” (ใช้คัดลอกไฟล์ข้อมูล) โดยเก็บเป็นไบนารีไฟล์ (binary file) ซึ่งเป็นภาษาเครื่อง
- "/boot" เก็บไฟล์ที่เก็บส่วนสำคัญสำหรับการบูตระบบ
- "/dev" เป็นไฟล์ที่ใช้แทนอุปกรณ์ต่าง ๆ ของระบบ เช่น “/dev/hda” จะแทนฮาร์ดดิสก์, “/dev/tty” จะแทนเทอร์มินอล (terminal)
- "/etc" ค่าที่ใช้เซตการทำงานของโปรแกรมต่าง ๆ จะถูกเก็บลงในไดเรกทอรีนี้ เช่น lilo.conf , syslog.conf เป็นต้น
- "/home" เก็บข้อมูลต่าง ๆ ของ ผู้ใช้งานในระบบ เช่น ถ้าผู้ใช้ชื่อ “mint” ค่าปกติของไดเรกทอรีที่เก็บไฟล์จะเป็น “/home/mint” เป็นต้น
- "/lib" เก็บไฟล์ไลบรารี ของระบบ
- "/proc" เก็บข้อมูลที่เกี่ยวกับโพรเซสและเคอร์เนล
- "/root" เก็บข้อมูลของผู้ดูแลระบบ
- "/sbin" เก็บโปรแกรมที่ผู้ดูแลระบบใช้ในการดูแลรักษาระบบ เช่น “shutdown”, “fsck” เป็นต้น
- "/usr" เก็บโปรแกรมที่ติดตั้งโดยผู้ใช้งาน และนอกจากนั้นหากได้ติดตั้งส่วนของซอร์สโค้ด (source code) ของเคอร์เนลไว้ ก็จะสามารถดูได้ที่ ”/usr/src”
- "/var" เก็บไฟล์ที่ใช้ดูแลระบบเช่น ล็อกไฟล์ (log file)ของระบบ และข้อมูลที่มีการเปลี่ยนแปลงอยู่บ่อยครั้ง เช่น อี-เมล์ เป็นต้น

3.1.5 การพัฒนาระบบงานบนลีนุกซ์

ลีนุกซ์ได้ทำการเตรียม เครื่องมือพัฒนาโปรแกรมให้เราไว้อย่างครบครันซึ่งจะมีตั้งแต่แอพพลิเคชันมาตรฐานคือ C/C++ คอมไพเลอร์ของ GNU และหากเราต้องการพัฒนาระบบบนเอ็กซ์ (X) ก็มี TCL/TK เตรียมไว้ให้ด้วย

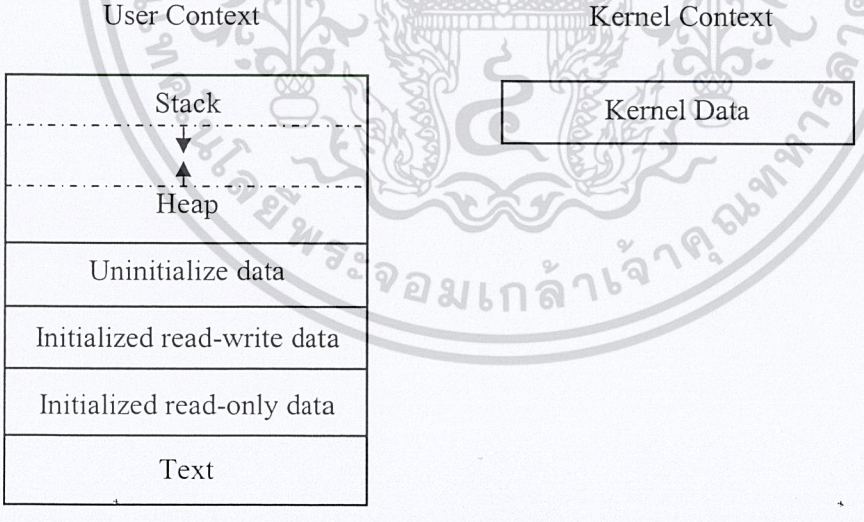
สำหรับคอมไพเลอร์ภาษาอื่นๆก็มีเช่น เพิร์ล(Perl), สมอลล์ทอล์ก(Smalltalk), พาสคาล (Pascal), ลิสป์(Lisp) เป็นต้น ถ้าคุณมีความเชี่ยวชาญการเขียนโปรแกรมแบบเอ็กซ์เบส (X-Base) หรือฟ็อกซ์โปร(FoxPro) บนลีนุกซ์ก็มีดาต้าเบสที่มีการเขียนโปรแกรมแบบนี้ให้เช่นกัน และล่าสุดลีนุกซ์ก็มีจาวาคอมไพเลอร์ให้สำหรับผู้ที่ชื่นชอบการเขียนแอพเพลตจาวา สำหรับรันบนอินเทอร์เน็ตด้วย

3.2 โพรเซสและข้อมูลเกี่ยวกับโพรเซส

3.2.1 ความหมายของโพรเซส

โพรเซส หมายถึง ชุดคำสั่งในหน่วยความจำที่กำลังทำงานอยู่ในขณะนั้น โดยมีชุดควบคุมการทำงาน 1 ชุดเป็นของตัวเองซึ่งได้ใช้ทรัพยากรของระบบไปส่วนหนึ่ง ทั้งนี้ต้องทำความเข้าใจไว้ก่อนว่าโพรเซสกับโปรแกรมไม่ใช่สิ่งเดียวกันยกตัวอย่างเช่น ในระบบมัลติยูสเซอร์สมมติว่ามีผู้ใช้งาน 50 คน กำลังใช้โปรแกรม pine เพื่อทำการอ่านอีเมล์พร้อมๆ กันก็จะมีโพรเซสเกิดขึ้น 50 โพรเซสที่กำลังใช้โปรแกรม pine เพียงโปรแกรมเดียว

3.2.2 โครงสร้างของโพรเซส



รูปที่ 3-2 แสดงโครงสร้างอย่างคร่าว ๆ ของโพรเซส

จากรูปเราแบ่งโครงสร้างโพรเซส 2 ส่วน คือ

ส่วน User context

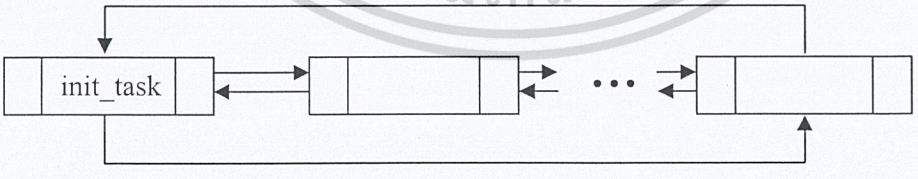
ประกอบไปด้วยส่วนต่าง ๆ ดังนี้

- text เก็บภาษาเครื่อง (machine instruction) ที่ใช้รัน โดยปกติถูกกำหนดให้มีเพอร์มิสชัน (permission) เป็น read only ดังนั้นโพรเซสจึงไม่สามารถแก้ไขข้อมูลส่วนนี้ได้
- data เก็บข้อมูลที่ใช้ในโปรแกรม แบ่งเป็น 3 ส่วนย่อยคือ uninitialized data, initialized read-write data และ initialized read only data
- heap ใช้งานในขณะรันโพรเซส เพื่อกำหนดพื้นที่หน่วยความจำให้เพียงพอต่อความต้องการในการใช้งาน
- stack สำหรับควบคุมการใช้งานตัวแปร การเรียกใช้ฟังก์ชันและการส่งค่ากลับ

ส่วน Kernel context

ถูกควบคุมและใช้งานโดยเคอร์เนลเท่านั้น เพื่อเก็บข้อมูลที่จำเป็นสำหรับควบคุมและติดตามการทำงานของโพรเซสซึ่งการที่เคอร์เนลจะจัดการกับโพรเซสได้นั้น ตัวเคอร์เนลเองต้องทราบข้อมูลต่างๆ ที่เกี่ยวกับโพรเซสแต่ละตัวเก็บไว้เช่น มีหมายเลขประจำโพรเซส (Process ID) มีค่าเท่าไร สถานะการทำงานของโพรเซสกำลังประมวลผลอยู่หรือถูกบล็อก (block) อยู่หรือไม่ มีแอดเดรสสเปซ (address space) อยู่ที่ตำแหน่งใด มีไพรออริตี้(priority)เท่าไร เป็นต้น ซึ่งข้อมูลต่าง ๆ เหล่านี้ แต่ละโพรเซสมีเก็บไว้ในแต่ละโพรเซสในส่วนที่เรียกว่า process descriptor โดยรายละเอียดโครงสร้างที่เป็นส่วนประกาศในภาษาC สามารถดูได้ที่ /usr/include/linux/sched.h ซึ่งอาจจะแตกต่างกันไปตามเวอร์ชันของเคอร์เนล

ในโครงสร้างข้อมูลนี้ มีโครงสร้างข้อมูลที่ใช้ชี้ไปยัง Process descriptor ถัดไปและก่อนหน้านี้ ซึ่งหากลองพิจารณาดูจะเห็นว่ามีโครงสร้างเป็น circular doubly link list ดังรูปที่3-3 ซึ่งส่วนหัวของลิงค์ลิสนี้เป็น init_task descriptor ซึ่งเป็นโพรเซสแรกของลีนุกซ์ที่เกิดขึ้นในช่วงการบูทระบบซึ่งมักถูกเรียกว่า process 0 หรือ swapper จะมีหน้าที่ในการบริหารงานโพรเซส



รูปที่ 3-3 ลักษณะของการเชื่อมโยงกันของ process descriptor

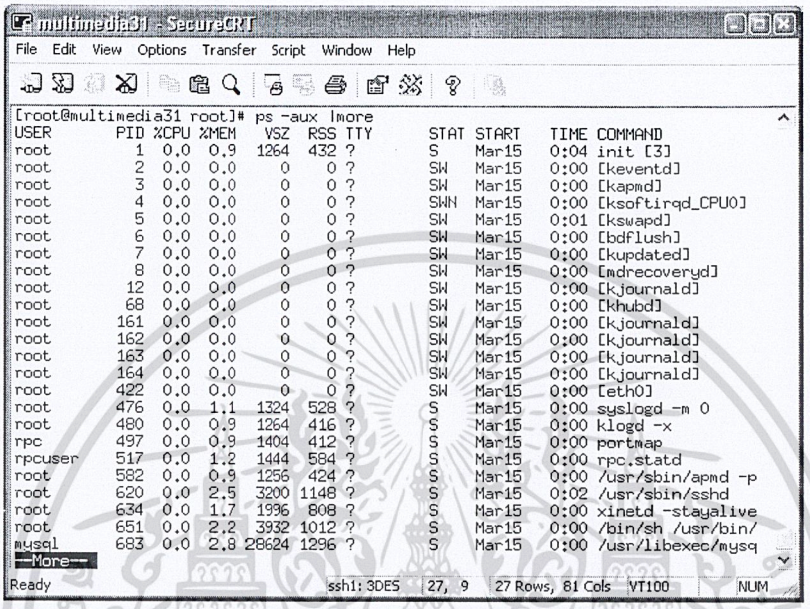
ในการจัดการกับโพรเซสทั้งหมดนั้นเคอร์เนลมีอาร์เรย์ ที่เรียกว่า Task array โดยสมาชิกในarray นี้ประกอบด้วย pointer ชี้ไปยัง process descriptor ของโพรเซสต่างๆ มีไว้เพื่อความสะดวกของการจัดการโพรเซสของเคอร์เนลเอง

3.2.3 การตรวจสอบข้อมูลของโพรเซส

มีคำสั่งอยู่หลายคำสั่งในลีนุกซ์ที่ใช้ในการตรวจสอบข้อมูลของโพรเซส เช่น

- **คำสั่ง ps**

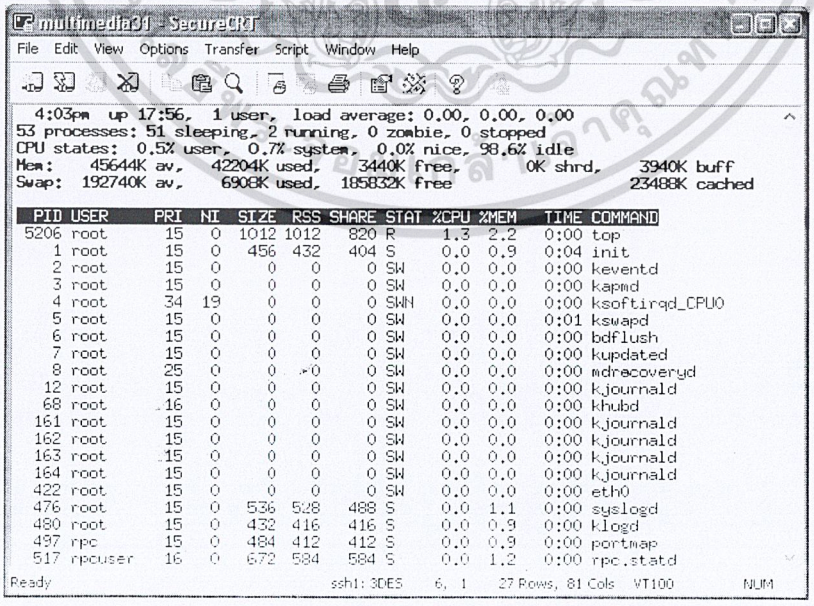
  แสดงข้อมูลของโพรเซสที่อยู่ในไดเรกทอรี /proc (ซึ่งจะขออธิบายส่วนนี้ในหัวข้อถัดไป)



รูปที่ 3-4 แสดงการผลลัพธ์จากการรันคำสั่ง ps

- **คำสั่ง top**

  แสดงข้อมูลของโพรเซสเช่นกัน แต่มีการสแกนข้อมูลอยู่ตลอดเวลา



รูปที่ 3-5 แสดงผลลัพธ์ของการรันคำสั่ง top

3.2.4 การจัดเก็บข้อมูลใน /proc

ภายในไดเรกทอรี /proc เป็นที่เก็บข้อมูลรายละเอียดเกี่ยวกับเคอร์เนลและโพรเซสซึ่งในที่นี้ขอกล่าวถึงเฉพาะในส่วนที่เกี่ยวกับโพรเซส

/proc/[number]

เก็บข้อมูลเกี่ยวกับโพรเซสตามหมายเลขที่กำกับไว้ เช่น /proc/1243 ภายในไดเรกเทอรีนี้เก็บข้อมูลเกี่ยวกับโพรเซสที่มีหมายเลขโพรเซสเป็น 1243 ไว้ ซึ่งภายในมีรายละเอียดดังนี้

1) ไฟล์ **cmdline** : เก็บคำสั่งที่โพรเซสนั้นประมวลผล
2) ไฟล์ **environ** : เก็บค่าตัวแปรสภาพแวดล้อมของโพรเซสนั้น ๆ
3) ไดเรกทอรี **fd** : เก็บ Symbolic link ของ file descriptor ของไฟล์ที่โพรเซสนั้นได้เรียกใช้งาน
4) **cwd Symbolic link** : เป็นลิงค์ที่ชี้ไปยังไดเรกทอรีที่โพรเซสทำงานอยู่ (Current working directory)
5) **exe Symbolic link** : เป็นลิงค์ที่ชี้ไปยังไฟล์ที่โพรเซสกำลังประมวลผลอยู่
6) ไฟล์ **maps** : เก็บส่วน Memory map ของโพรเซส ซึ่งประกอบด้วยช่วงตำแหน่ง, permission, offset เป็นต้น
7) **root Symbolic link** : เป็นลิงค์ไปยังรูตไดเรกทอรี (root directory)
8) ไฟล์ **statm** : เก็บข้อมูลการใช้งานหน่วยความจำของโพรเซส
9) ไฟล์ **stat** : เก็บข้อมูลรายละเอียดเกี่ยวกับโพรเซส ซึ่งไฟล์นี้เป็นไฟล์ที่มีรายละเอียดมากที่สุดโดยมีการเก็บข้อมูลเรียง ๆ กันซึ่งลำดับการเรียงข้อมูลนั้นอาจแตกต่างกันไปตามเวอร์ชันของเคอร์เนล ซึ่งความหมายของแต่ละตัวแปรดูได้ตามตารางที่ 2-1

ตัวแปร	ความหมาย
pid	หมายเลขประจำโพรเซส
cmd	คำสั่งที่โพรเซสทำการประมวลผล
state	สถานะของโพรเซส
ppid	หมายเลขประจำโพรเซสของโพรเซสที่เป็นผู้สร้างโพรเซสนี้
pgrp	หมายเลขกลุ่มของโพรเซส
session	หมายเลข session ของโพรเซส
tty	tty ที่โพรเซสใช้
tpgid	หมายเลขประจำโพรเซสของ tty ที่ใช้
flags	flag ของโพรเซส
min_flt	ค่า minor page fault
cmin_flt	ค่า minor page falut ที่โดยรวมค่าที่โพรเซสลูกของโพรเซสนี้ด้วย
maj_flt	ค่า major page fault
cmaj_flt	ค่า major page falut ที่โดยรวมค่าที่โพรเซสลูกของโพรเซสนี้ด้วย

utime	User time ในหน่วย jiffies (1/100 วินาที)
stime	System time ในหน่วย jiffies
cutime	User time โดยรวมเวลาของโพรเซสลูกด้วย ในหน่วย jiffies
cstime	System time โดยรวมเวลาของโพรเซสลูกด้วย ในหน่วย jiffies
priority	ค่า static priority ของโพรเซส
nice	เป็นค่าไพรออริตี้ที่สามารถเปลี่ยนแปลงได้ (Dynamic priority)
start_time	เวลาที่โพรเซสเริ่มทำงาน
vsize	ขนาดของ Visual Memory ในหน่วย byte
rss	Resident set size
rlim	ค่าจำกัดของ RSS
start_code	จุดเริ่มต้นของโค้ดเซ็กเมนต์ (code segment)
end_cod	จุดสิ้นสุดต้นของโค้ดเซ็กเมนต์
start_stack	จุดเริ่มต้นของสแต็กเซ็กเมนต์ (stack segment)
esp	ตำแหน่งของสแต็กเฟรมปัจจุบัน (current stack frame)
eip	ตำแหน่งของสแต็กเฟรมปัจจุบัน (current stack frame)
pending.signal	เก็บสัญญาณที่โพรเซสต้องการตรวจสอบว่ายังค้างอยู่ในระบบหรือไม่
blocked.signal	เก็บสัญญาณที่โพรเซสต้องการบล็อก
Sigign	เก็บสัญญาณที่โพรเซสไม่สนใจ
Sigcatch	เก็บสัญญาณที่โพรเซสกำลังรอ

ตารางที่ 3-1 แสดงความหมายของข้อมูลในไฟล์ /proc/[number]/stat

10) ไฟล์ **status** : เก็บข้อมูลของโพรเซสเช่นเดียวกันโดยแสดงให้อยู่ในรูปที่เข้าใจได้ง่าย แต่ข้อมูลน้อยกว่าในไฟล์ stat

3.2.5 ล็อกไฟล์ pacct

ไฟล์นี้มีหน้าที่เก็บข้อมูลของโพรเซสที่ผู้ใช้ได้เรียกใช้งานซึ่งถูกบันทึกเมื่อโพรเซสจบการทำงานปกติแล้วจะเก็บไฟล์นี้ไว้ที่ /var/account โดยจะเก็บข้อมูลเป็นรายวันแล้วมีการตัดข้อมูลที่ผ่านมาแบ็กอัพไว้ซึ่งทั้งหมดนี้ทำโดยใช้ logrotate ลักษณะโครงสร้างจะสามารถเข้าไปดูได้ที่ /usr/include/linux/acct.h ซึ่งมีลักษณะดังนี้

ตัวแปร	คำอธิบาย
ac_flag	Flags ของโพรเซส
ac_uid	หมายเลขประจำตัวของผู้ที่เป็นเจ้าของโพรเซส
ac_gid	หมายเลขประจำกลุ่มของผู้ที่เป็นเจ้าของโพรเซส

ac_tty	เทอร์มินอลที่ใช้ในการรันโพรเซส
ac_btime	เวลาที่โพรเซสเกิดขึ้น
ac_utime	User time
ac_stime	System Time
ac_etime	Elapsed Time
ac_mem	ปริมาณการใช้หน่วยความจำโดยเฉลี่ย
ac_io	ปริมาณการใช้ i/o
ac_rw	จำนวนบล็อกที่เขียนหรืออ่าน
ac_minflt	ค่า Minor Pagefaults
ac_majflt	ค่า Major Pagefaults
ac_swaps	จำนวน swap ที่ใช้
ac_exitcode	Exit code ของโพรเซส
ac_comm[ACCT_COMM + 1]	คำสั่งที่ใช้ในโพรเซสนี้

ตารางที่ 3-2 แสดงความหมายของตัวแปรในโครงสร้าง struct acct

คำสั่งที่ใช้ตรวจสอบดูข้อมูลในไฟล์นี้ก็มีเช่น lastcomm, sa เป็นต้น

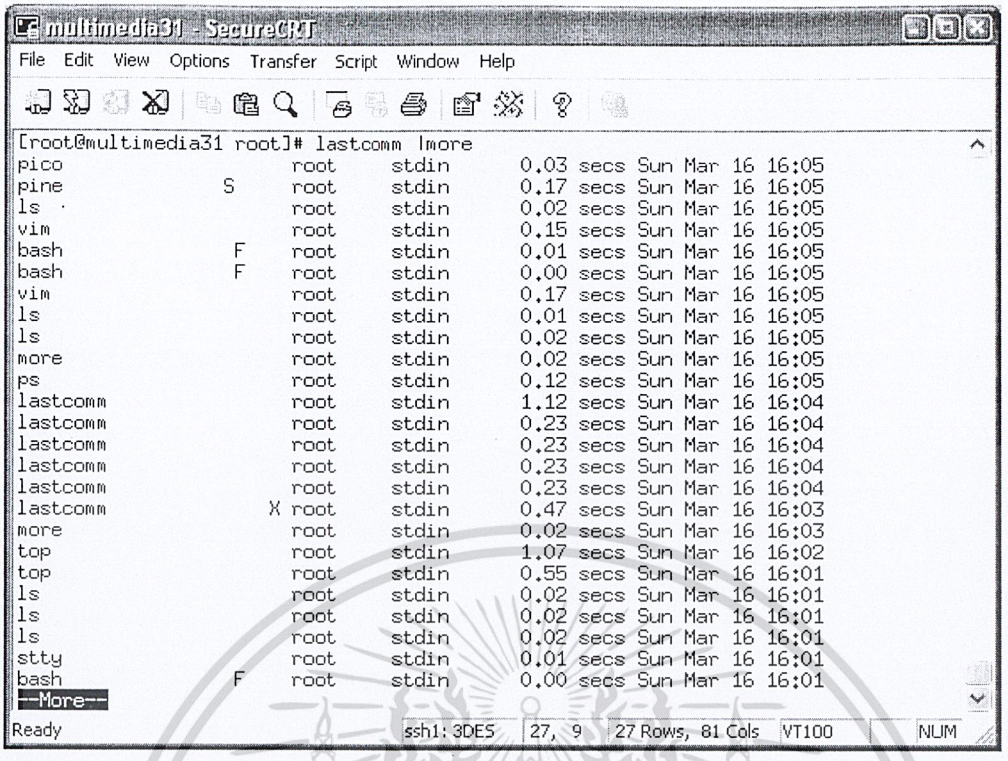
คำสั่ง lastcomm

เป็นคำสั่งที่ใช้แสดงข้อมูลการใช้งานคำสั่งหรือโพรเซสต่างๆที่ผ่านมา โดยจะดึงค่าจากไฟล์ pacct มาใช้งาน ถ้าไม่มีการระบบออปชั่นใดๆ จะแสดงข้อมูลทั้งหมดที่มีอยู่ในไฟล์ pacct ออกมา แต่เราสามารถระบุ ชื่อผู้ใช้งาน, คำสั่ง หรือชนิดของเทอร์มินอลที่เรียกใช้ตามหลังได้ด้วย เช่น

```
lastcomm a.out tty0
```

รายละเอียดของข้อมูลที่แสดงมีดังนี้

- คำสั่งหรือโพรเซสที่ถูกเรียกใช้
- แฟลก
  - S : คำสั่งนั้นถูกเรียกใช้โดย super-user
  - F : คำสั่งนั้นมีการ fork แต่ไม่มีการเรียกใช้งาน
  - C : คำสั่งนั้นทำงานในโหมดที่เข้ากันได้กับ PDP-11
  - D : คำสั่งมีการสร้างไฟล์ core เมื่อจบการทำงาน
  - X : คำสั่งส่งสัญญาณ SIGTERM เมื่อจบการทำงาน
- ชื่อผู้ที่เรียกใช้งาน
- เวลาที่จบการทำงาน



รูปที่ 3-6 แสดงสถิติของผู้ใช้งานโดยคำสั่ง lastcomm

3.3 ครอนและครอนแท็ป (Cron,Crontab)

ครอนเป็นระบบจัดการเอ็กซิคิวต์คำสั่งตามเวลาที่กำหนดไว้ล่วงหน้าโดยตัวครอนเดมอน (crond) ถูกรันจากไฟล์ /etc/rc หรือ /etc/rc.local โดยตัวเดมอนนี้เมื่อเริ่มต้นทำงานจะอ่านค่าเริ่มต้นจากไฟล์ /var/spool/cron ซึ่งเป็นส่วนที่ผู้ใช้แต่ละคนสามารถสร้างตารางเวลาของแต่ละคนได้ ชื่อไฟล์มีชื่อเดียวกันกับชื่อผู้ใช้ ส่วนอีกไฟล์หนึ่งเป็นการกำหนดตารางเวลาของระบบคือไฟล์ /etc/crontab โดยผู้ที่จัดการกับไฟล์นี้ได้ต้องเป็น root เท่านั้น

ครอนแท็ป เป็นไฟล์ที่ใช้กำหนดเวลาล่วงหน้าของระบบ อยู่ใน /etc/crontab คำสั่งที่รันจากครอนของระบบจะมีสิทธิ์ของ root ไฟล์ครอนแท็ปมีข้อกำหนดการเขียนคือ บรรทัดที่ว่างและบรรทัดที่ขึ้นต้นด้วย # ถือว่าเป็นคอมเมนต์ ในไฟล์ครอนแท็ปสามารถกำหนดค่าตัวแปรในรูปแบบดังนี้

```
name = value
```

รูปแบบการกำหนดตารางการทำงาน

ฟิลด์	minute	hour	day_of_month	month	day_of_week	[user]	command
ช่วงเวลาที่ใช้	0-59	0-23	0-31	0-12	0-7		

ถ้าค่าในฟิลด์เวลาเป็น * หมายถึงทุกค่าที่เป็นไปได้ นอกจากนี้ยังสามารถกำหนดค่าเป็นช่วง โดยใช้เครื่องหมาย – ได้ เช่น 8-11 ก็หมายถึงเวลา 8,9,10,11 หรือกำหนดเป็นลิสต์ได้เช่น 1,2,5,9 หรือ 1-4,8-12 ก็ได้

นอกจากนี้สามารถกำหนดเป็นสเต็ปได้ เช่น สำหรับ hour กำหนดเป็น 0-23/2 หมายถึง 0,2,4,6,8,10,12,14,16,18,20,22 ซึ่งหมายถึงให้ทำคำสั่งนั้นทุก 2 ชั่วโมง อาจเขียนง่ายๆได้ว่า */2 และเครื่องหมาย % มีความหมายเท่ากับเครื่องหมายขึ้นบรรทัดใหม่ (new line)

สำหรับฟิลด์ month และ day of week สามารถใช้ชื่อเป็น ภาษาอังกฤษสามตัวแรก เช่น อาทิตย์ก็เป็น sun เดือนธันวาคมก็จะเป็น Dec โดยไม่สนใจว่าจะเป็นตัวพิมพ์เล็กหรือพิมพ์ใหญ่

ถ้าใช้เป็นตัวเลขในฟิลด์ day of week เลขแต่ละตัวมีความหมายดังนี้

เลข 0,7 หมายถึง วันอาทิตย์

เลข 1 หมายถึง วันจันทร์

เลข 2 หมายถึง วันอังคาร

เลข 3 หมายถึง วันพุธ

เลข 4 หมายถึง วันพฤหัส

เลข 5 หมายถึง วันศุกร์

เลข 6 หมายถึง วันเสาร์

ตัวอย่าง

30	4	1,15	*	5	- หมายถึงคำสั่งรันที่เวลา 4:30 ทุกวันที่ 1 และ 15 ของทุกเดือนรวมถึงวันศุกร์ด้วย
*	*	*	*	*	- หมายถึงคำสั่งรันทุกหนึ่งนาทีของทุกวัน
23	0-23/2	*	*	*	- รันที่เวลา 23 นาทีหลังเที่ยงคืนแล้วรันทุกๆ 2 ชั่วโมงทุกวัน
15	4	*	*	sun	- รันทุกวันอาทิตย์เวลา 4:15

3.4 ภาษา C และภาษา C++

3.4.1 ภาษา C

ภาษา C เป็นภาษาเขียนโปรแกรมที่ถูกนำไปใช้อย่างกว้างขวาง โดยภาษา C เริ่มมีมาตั้งแต่ช่วงแรกๆ ที่ระบบปฏิบัติการยูนิกซ์ (UNIX) ถือกำเนิดขึ้นมา ภาษา C ได้ถูกคิดค้นขึ้นโดยนายเดนิส ริตชี (Denis Ritchie) ที่ห้องแล็บเบล โดยพัฒนาปรับปรุงมาจากภาษา B ภาษา C สามารถติดต่อในระดับฮาร์ดแวร์ได้ดีกว่าภาษาระดับสูงเช่น เบสิก (Basic) ฟอร์แทรน (Fortran) ขณะเดียวกันก็มีคุณสมบัติของภาษาระดับสูงอยู่ด้วย ดังนั้นจึงได้จัดให้ภาษา C เป็นภาษาระดับกลาง สำหรับที่ภาษา C ได้รับการยอมรับอย่างกว้างขวางนั้นก็เนื่องมาจากเหตุผลดังต่อไปนี้

- ภาษา C เป็นภาษาที่มีการกำหนดมาตรฐานสำหรับทุกแพลตฟอร์ม ทำให้โครงสร้างทางภาษา, ไลบรารี (library) และฟังก์ชันต่างๆสามารถนำไปใช้ข้ามแพลตฟอร์มได้
- โปรแกรมที่เขียนด้วยภาษา C ทำงานได้เร็ว

- ภาษา C เป็นภาษาระบบของยูนิกซ์ทุกเวอร์ชัน

ในช่วงปี 1980 ได้มากำหนดมาตรฐานของภาษา C ให้มีความเป็นสากล โดยมาตรฐานที่กำหนดขึ้นมานี้เรียกว่ามาตรฐาน ANSI C ซึ่งเป็นมาตรฐานที่ผู้ผลิตคอมไพเลอร์ภาษา C จะต้องอ้างอิง ทำให้ภาษา C สามารถใช้งานข้ามแพลตฟอร์มได้

3.4.2 ภาษา C++

C++ (ซีพลัสพลัส) เป็นภาษาระดับสูง (High-level language) ซึ่งกำลังได้รับความนิยมอยู่ในปัจจุบัน เนื่องจาก C++ ได้รวบรวมเทคนิคต่างๆ ซึ่งมีการคิดค้นและสั่งสมต่อเนื่องกันมาในช่วงเวลาหลายปี ดังนั้น C++ จึงเป็นภาษาที่มีประสิทธิภาพสูงสุดภาษาหนึ่งสำหรับการเขียนโปรแกรม

ผู้สร้างภาษา C++ คือ บจาร์น สโตรสรัป (Bjarne Stroustrup) ซึ่งทำงานอยู่ห้องปฏิบัติการเบล (Bell Laboratories) ประเทศสหรัฐอเมริกา ภาษานี้สร้างขึ้นประมาณปี ค.ศ. 1980

C++ ประกอบด้วยทุกสิ่งที่มีในภาษา C และส่วนที่เพิ่มเติมเข้ามา ดังนั้นสามารถเขียนโปรแกรมภาษา C ใน C++ ได้แต่ทำกลับกันคือเขียนภาษา C++ ใน C ไม่ได้ เป็นสาเหตุให้มีผู้นิยมใช้ C++ มากขึ้นเป็นลำดับ ส่วนสำคัญที่เพิ่มเติมจากภาษา C ได้แก่ การเขียนโปรแกรมแบบออบเจ็กต์ (Object) หรือ OOP (Object-Oriented Programming) ซึ่งเป็นวิธีการเขียนโปรแกรมที่สามารถใช้งานได้อย่างกว้างขวาง นิยมใช้ในการเขียนโปรแกรมขนาดใหญ่ที่มีการทำงานสลับซับซ้อน เพราะสะดวกและง่ายแก่การเขียน การตรวจสอบ การปรับปรุง และการนำมาใช้ใหม่

C++ ได้รับมาตรฐาน 4 มาตรฐาน คือ ANSI (The American National Standards Institute - สถาบันมาตรฐานแห่งชาติเมริกัน) มาตรฐาน ISO (International Organization of Standardization - องค์การระหว่างประเทศเกี่ยวกับมาตรฐาน) มาตรฐาน BSI (The British Standards Institute - สถาบันมาตรฐานแห่งอังกฤษ) มาตรฐาน DIN (The German National Standards Organization -องค์การมาตรฐานแห่งชาติเยอรมัน) การกำหนดมาตรฐานมีส่วนดีทำให้โปรแกรมที่เขียนขึ้นสามารถนำไปใช้ในระบบคอมพิวเตอร์ที่แตกต่างกันได้ทันทีโดยไม่ต้องปรับปรุงเปลี่ยนแปลงส่วนใดๆ การกำหนดมาตรฐานอาจเป็นเครื่องบ่งชี้ให้ทราบว่า C++ เป็นภาษาที่ได้รับความนิยมมากภาษาหนึ่ง

3.5 เชลล์สคริปต์ (Shell Script)

ในการติดต่อกับระบบปฏิบัติการลีนุกซ์มักติดต่อผ่านโปรแกรมเล็กๆ โปรแกรมหนึ่ง เรียกว่าเชลล์ (Shell) เชลล์เป็นชั้นของแอพพลิเคชันที่ตีความคำสั่งจากผู้ใช้และส่งต่อไปให้กับเคอร์เนลของระบบ จากนั้นแสดงผลลัพธ์ที่ได้จากเคอร์เนลล์กลับมาให้ผู้ใช้อีกที และจากการที่ส่วนใหญ่ผู้ใช้ต้องติดต่อกับระบบปฏิบัติการลีนุกซ์ผ่านทางเชลล์นี้เองจึงทำให้คนส่วนใหญ่เข้าใจว่าเชลล์ที่เห็นนั้นคือระบบปฏิบัติการ ซึ่งอันที่จริงแล้วเชลล์เป็นเพียงแอพพลิเคชันบนระบบเท่านั้น

3.5.1 ประวัติของเชลล์

ในระบบยูนิกซ์รวมถึงลีนุกซ์นั้น สามารถเลือกใช้เชลล์ได้หลายแบบ ซึ่งแตกต่างจากระบบดอสที่จำกัดให้ใช้เชลล์ที่ติดตั้งมาแล้วกับดอส คือ command.com ได้เพียงอย่างเดียว สำหรับโปรแกรมเชลล์ตัวแรกที่มีใช้กันนั้นคือ เบอร์นเชลล์ (Bourne Shell) ซึ่งตั้งชื่อตามชื่อผู้คิดค้นโปรแกรมนี้ขึ้น คือนาย สตีเฟน บอร์น (Sreven Bourne) เบอร์นเชลล์นี้แจกจ่ายไปพร้อมกับยูนิกซ์เวอร์ชัน 7 ซึ่งเป็นเวอร์ชันที่ได้รับความนิยมมากที่สุดตัวหนึ่งในปี ค.ศ.1979 ผู้ใช้ส่วนใหญ่รู้จักเบอร์นเชลล์จากการเรียกคำสั่งในระบบว่า sh ถึงแม้ว่าในปัจจุบันมีโปรแกรมเชลล์ออกมามากมาย เบอร์นเชลล์ก็ยังถือว่าเป็นโปรแกรมสำคัญพื้นฐานที่ยูนิกซ์ทุกระบบต้องมี นอกจากนี้โปรแกรมเชลล์ตัวอื่นๆ ที่ถูกพัฒนาขึ้นมาภายหลังส่วนใหญ่ก็มีความเข้ากันได้กับเบอร์นเชลล์อยู่ทั้งสิ้น

โปรแกรมเชลล์ที่พัฒนาขึ้นมาภายหลังและได้รับความนิยมอย่างสูงอีกตัวหนึ่งก็คือ ซีเชลล์ (C Shell) หรือ csh ซึ่งพัฒนาโดยนายบิล จอย (Bill Joy) (หนึ่งในผู้ก่อตั้งบริษัทซันไมโครซิสเต็ม (Sun Microsystem) ซึ่งเป็นผู้ผลิตและจำหน่ายระบบคอมพิวเตอร์ที่ใช้ยูนิกซ์รายใหญ่ที่สุดรายหนึ่งในปัจจุบัน) ซึ่งในสมัยนั้นอยู่ที่มหาวิทยาลัยแคลิฟอเนีย (California) โปรแกรมนี้ถือเป็นส่วนหนึ่งของระบบยูนิกซ์รุ่นของเบิร์กเลย์ หรือที่เรียกกันว่าเบิร์กเลย์ซิสเต็มดิสทริบิวชัน (Berkeley System Distribution) หรือ BSD Unix สาเหตุที่เรียกเชลล์นี้ว่าซีเชลล์ (C Shell) เพราะมีรูปแบบของคำสั่งและการใช้งานที่คล้ายกับโปรแกรมภาษา C ทำให้ผู้ใช้ที่เป็นโปรแกรมเมอร์มีความคุ้นเคยกับการใช้งานเชลล์นี้มาก

โปรแกรมเชลล์ตัวอื่นที่ถูกพัฒนาขึ้นภายหลังและได้รับความนิยมสูงอีกโปรแกรมหนึ่งก็คือคอร์นเชลล์ (Korn Shell) หรือ ksh ถูกพัฒนาขึ้นมาโดยนายเดวิด คอร์น (David Korn) จากเอทีแอนด์ทีเบลแลบ (AT&T Bell Lab) ในราวกลางทศวรรษ 1980 คอร์นเชลล์นี้มีความเข้ากันได้กับเบอร์นเชลล์ได้โดยง่าย นอกจากนี้คอร์นเชลล์ก็ยังมีรูปแบบพิเศษบางอย่างที่ถูกเพิ่มเติมเข้ามาเพื่อให้ผู้ใช้สามารถใช้งานได้สะดวกขึ้นด้วย

ในระบบลีนุกซ์สามารถเลือกติดตั้งเชลล์ได้หลายแบบ ซึ่งสามารถตรวจสอบได้ว่าขณะนี้กำลังใช้เชลล์อะไรอยู่โดยใช้คำสั่งต่อไปนี้

```
$ echo $SHELL
/bin/bash
```

ตัวอย่างนี้แสดงว่ากำลังใช้ bash หรือ Bourne again shell อยู่ หากต้องการเปลี่ยนหรือเรียกใช้งาน เชลล์ชนิดอื่นๆ ขึ้นมา ก็สามารถใช้คำสั่งโดยป้อนชื่อของเชลล์ที่ต้องการลงไปได้เช่น

```
$ echo (call Bourne again shell)
$  csh (call c shell)
```

หรือสามารถเปลี่ยนแปลงดีฟอลต์เชลล์ (default shell) ที่ถูกเรียกขึ้นมาทุกครั้งที่มีการล็อกอินเข้าใช้งานระบบได้จากแฟ้มข้อมูล /etc/passwd

3.5.2 การเขียนโปรแกรมเชลล์

การเขียนโปรแกรมเชลล์หรือ เชลล์สคริปต์ (Shell script) เป็นการนำเอาคำสั่งต่างๆ ของระบบมารวมเข้าด้วยกัน โดยมีคำสั่งควบคุมการทำงาน (flow control) คอยเป็นตัวบังคับ และควบคุมรูปแบบของโปรแกรม

- **การควบคุมลักษณะของการแสดงผล**

การควบคุมการแสดงผลบนจอภาพใช้รหัสพิเศษที่ขึ้นต้นด้วยเอสเคป (escape) หรือ \033 เรียกรหัสชุดเหล่านี้ว่าเอสเคปCเควนต์ (escape sequence) สามารถบังคับให้ระบบแสดงผลด้วยตัวอักษรแบบกระพริบ,แบบกลับขาวเป็นดำ หรือขีดเส้นใต้ได้ ตัวอย่างของรหัสต่างๆ ที่ใช้มีดังนี้

\033[line;colH	ตำแหน่งของเคอร์เซอร์ไปอยู่บนบรรทัด line และคอลัมน์ col
\033[2J	ลบหน้าจอ (คล้ายกับคำสั่ง tput clear บนยูนิกส์หรือคำสั่ง CLS บน DOS)
\033[4m	เข้าสู่โหมดขีดเส้นใต้ ตัวอักษรที่พิมพ์ต่อจากรหัสชุดนี้เป็นตัวอักษรที่มีการขีดเส้นใต้ทั้งหมด
\033[5m	เข้าสู่โหมดกระพริบ (blink)
\033[7m	โหมดกลับขาวดำ (reverse)
\033[m	กลับสู่โหมดปกติ

สามารถใช้คำสั่ง echo เพื่อส่งรหัสเหล่านี้ได้ สำหรับรหัส Esc ให้กดปุ่ม Ctrl-v (กดปุ่ม Ctrl ค้างไว้และตามด้วย v) หลังจากนี้ เมื่อกดปุ่มใดก็เป็นการส่งรหัสประจำปุ่มนั้นไปให้ระบบ ในที่นี้จะกดปุ่ม Esc เมื่อกดแล้วเห็นเป็นตัวอักษร "^[" หากต้องการกลับสู่โหมดตัวอักษรกระพริบ ต้องกดปุ่ม Ctrl-v ตามด้วยปุ่ม Esc ปุ่ม [,5 และ m ตามลำดับ

```
$ echo ^[[5m
```

เมื่อพิมพ์ตัวอักษรหลังจากนี้ เห็นเป็นตัวอักษรกระพริบทั้งหมด (สำหรับจอที่มีการกำหนดเป็นแอตทริบิวต์สี ก็จะเห็นเป็นสีที่แตกต่างจากปกติ) จากนั้นให้ป้อนคำสั่งต่อไปนี้เพื่อกลับสู่โหมดปกติ

```
$ echo ^[[m
```

ถ้าต้องการลบหน้าจอ ให้ใช้คำสั่ง tput clear คำสั่งต่างๆ สามารถนำมาใส่ไว้ในไฟล์เดียวกันได้ เรียกว่า เชลล์สคริปต์ และเรียกใช้งานโดยใช้คำสั่ง

```
$ bash myscript          (สำหรับ Bourne Again Shell)
```

หรือ

$ ksh myscript (สำหรับ Korn shell)

หรือใช้คำสั่ง chmod เพื่อเปลี่ยนโหมดของไฟล์ให้เป็นชนิดที่ทำงานได้ (executable) แล้วเรียกเชลล์สคริปต์นั้นให้ทำงานด้วยการป้อนชื่อเข้าไปโดยตรง ดังนี้

$ chmod +x myscript -> เปลี่ยนโหมดของไฟล์ให้เป็นแบบ executable
$ myscript -> เรียกให้สคริปต์ทำงานโดยพิมพ์ชื่อลงไปโดยตรง

ตัวอย่างซอร์สโค้ดของสคริปต์

```
tput clear
echo    "^[[07;20H+-------------------------------------------------+"
echo    "^[[07;38H^[[7m MENU  ^[[m"
echo    "^[[08;20H|                                                 |"
echo    "^[[09;20H|      1.Option 1.                                |"
echo    "^[[10;20H|      2.Option 2.                                |"
echo    "^[[11;20H|      3.Option 3.                                |"
echo    "^[[12;20H|      4.Option4.                                 |"
echo    "^[[13;20H|      5.Exit.                                    |"
echo    "^[[14;20H|                                                 |"
echo    "^[[15;20H+-------------------------------------------------+"
echo    "^[[16;20H|      Please select your choise  :"
read     choise
```

- **การเปลี่ยนแปลงทิศทางของช่องทางการสื่อสาร (I/O redirection)**

ในการสื่อสารระหว่างเทอร์มินอลกับตัวโฮสต์ของคอมพิวเตอร์มีช่องทางที่ใช้กันอยู่ 3 ทาง คือ STDIN (ช่องทางรับข้อมูลมาตรฐาน), STDOUT (ช่องทางแสดงผลลัพธ์มาตรฐาน) และ STDERR (ช่องทางแสดงผลลัพธ์ที่เป็นข้อผิดพลาดมาตรฐาน) ทั้งนี้ความสามารถในเรื่องการจัดการเทอร์มินอลหรือเรื่องของการเปลี่ยนทิศทางของช่องทางสื่อสาร (I/O redirection) ไม่ได้เป็นความสามารถจากโปรแกรมแอพพลิเคชันแต่เป็นความสามารถจากตัวระบบปฏิบัติการเอง

ลีนุกซ์ได้รับอิทธิพลมาจากระบบยูนิกซ์ จึงยังคงความสามารถเกี่ยวกับเรื่องของการจัดการเทอร์มินอลและการเปลี่ยนทิศทางของช่องทางสื่อสารไว้อย่างครบถ้วน จึงสามารถเปลี่ยนแปลงช่องทางการสื่อสารเหล่านี้จากเชลล์ได้ตลอดเวลา

- **การเปลี่ยนแปลงช่องทางการแสดงผลลัพธ์ออก (standard output redirection)**

ในที่นี้ยกตัวอย่างการใช้งานคำสั่ง "cat" ซึ่งรอรับข้อมูลจากคีย์บอร์ด แล้วแสดงผลลัพธ์กลับมาทางจอภาพ ดังนั้นในที่นี้ STDIN คือคีย์บอร์ด และ STDOUT คือจอภาพหรือมอนิเตอร์

```
$ cat
This is first line          ->    พิมพ์ข้อความ "This is first line"
This is first line          ->    cat แสดงผลลัพธ์ คือ ข้อความที่เพิ่งพิมพ์ลงไป
This is second line         ->    พิมพ์ข้อความ "This is second line"
This is second line         ->    cat แสดงผลลัพธ์ คือ ข้อความที่เพิ่งพิมพ์ลงไป
```

ต่อไปนี้จะเปลี่ยนช่องทางแสดงผลลัพธ์ของโปรแกรม ซึ่งปกติแสดงออกทางจอภาพให้ไปเก็บลงไฟล์แทน

```
$ cat
This is first line          ->    พิมพ์ข้อความ "This is first line"
This is second line         ->    พิมพ์ข้อความ "This is second line"
```

โปรแกรม cat จัดเก็บผลลัพธ์ลงในไฟล์ที่ชื่อ "my file" แทน นั่นคือ STDOUT ได้ถูกเปลี่ยนเป็นไฟล์ "myfile" แล้ว เครื่องหมาย ">" ใช้ในการเปลี่ยนช่องทางการแสดงผลลัพธ์ เป็นการเขียนทับไฟล์ที่ระบุชื่อ ถ้าต้องการให้เขียนผลลัพธ์ต่อท้ายข้อมูลเดิมที่มีอยู่แล้วในไฟล์ ต้องใช้เครื่องหมาย ">>" ดังตัวอย่าง

```
$ cat >> myfile
```

- **การเปลี่ยนแปลงช่องทางการนำข้อมูลเข้า (standard input redirection)**

ในที่นี้ ขอยกตัวอย่างคำสั่ง sort ซึ่งโดยปกติรับข้อมูลเข้าจากคีย์บอร์ดแล้วแสดงผลลัพธ์กลับออกมาทางจอภาพ ดังนั้น STDIN คือคีย์บอร์ด และ STDOUT คือจอภาพ

การทำงานของคำสั่ง sort

```
$ sort
Peter          ->    พิมพ์ Peter
Mary           ->    พิมพ์ Mary
John           ->    พิมพ์ John เสร็จแล้วกด Ctrl-D
John           ->    sort จัดเรียงข้อมูลใหม่
Mary
```

Peter

เตรียมไฟล์ชื่อ “namelist” ไว้ให้กับโปรแกรม sort

$ cat > namelist

Peter -> พิมพ์ Peter

Mary -> พิมพ์ Mary

John -> พิมพ์ John เสร็จแล้วกด Ctrl-D

การเปลี่ยนทิศทางของช่องทางนำข้อมูลเข้าจากคีย์บอร์ด ไปเป็นการนำข้อมูลเข้าจากไฟล์ที่ชื่อ “namelist” ทำดังต่อไปนี้

$ sort < namelist
Peter
Mary
John

กล่าวคือ เครื่องหมายที่ใช้ในการเปลี่ยนช่องทางนำข้อมูลเข้าคือ “<” จากตัวอย่างข้างต้น STDIN เปลี่ยนไปเป็นไฟล์ที่ชื่อ “namelist”

- **การเปลี่ยนทั้ง STDIN และ STDOUT พร้อมกัน**

ยกตัวอย่าง เช่น

$ sort < namelist > sortlist

คำสั่งนี้โปรแกรม sort รับข้อมูลจากไฟล์ “namelist” และเขียนผลลัพธ์ไปใส่ในไฟล์ “sortlist” สามารถลองใช้คำสั่ง cat ตรวจดูข้อมูลในไฟล์ “sortlist” ได้

- **การใช้งานไปป์ (pipe)**

การใช้งานไปป์ (pipe) เป็นการนำเอาผลลัพธ์จากคำสั่งหนึ่งไปเป็นข้อมูลเข้าของอีกคำสั่งหนึ่ง ซึ่งโปรแกรมที่มีลักษณะการใช้งานเป็นแบบฟิลเตอร์ ได้แก่ cat, sort, more, grep, tee และ tr

ตัวอย่างการใช้ไปป์

$ ls | grep file | sort

เป็นการเรียงลำดับข้อมูลที่ได้จากการคัดเลือกรายชื่อไฟล์ที่มีชื่อ “file” เป็นส่วนประกอบ

- **คำสั่งควบคุมทิศทางแบบวนรอบ (loop flow control)**

  คำสั่งควบคุมทิศทางแบบวนรอบหรือวนลูป มีอยู่ 3 แบบคือ for, while และ until

for loop

โครงสร้าง

```
For  VAR  in Arg-list
do
    command list
done
```

คำสั่ง for นี้ ทำงานวนรอบ จนกระทั่งสมาชิกใน “Arg-list” (หมายถึง argument list) ถูกใช้หมด ชุดคำสั่งในลูป for ต้องอยู่ภายใต้คำบังคับ “do” และ “done” การใช้ for ส่วนใหญ่ ใช้กับลูปที่รู้จำนวนรอบที่ต้องทำและ “Art-list” แน่นอน

ตัวอย่าง

```
SUM=0
for  i in  1  2  3  4  5  6  7  8  9
do
        SUM=’expr  $SUM + $ i’
done
echo  “Sum is $SUM”
```

ให้บันทึกเชลล์สคริปต์ข้างต้นเป็นไฟล์ชื่อ ex8 แล้วสั่งให้โปรแกรมทำงานดังนี้

```
$ ex8
Sum is  45
```

จากตัวอย่างเป็นการรวมตัวเลขจำนวนเต็มตั้งแต่ 1 ถึง 9 เสร็จแล้วพิมพ์ค่าที่ได้

while และ until

โครงสร้างของ while

```
while  [true condition]
do
    command list
done
```

คำสั่ง while ใช้กับเงื่อนไขที่เป็นจริง ตัวอย่าง เช่น

```
SUM=0
i  = 1
while [$i – lt   10]
do
        SUM=’expr  $SUM + $i’
        i =’expr  $i + 1’
done
echo  “Sum is $SUM”
```

โปรแกรมตัวอย่างข้างบนทำงานเหมือนกับ ex8 แต่ในที่นี้ใช้รูปแบบของ while loop โดยโปรแกรมบวกตัวเลข “i” ไปเรื่อยๆ เริ่มตั้งแต่ 1 และเพิ่มค่า i ขึ้นทีละหนึ่ง ลูปนี้ทำงานไปเรื่อยๆ ตราบใดที่ค่า i ยังน้อยกว่า 10 และเมื่อค่า i มากกว่าเท่ากับ 10 แล้ว ออกจากลูปและพิมพ์ผลบวกที่ได้ทั้งหมด

โครงสร้างของ until

```
until   [false condition]
do
    command list
done
```

คำสั่ง until ใช้กับเงื่อนไขที่เป็นเท็จ ตัวอย่างเช่น

```
SUM=0
i=1
until [$i –eq 10]
do
        SUM=’expr $SUM + $i’
        i=’exp $i +1’
done
echo “Sum is $Sum”
```

ตัวอย่างข้างต้นนี้ก็เป็นโปรแกรมที่ให้ผลลัพธ์แบบเดียวกับในตัวอย่างที่ใช้โครงสร้างของ for และ while นั่นเอง และจะเห็นว่าลักษณะคล้ายกับโปรแกรมที่ใช้โครงสร้างแบบ while-loop มาก เพียงแต่เงื่อนไขที่ใช้ตรวจสอบกลับกันเท่านั้น กล่าวคือเงื่อนไขนี้ให้ตรวจสอบว่าค่าของตัวแปร i เท่ากับ 10 หรือยัง ทั้งนี้ until-loop จะทำการวนลูปทำงาน ไปเรื่อยๆ ตราบใดที่เงื่อนไขที่ตามหลัง until มานั้นยังคงเป็นเท็จ

ในการเขียนโปรแกรมสามารถเลือกใช้เฉพาะ while-loop หรือ until-loop ตัวใดตัวหนึ่งเพียงอย่างเดียวก็ได้ เพราะเงื่อนไขของลูปทั้งสองแบบสามารถดัดแปลงให้ทดแทนกันได้ ดังที่ได้แสดงในโปรแกรมตัวอย่าง

Break และ continue

คำสั่ง break และ continue จะหยุดการทำงานของชุดคำสั่งที่ตามหลังสองคำสั่งนี้ สำหรับคำสั่ง break เมื่อหยุดแล้วก็กระโดดออกไปนอกลูป ตรงจุดที่อยู่ถัดจากคำบังคับ "done" ส่วน continue กลับไปทำงานในลูปต่อตรงจุดต้นลูปที่อยู่ถัดจากคำบังคับ "do" ดังตัวอย่างต่อไปนี้

```
SUM=0
While [true]
do
        echo –n "Please enter number 1-9, except 5, q=quit)"
        read INPUT
        if [ "$INPUT" = "q" ]
        then
                echo "Break loop now!"
                break
                elif [$INPUT –eq 5 ]
                then
                        echo "Please not input 5"
                        continue
                elif [$INPUT –gt 9 ]
                then
                        echo "Please do not input data > ?"
                        continue
                fi
                SUM='expr $SUM + $INPUT'
done
echo "Sum is $SUM"
```

ในตัวอย่างนี้นอกจากแสดงถึงการใช้ break กับ continue แล้ว ยังแสดงให้เห็นถึงการใช้ while ในแบบที่นิยมใช้กันอีกลักษณะหนึ่ง คือการมใช้กับเงื่อนไข "true" วิธีการใช้งานแบบนี้ทำให้โปรแกรมวนลูปแบบไม่รู้จบ ซึ่งมักใช้กับการวนทำงานเพื่อรับข้อมูลเข้าไปตามตัวอย่างข้างต้น โดยมีเงื่อนไขของการรับข้อมูลบางรูปแบบที่ใช้กระโดดออกจากลูปเพื่อจบการทำงาน ซึ่งในที่นี้ก็ใช้คำสั่ง "break" เมื่อผู้ใช้มีการป้อนตัวอักษร "q" เข้ามา ทั้งนี้อาจใช้คำสั่ง "exit" แทน "break" ก็ได้ แต่ทำให้ไม่มีการทำงานตามคำสั่งในส่วนที่พิมพ์ผลลัพธ์ ซึ่งอยู่ท้ายโปรแกรมไปด้วย

สำหรับคำสั่ง "continue" ใช้กับการป้อนข้อมูลที่เป็นตัวเลข 5 ซึ่งในที่นี้เป็นการข้ามการบวกเลข 5 ไป ตัวเลขที่ใส่ได้และโปรแกรมทำการบวกเลขให้มีเฉพาะตัวเลข 1-4 และ 6-9 เท่านั้น ส่วนตัวเลขที่น้อยกว่า 1 และมากกว่า 9 ก็จะถูกคำสั่ง "continue" ข้ามส่วนของโปรแกรมที่เป็นการบวกเลขไปเช่นเดียวกัน

คำสั่ง continue และ break นี้ กระโดดออกไปจากลูปชั้นที่โปรแกรมกำลังทำงานอยู่ แต่สามารถระบุให้ continue และ break กระโดดออกไปยังลูปที่ชั้นใดก็ได้ (ในกรณีที่ลูปมีการครอบซ้อนอยู่หลายชั้น) วิธีการใช้งานก็เพียงแต่เติมตัวเลขตามท้ายคำสั่ง break และ continue เท่านั้น ตัวอย่างเช่น "break2" ทำการกระโดดออกจากลูปไปสองชั้น (ในอยู่หลัง done ของลูปที่สอง)

3.5.3 ตัวแปรเชลล์

ตัวแปรเชลล์ (shell variable) มีการเก็บค่าไว้ในลักษณะของข้อความหรือสตริง (string) ในกรณีที่ใช้งานเป็นตัวเลขก็ต้องใช้โปรแกรม expr เข้ามาช่วยแปลงข้อมูลก่อน นอกจากนี้การกำหนดค่าให้กับตัวแปร ถือว่าเป็นการประกาศตัวแปรนั้นขึ้นมาด้วย ซึ่งมีรูปแบบดังต่อไปนี้

ชื่อตัวแปร=ค่าตัวแปร

โดยห้ามมีช่องว่าง (space) คั่นระหว่างเครื่องหมายเท่ากับ ตัวอย่างเช่น

NAME=KLAUSE

หากมีการกำหนดค่าที่เป็นประโยคข้อความยาวๆ ที่มีช่องว่างอยู่ในประโยคนั้น ต้องใช้เครื่องหมายอัญประกาศ (" ") ครอบประโยคนั้นเช่น

NAME="Peter Klause"

การตั้งชื่อของตัวแปรสามารถใช้ตัวอักษร ตัวเลข ตัวขีดเส้นใต้ (underscore) แต่ห้ามใช้ตัวอักษรพิเศษต่างๆ เช่น $, * , # , ( ) และห้ามขึ้นต้นด้วยตัวเลข ตัวอย่างชื่อตัวแปรเช่น VARI, VAR_NAME, Variable แต่โดยทั่วไปแล้วการตั้งชื่อของตัวแปรมักใช้อักษรตัวใหญ่ในการตั้งชื่อ ส่วนการใช้งานตัวแปรเชลล์หรือการอ้างถึงค่าของตัวแปร ต้องใช้เครื่องหมาย "$ " นำหน้าตัวแปรนั้นเช่น ถ้าใช้คำสั่ง
$echo $NAME

ในกรณีที่กำหนดค่าของตัวแปรเป็น "Peter Klause" ได้ผลลัพธ์เป็น Peter Klause แต่ถ้าใช้คำสั่ง $echo NAME เนื่องจากไม่ได้อ้างถึงค่าของตัวแปร ดังนั้นได้ผลลัพธ์เป็น

NAME

ในกรณีต้องการให้เชลล์สามารถแยกชื่อของตัวแปรออกจากค่าอื่นๆ สามารถทำได้โดยครอบชื่อของตัวแปรนั้นด้วยเครื่องหมายปีกกา "{ }" ดังตัวอย่าง

```
$ MYDIR=/home/mary/
$ cat ${MYDIR}myfile
```

คำสั่งข้างต้นเป็นการสั่งให้พิมพ์รายละเอียดของแฟ้มที่ชื่อ /home/mary/myfile (ซึ่งเป็นการเอาค่าของตัวแปร(home/mary/) มาต่อกับชื่อไฟล์(myfile)) ออกมาให้ หากไม่ครอบชื่อของตัวแปรด้วยเครื่องหมายปีกกาแล้ว เชลล์จะมองชื่อตัวแปรเป็น $MYDIRmyfile (ซึ่งไม่มีค่าอะไรอยู่)

การยกเลิกตัวแปรสามารถทำได้โดยใช้คำสั่ง

$ unset ชื่อตัวแปร

- **ขอบเขตของตัวแปรเชลล์**

ตัวแปรเชลล์ที่ถูกกำหนดขึ้นมามีลักษณะเป็น ตัวแปรท้องถิ่น (local variable) คือรู้จักเฉพาะในเชลล์ของชั้นตนเองเทานั้น เชลล์ในชั้นอื่นๆ ที่ถูกเรียกซ้อนกันจะไม่รู้จักตัวแปรดังกล่าว จนกว่าจะประกาศตัวแปรนี้ให้เชลล์ชั้นอื่นๆ ได้รู้จักก่อน ซึ่งทำได้โดยใช้คำสั่ง “export”

การเรียกเชลล์ซ้อนกันหลายชั้นหรือ “ซับเชลล์” (sub shell) หมายถึงการพิมพ์คำสั่ง “sh” หรือ “ksh” ซึ่งเป็นการเรียกเบอร์นเชลล์หรือคอร์นเชลล์ซ้อนขึ้นมาบนเชลล์ปัจจุบัน เมื่อเรียกขึ้นมาแล้วจะเห็นเป็นเครื่องหมาย prompt ของระบบตามปกติ ซึ่งหมายความว่าซับเชลล์ได้ถูกเรียกขึ้นมาใหม่แล้ว อาจจะใช้คำสั่ง “ps” เพื่อตรวจดูว่าซับเชลล์ถูกเรียกขึ้นมาหรือไม่ ดังนี้

```
$ ps
PID    TTY    STAT  TIME     COMMAND
962    p2     S     0:00     login -h localhost -p
963    p2     S     0:00     -bash
974    p2     R     0:00     ps
```

ตัวอย่างเชลล์แรกที่ลีนุกซ์กำหนดไว้ให้ เมื่อทำการล็อกอินเข้ามาใช้ระบบครั้งแรกคือ bourne again shell (bash) ซึ่งเป็นโพรเซสหมายเลข 963

```
$ sh
$ ps
PID    TTY    STAT  TIME     COMMAND
962    p2     S     0:00     login -h localhost -p
963    p2     S     0:00     -bash
975    p2     S     0:00     sh
976    p2     R     0:00     ps
```

เมื่อเรียกซับเชลล์เห็นโพรเซสหมายเลข 975 ซึ่งเป็นโพรเซสของเบอร์นเชลล์ที่เรียกด้วยคำสั่ง “sh”ขึ้นมา

```
$ exit
exit
$ ps
PID    TTY    STAT  TIME     COMMAND
```

```
962   p2   S   0:00   login -h localhost -p
963   p2   S   0:00   -bash
989   p2   R   0:00   ps
```

สามารถออกจากซับเชลล์ได้โดยการพิมพ์คำสั่ง “exit” ตามปกติ เมื่อใช้คำสั่ง exit เพื่อออกจากซับเชลล์ จะเห็นว่าโพรเซสของเบอร์นเชลล์หายไป เนื่องจากถูกฆ่าด้วยคำสั่ง exit ไปแล้ว แสดงว่าตอนนี้ได้กลับมาอยู่ที่เชลล์แรกซึ่งก็คือ bash อีกครั้ง (โพรเซสหมายเลข 963)

ตัวอย่างการเรียกซับเชลล์และการประกาศตัวแปรให้เชลล์ชั้นอื่น ๆ ได้รู้จัก

```
$ NAME=KLAUSE          ->   ประกาศตัวแปร NAME ในเชลล์ชั้นปัจจุบัน
$ sh                   ->   เรียกซับเชลล์ซ้อนขึ้นมา
$ echo  $NAME          ->   ไม่พิมพ์ผลลัพธ์ใดๆ ออกมา เนื่องจากซับเชลล์ไม่รู้จัก
$ exit                 ->   ออกจากซับเชลล์กลับมาสู่เชลล์ปัจจุบัน
$ export  NAME         ->   ประกาศตัวแปร NAME ให้กับเชลล์ชั้นอื่นๆ
$ sh                   ->   เรียกซับเชลล์อีกครั้งหนึ่ง
$ echo  $NAME
KLAUSE                 ->   เมื่อซับเชลล์รับรู้ตัวแปรแล้ว จึงพิมพ์ผลลัพธ์ออกมาให้
$ readonly  NAME       ->   คำสั่งนี้ทำให้ไม่สามารถเปลี่ยนแปลงค่าตัวแปรได้
```

- **ตัวแปรสงวนหรือตัวแปรระบบ**

ในระบบลีนุกซ์และยูนิกซ์มีตัวแปรสงวน (reserved) ที่ตัวระบบปฏิบัติการหรือโอเอส(OS) นำไปใช้ประโยชน์โดยเฉพาะเท่านั้น แต่หากรู้ความหมายของตัวแปรเหล่านั้นก็สามารถแก้ไขเปลี่ยนแปลงค่าของตัวแปรดังกล่าวได้ สามารถตรวจสอบค่าของตัวแปรระบบได้โดยใช้คำสั่งนี้

```
$ set
PPID=373
PS1=$
PS2=>
PS4=+
PWD=/home/twg
SHELL=/bin/bash
SHLVL=1
TERM=xterm
UID=406
```

```
USER=twg
WINDOWID=37748749
_=set
i=etc/profile.d/mh.sh
```

พบว่าระบบแสดงรายละเอียดของชื่อและค่าของตัวแปรสงวนต่างๆ ออกมาให้ ต่อไปเป็นคำอธิบายของตัวแปรสงวนบางตัว

$ HOME	->	Home directory
$ PATH	->	search patch เมื่อคำสั่งเรียกโปรแกรมให้ทำงานระบบทำการค้นหาตัวโปรแกรมนั้นจากรายชื่อของไดเรกทอรีที่ได้ระบุไว้ในตัวแปรนี้
$ MAIL	->	mailbox
$ PS1	->	prompt ตัวที่ 1 ของเชลล์
$ PS2	->	prompt ตัวที่ 2 ของเชลล์
$ LOGNAME	->	ชื่อทะเบียนผู้ใช้ระบบ (login name)
$ TERM	->	ชนิดของจอ

นอกจากการใช้คำสั่ง “set” เพื่อแสดงรายชื่อและค่าของตัวแปรต่างๆ ออกมาแล้ว ยังสามารถใช้คำสั่ง “echo <ชื่อตัวแปร >” เพื่อแสดงค่าของตัวแปรดังกล่าวออกมาก็ได้เช่นกัน และการเปลี่ยนหรือกำหนดค่าให้กับตัวแปรก็สามารถทำได้เช่นเดียวกับตัวแปรเชลล์ปกติทั่วไป ดังตัวอย่าง

$ echo $PS1	->	ให้แสดงค่าของตัวแปร PS1 (prompt ของระบบ)
$	->	ระบบพิมพ์ตัวอักษร “$” ซึ่งถูกใช้เป็น prompt ออกมาให้
$ PS1=”Linux:> “	->	กำหนดค่าของ prompt ใหม่เป็น “Linux:>”
Linux:>	->	prompt ของระบบเปลี่ยนเป็น “Linux:>”

3.5.4 การสร้างเชลล์สคริปต์

การสร้างเชลล์สคริปต์ เป็นการนำเอาคำสั่งต่างๆ ในลีนุกซ์มาประกอบเข้าเป็นไฟล์ จากคำสั่งต่อไปนี้ ลองสร้างไฟล์ชื่อ “ex1”

```
$ sh                        #call  subshell
$ echo   “Hello world”      #print  “Hello world”
$ date                      #print  date information
$ pwd                       #print current directory
$ exit                      #exit  from subshell
```

เมื่อบันทึกข้อมูลเก็บลงไฟล์ชื่อ “ex1” แล้ว สามารถสั่งให้โปรแกรมทำงานได้สองวิธี คือ

1) สั่งผ่านเชลล์ ให้พิมพ์คำสั่งดังนี้

$ sh ex1

วิธีการสั่งงานโปรแกรมแบบนี้ “ex1” ต้องถูกตั้งไว้ว่าอนุญาตให้สามารถอ่านไฟล์ได้ (readable) ไม่เช่นนั้นเชลล์จะไม่สามารถทำให้โปรแกรมทำงานได้

2) สั่งโดยตรงจากชื่อโปรแกรม ให้พิมพ์คำสั่ง

$ ex1

สำหรับวิธีนี้โปรแกรม “ex1” ต้องถูกตั้งว่าอนุญาตให้สามารถสั่งทำงานได้ (executable) ไม่เช่นนั้น เชลล์จะปฏิเสธการสั่งงานโปรแกรม (Permission denied)

สามารถกำหนดคุณสมบัติให้ไฟล์อนุญาตให้มีการสั่งทำงานได้โดยใช้คำสั่งดังนี้

$ chmod u+x ex1 อนุญาตให้สั่งทำงานได้ในระดับผู้ใช้

$ chmod g+x ex1 อนุญาตให้สั่งทำงานได้ในระดับกลุ่ม

$ chmod o+x ex1 อนุญาตให้สั่งทำงานได้ในระดับบุคคลอื่นๆ

$ chmod +x ex1 และ $ chmod a+x ex1 เปิดให้สั่งทำงานได้ในทุกระดับชั้น

หากกำหนดให้ไฟล์อนุญาตให้มีการสั่งทำงานได้แล้ว และพิมพ์ชื่อโปรแกรมเพื่อสั่งให้ทำงาน แต่ปรากฏว่าเชลล์ยังคงปฏิเสธการสั่งงาน (เช่นตอบว่า ex1 : command not found) แสดงว่าเชลล์ไม่ทราบว่าจะไปค้นหาชื่อโปรแกรมได้จากไดเรกทอรีใด วิธีการระบุชื่อที่อ้างอิงกับไดเรกทอรีก็ทำได้ 2 วิธีคือ

1) Relative Pathname วิธีระบุโดยอ้างอิงจากไดเรกทอรีปัจจุบัน

$ ./ ex1

2) Full Pathname ระบุแบบอ้างชื่อเต็ม เช่นสมมติว่าไฟล์อยู่ในไดเรกทอรี /home/user1

$ /home/user1/ex1

การเรียกใช้งานโปรแกรมโดยที่ต้องระบุชื่อไดเรกทอรีด้วย ค่อนข้างเป็นเรื่องยุ่งยาก หากเก็บโปรแกรมไว้ในไดเรกทอรีเฉพาะ เช่น /home/user1/shell สำหรับเก็บโปรแกรมเชลล์ที่เรียกใช้บ่อยๆ ก็ทำให้สะดวกขึ้น ในระบบลีนุกซ์มีวิธีการกำหนดให้เชลล์ทำการค้นหาโปรแกรมที่สามารถสั่งให้ทำงานได้จากไดเรกทอรีกลุ่มหนึ่ง ตามที่ระบุไว้ในตัวแปรของระบบชื่อ PATH (หรือที่เรียกว่า search path) สามารถตรวจสอบชื่อไดเรกทอรีที่เชลล์ค้นหาเมื่อมีการเรียกใช้งานโปรแกรมโดยไม่ระบุชื่อไดเรกทอรีได้ โดยการใช้คำสั่ง

$ echo SPATH

/usr/local/bin:/bin:/user/bin:/usr/X11R6/bin

จากผลลัพธ์ข้างต้น หากมีการสั่งงานโปรแกรมโดยไม่ระบุชื่อไดเรกทอรีดังกรณีข้างต้น เชลล์ค้นหาโปรแกรมจากไดเรกทอรีทั้งสี่ที่ระบุไว้ในตัวแปร PATH คือ /usr/local/bin , /bin, /usr/bin และ /usr/X11R6/bin ซึ่งแต่ละไดเรกทอรีเหล่านี้ไว้สำหรับค้นหาและเรียกโปรแกรมที่ใช้บ่อยๆ (เช่น “ls” อยู่

ในไดเรกทอรี “/bin” และ “ftp” อยู่ในไดเรกทอรี “/user/bin” เป็นต้น) สำหรับโปรแกรม ex1 นั้น เนื่องจากอยู่ในไดเร็คทอรี /home/user1 ดังนั้นเชลล์จึงไม่สามารถค้นหาพบได้ จึงต้องเพิ่มไดเรกทอรีนี้ลงไปในตัวแปร PATH เอง

วิธีการเพิ่มไดเรกทอรีใช้คำสั่งดังต่อไปนี้

```
$ PATH=$ PATH:/home/user1
$ export PATH
```

สำหรับคำสั่งแรกเป็นการนำเอาชื่อของกลุ่มไดเรกทอรีเดิม ($PATH) มาต่อเข้ากับไดเรกทอรีใหม่ (/home/user) หากไม่ระบุชื่อไดเรกทอรีเดิม ($PATH) กลายเป็นการแทนที่ของเดิมทั้งหมดด้วยของใหม่ไป ซึ่งถ้าเป็นเช่นนั้นทำให้ไม่สามารถเรียกใช้โปรแกรมที่เคยใช้ได้อีก (เช่น โปรแกรม ls) นอกจากนี้ หากต้องการให้มีการสั่งงานโปรแกรมที่อยู่ในไดเรกทอรีปัจจุบัน (current directory) ได้ด้วย ต้องใช้คำสั่งต่อไปนี้

```
$ PATH=$ PATH
$ export PATH
```

คำสั่งข้างต้นเป็นการเพิ่มไดเรกทอรีปัจจุบัน (:.) ลงไปในตัวแปร PATH ด้วย คือจุด (dot) “.” ซึ่งเป็นการแทนความหมายของไดเรกทอรีปัจจุบัน ให้ใช้คำสั่ง “echo” เพื่อตรวจดูว่าได้เพิ่มไดเรกทอรีเข้าไปในตัวแปร PATH เรียบร้อยหรือไม่ ดังนี้

```
$ echo   $PATH
/usr/local/bin:/bin:/ust/bin:/usr/X11R6/cin:/home/usr1:.
```

สำหรับการกำหนดให้เพิ่มไดเรกทอรีปัจจุบัน (:.) ลงไปในตัวแปร PATH นี้ ถ้าคำนึงถึงเรื่องของการรักษาความปลอดภัยแล้วเป็นเรื่องที่ไม่ควรทำอย่างยิ่ง ปกติระบบยูนิกซ์ส่วนใหญ่ไม่นิยมให้ผู้ใช้กลุ่มที่อยู่ในระดับของการดูแลรักษาระบบและมีสิทธิ์สูงที่สุดในระบบอย่าง root มีไดเรกทอรีปัจจุบัน (:.) อยู่ในตัวแปร PATH เลยเพราะหากกำหนดให้ไดเรกเทอรีปัจจุบันอยู่ในตัวแปร PATH แล้ว อาจมีการอัพโหลดโปรแกรมหรือสคริปต์ที่มีจุดประสงค์ในการบุกรุกระบบไว้ในไดเรกทอรีปัจจุบัน และผู้บุกรุกที่เข้าใช้ระบบก็สามารถเรียกโปรแกรมให้ทำงานได้ทันที โดยไม่จำเป็นต้องระบุไดเรกทอรีปัจจุบัน และผู้บุกรุกที่เข้าใช้ระบบก็สามารถเรียกโปรแกรมให้ทำงานได้ทันที โดยไม่จำเป็นต้องระบุไดเรกทอรีเต็มรูปแบบกำกับไปด้วยวิธีการกำหนดตัวแปร PATH แบบนี้ช่วยป้องกันการบุกรุกระบบได้ในระดับหนึ่ง

โดยปกติแล้วเมื่อเรียกเชลล์สคริปต์ให้เริ่มทำงาน หมายถึงการเริ่มต้นทำคำสั่งในสคริปต์ทีละคำสั่งตามลำดับ โดยก่อนการทำคำสั่งแรกในสคริปต์ก็จะมีการเรียกซับเชลล์ให้ด้วย และหลังจากทำคำสั่งทุกคำสั่งในเชลล์สคริปต์เสร็จสิ้นแล้วก็เสมือนกับการออกจากซับเชลล์นั้นด้วยคำสั่ง “exit”

แม้ว่าไม่ได้ใส่คำสั่ง exit ไว้ในเชลล์สคริปต์ แต่ก็เสมือนว่ามีคำสั่งนั้นอยู่ท้ายเชลล์สคริปต์ด้วย และโดยทั่วไปหากเป็นการจบการทำงานแบบปกติที่ไม่มีข้อผิดพลาดเกิดขึ้น มีการส่งรหัสค่า “0” (return code0) ออกมาให้ (คำสั่ง exit ที่ไม่มีตัวเลขตามท้ายนี้ถือว่าส่ง return code 0) แต่ถ้าเป็นการจบการ

ทำงานแบบไม่ปกติ คืออาจมีข้อผิดพลาดเกิดขึ้น ก็ส่งรหัสในค่าที่ไม่เป็น "0" ออกมาให้ (ปกติเป็นรหัส "1") สำหรับรหัสค่าศูนย์ดังกล่าวนี้ นำไปใช้ทดสอบเงื่อนไขร่วมกันกับคำสั่งอื่นๆ ในเชลล์ต่อไป

ที่จริงไม่ได้มีการใส่ทั้งคำสั่งในการเรียกซับเชลล์และคำสั่งที่ให้ออกจากซับเชลล์นั้นลงไปในเชลล์สคริปต์ด้วย แต่เมื่อสั่งให้เชลล์สคริปต์ทำงานแล้ว โปรแกรมเชลล์จะเติมคำสั่งทั้งสองลงไปให้โดยอัตโนมัติ นั่นหมายความว่าโดยปกติแล้วทุกคำสั่งในเชลล์สคริปต์ต้องทำงานภายใต้ซับเชลล์ ซึ่งการทำงานภายใต้ซับเชลล์นี้ก็มีข้อจำกัดดังที่อธิบายแล้ว นั่นคือภายในซับเชลล์ไม่สามารถเรียกใช้งานตัวแปรจากเชลล์ชั้นบนก่อนหน้านั้นทั้งหมดได้ หากไม่ได้ทำการเอ็กซ์พอร์ตตัวแปรนั้นๆ เสียก่อน อย่างไรก็ตามก็มีวิธีที่บังคับให้เชลล์สคริปต์ทำงานในเชลล์ชั้นปัจจุบันได้โดยไม่ไปเรียกซับเชลล์ ดังจะได้อธิบายต่อไป

3.5.5 วิธีการสั่งให้เชลล์สคริปต์ทำงาน

รูปแบบการสั่งงานเชลล์สคริปต์ มีได้ 4 แบบดังต่อไปนี้

แบบที่ 1

$ script-name สคริปต์ต้องการ execute permission และวิ่งที่ subshell

แบบที่ 2

$ sh script-name สคริปต์ต้องการ read permission และวิ่งที่ subshell

แบบที่ 3

$ .script-name สคริปต์วิ่งที่ current shell

แบบที่ 4

$ exec scriptname สคริปต์วิ่งที่ current shell โดยทับเชลล์ชั้นปัจจุบันที่ทำงานอยู่เดิม

สำหรับการสั่งงานเชลล์สคริปต์ในแบบที่ 1 และแบบที่ 2 เมื่อล็อกอินเข้ามา ได้เชลล์ใช้งาน (ซึ่งปกติลีนุกซ์เป็น bash) ในที่นี้เรียกว่าเชลล์เริ่มต้น (Parent Shell) เมื่อมีการสั่งให้เชลล์สคริปต์ทำงานตามรูปแบบที่ 1 หรือ 2 ก็เปรียบเสมือนมีการเรียกซับเชลล์เกิดขึ้น (คำสั่ง sh) เมื่อเรียกซับเชลล์แล้วก็ทำงานตามคำสั่งในเชลล์สคริปต์ทีละคำสั่งจนจบ ก็กลับไปสู่เชลล์เริ่มต้น (คำสั่ง exit) อีกทีหนึ่ง ในกรณีนี้หากเชลล์สคริปต์ต้องการใช้งานตัวแปรที่อยู่ในเชลล์เริ่มต้น ต้องมีการเอ็กซ์พอร์ตตัวแปรนั้นๆ ออกมาจากเชลล์เริ่มต้นเสียก่อน

สำหรับการสั่งงานในแบบที่ 3 มีผลให้เชลล์สคริปต์ทำงานในเชลล์ชั้นปัจจุบันทันที (ในที่นี้คือเชลล์เริ่มต้น) และทำงานตามคำสั่งในเชลล์สคริปต์ทีละคำสั่งจนจบ โดยไม่มีการเรียกซับเชลล์ขึ้นมา ซึ่งทำให้สามารถใช้งานตัวแปรเชลล์ในเชลล์เริ่มต้นได้ทันที โดยไม่ต้องเอ็กซ์พอร์ตตัวแปรออกมาก่อน

สำหรับการสั่งงานในแบบที่ 4 เชลล์เริ่มต้นถูกทำลายไป แล้วนำเอาเชลล์สคริปต์มาแทนที่โพรเซสของ Parent Shell ที่ถูกทำลาย ดังนั้น เมื่อเชลล์สคริปต์ทำงานทีละคำสั่งจนจบแล้วก็กลับไปสู่หน้าจอของการล็อกอินโดยอัตโนมัติ

บทที่ 4
ปัญญาประดิษฐ์

เป็นเวลากว่าหลายศตวรรษมาแล้วที่นักวิทยาศาสตร์ ต้องการเรียนรู้และเข้าใจจิตใจของมนุษย์ แต่ปัญหาที่เกิดขึ้นก็ยังคงไร้ซึ่งคำตอบ ถึงกระนั้นก็ตามก็ยังมีนักวิทยาศาสตร์บางกลุ่มทำการสร้างและพัฒนาโปรแกรมเพื่อให้มีความสามารถเทียบเท่ามนุษย์ คือสามารถรู้สึกนึกคิดได้ โดยสิ่งที่พวกเขาได้สร้างขึ้นมานั้นเราเรียกว่า ปัญญาประดิษฐ์

ปัญญาประดิษฐ์ สามารถจำกัดความได้ 2 ความหมาย คือ

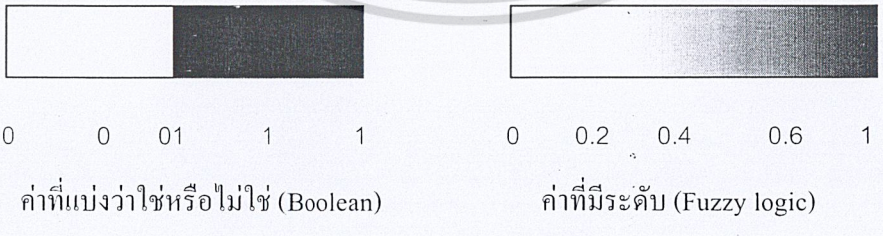
1) สิ่งที่สามารถเข้าใจ และเรียนรู้เองได้

2) สิ่งที่สามารถคิด และเข้าใจในสิ่งที่ได้กระทำลงไปโดยสามัญสำนึก หรือทำโดยอัตโนมัติ

ความคิดที่สามารถแยกแยะสิ่งต่างๆ ที่มีความคลุมเครือได้ และการเรียนรู้ด้วยตัวเอง หนึ่งในวิธีจัดการกับปัญหาที่เกิดขึ้นคือ การนำฟัซซี่ลอจิก (Fuzzy Logic) และนิวรอนเน็ตเวิร์ค (Neural Network) มาใช้งาน

4.1 ฟัซซี่ลอจิก (Fuzzy Logic)

ฟัซซี่ลอจิก เป็นการแก้ปัญหาที่ไม่สามารถระบบช่วงของข้อมูลได้ชัดเจน เช่น การแยกความสูงของมนุษย์ ซึ่งไม่สามารถระบบขอบเขตที่ชัดเจนในการแบ่งได้ว่าใครสูงใครต่ำ ดังนั้นจึงมีการนำฟัซซี่ลอจิกมาใช้งานเพื่อแยกแยะระดับต่างๆ โดยแต่ละระดับจะมีค่าอยู่ระหว่าง 0-1 โดยใช้เป็นตัวกำหนดค่าความเป็นสมาชิก (Membership) คือ หากมีค่าเป็น 0 แสดงว่าไม่มีความเป็นสมาชิกเลย 1 คือมีความเป็นสมาชิก(Membership)มากที่สุด และค่าที่อยู่ระหว่าง 0-1 ก็คือระดับการเป็นสมาชิก เช่น ความสูง อาจมีการระบุว่าบุคคลผู้นี้มีความสูงมาหรือน้อยเป็นเช่นไร สามารถแบ่งระดับดังนี้ เตี้ยมาก เตี้ย ปานกลาง สูง สูงมาก เป็นต้น



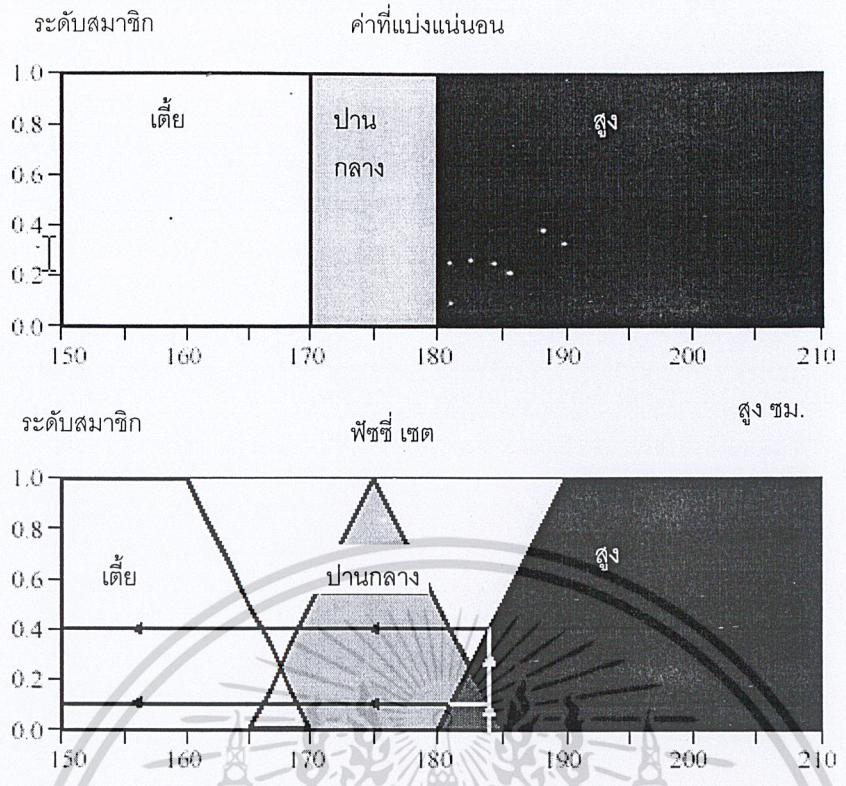
รูปที่ 4-1 แสดงการแบ่งระดับระหว่างบูลีน(Boolean) และ ฟัซซี่ลอจิก(fuzzy)

ชื่อ	ความสูง	ระดับความสูง	
		บูลีน	ฟัซซี่
สมชาย	208	1	1.00
สมยศ	205	1	1.00
สมรักษ์	198	1	0.98
สมสมัย	181	0	0.82
สมศรี	179	0	0.74
สมพร	172	0	0.24
สมพิศ	167	0	0.15
สมหมาย	158	0	0.06
สมจิต	155	0	0.01
สมใจ	152	0	0.00

ตารางที่ 4-1 แสดงตัวอย่างการแบ่งระดับความสูงโดยใช้บูลีนและฟัซซี่

4.1.1 การนำฟัซซี่มาใช้งานในคอมพิวเตอร์

ฟัซซี่ลอจิกเป็นระบบการทำงานหนึ่งที่นำมาใช้งานในเรื่องของการเรียนรู้ เช่น ระบบต้องการทราบความคิดเห็นเป็นเช่นไร จากตัวอย่างในเรื่องของความสูง เราจะมีการแบ่งค่าต่างๆ ออกเป็นระดับ 3 ระดับ คือ เตี้ย ปานกลาง และ สูง แต่หากเป็นฟัซซี่ลอจิกบุคคลที่มีความสูง 184 ซม. จะมีค่าความเป็นสมาชิกในระดับปานกลางอยู่ 0.1 และมีค่าสมาชิกในระดับสูงอยู่ 0.4 หมายความว่าความสูง 184 อยู่ในเซตหลายๆ เซต



รูปที่ 4-2 แสดงการทำฟัซซี่เซต (fuzzy set)

ฟัซซี่เซต (fuzzy set) คือ ค่าที่ใช้ในการแบ่งขอบเขตของฟัซซี่ เช่น

$$f_A(x) = \begin{cases} 1, \text{if } x \in A \\ 0, \text{if } x \notin A \end{cases}$$

ลิงกีคทิค วาริเอเบิ้ล (linguistic variable) คือ ค่าที่ใช้อธิบายค่าของฟัซซี่ลอจิก ว่ามีค่าเป็นเช่นใดบาง เป็นค่าที่แสดงอยู่ในฟัซซี่เซต เช่น เตี้ย ปานกลาง สูง

กฎของฟัซซี่ สามารถอธิบายเงื่อนไขของแต่ละอันในฟอร์ม เช่น

กฎข้อที่ 1

ถ้า ระดับความเร่งมีความเร็วสูง แล้ว ระยะการหยุดยาว

กฎข้อที่ 2

ถ้า ระดับความเร่งมีความเร็วต่ำ แล้ว ระยะการหยุดสั้น

4.1.2 วิธีการทำงานของฟัซซี่ลอจิก

ในการทำงานของฟัซซี่มีเทคนิคที่นำมาใช้งานอยู่ 2 รูปแบบคือ

1) **แมนดานิ สไตล์ (Mandani-style)** โดยมีขั้นตอนการทำงานแบ่งออกเป็น 4 ขั้นตอน คือ นำค่าอินพุตมาทำฟัซซี่ (fuzzification) คำนวณค่าจากกฎ (Rule evaluation) สรุปผลกฎต่างๆ ออกเป็นเอาท์พุต (aggregation of the rule outputs) ประมวลผลออกเป็นค่า (defuzzification)

ตัวอย่างของ Mandani-style

กำหนดให้ ปัญหาประกอบด้วย 2 อินพุต 1 เอาท์พุต และมี 3 กฎ

กฎข้อที่ 1	***กฎข้อที่ 1***
ถ้า x = A3 หรือ y = B1 **แล้ว** z = C1	ถ้า น้ำหนักมาก หรือ คนสูงมาก แล้ว ตัวใหญ่
ถ้า x = A2 หรือ y = B2 **แล้ว** z = C2	ถ้า น้ำหนักปานกลาง และ คนสูงปานกลาง แล้ว ตัวปานกลาง
ถ้า x = A1 **แล้ว** C3	ถ้า คนน้ำหนักน้อย แล้ว ตัวเล็ก

ขั้นตอนที่1 การทำฟัซซี่ (Fuzzification)

นำค่าอินพุตที่มีมากำหนดเป็นฟัซซี่

ขั้นตอนที่ 2 คำนวณหาค่าจากกฎ (Rule evolution)

$\mu_{(x=A1)} = 0.5$, $\mu_{(x=A2)} = 0.2$, $\mu_{(y=B1)} = 0.1$ และ $\mu_{(y=B2)} = 0.7$

กฎข้อที่ 1

$\mu_{C1}(z) = \max[\mu_{A3}(x), \mu_{B1}(y)] = \max[0.0,0.1] = 0.1$

หรือ

$\mu_{C1}(z) = \text{probor}[\mu_{A3}(x), \mu_{B1}(y)] = 0.0+0.1-0.0*0.1 = 0.1$

กฎข้อที่ 2

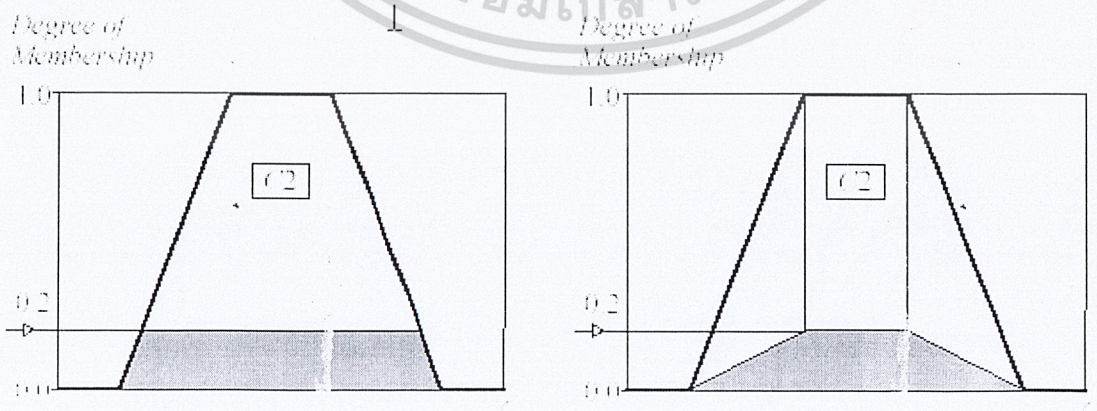
$\mu_{C2}(z) = \min[\mu_{A2}(x), \mu_{B}2(y)] = \min[0.2,0.7] = 0.2$

หรือ

$\mu_{c2}(z) = \text{prod}[\mu_{A2}(x), \mu_{B2}(y)] = 0.2*0.7 = 0.14$

ขั้นตอนที่ 3 สรุปผลกฎต่างๆ ออกเป็นเอาท์พุต (Aggregation of the rule outputs)

ในการคำนวณผลเอาท์พุตแบ่งเป็น 2 ส่วนคือ แบบสเกล (Scale) และแบบเมมเบอร์ชิฟ ฟังก์ชัน (membership function)



รูปที่ 4-3 แสดงการหาค่าเอาท์พุต ด้วยวิธีแบบสเกล และเมมเบอร์ชิฟ ฟังก์ชัน

ขั้นตอนที่ 4 ประมวลผลออกเป็นค่า (Defuzzification)

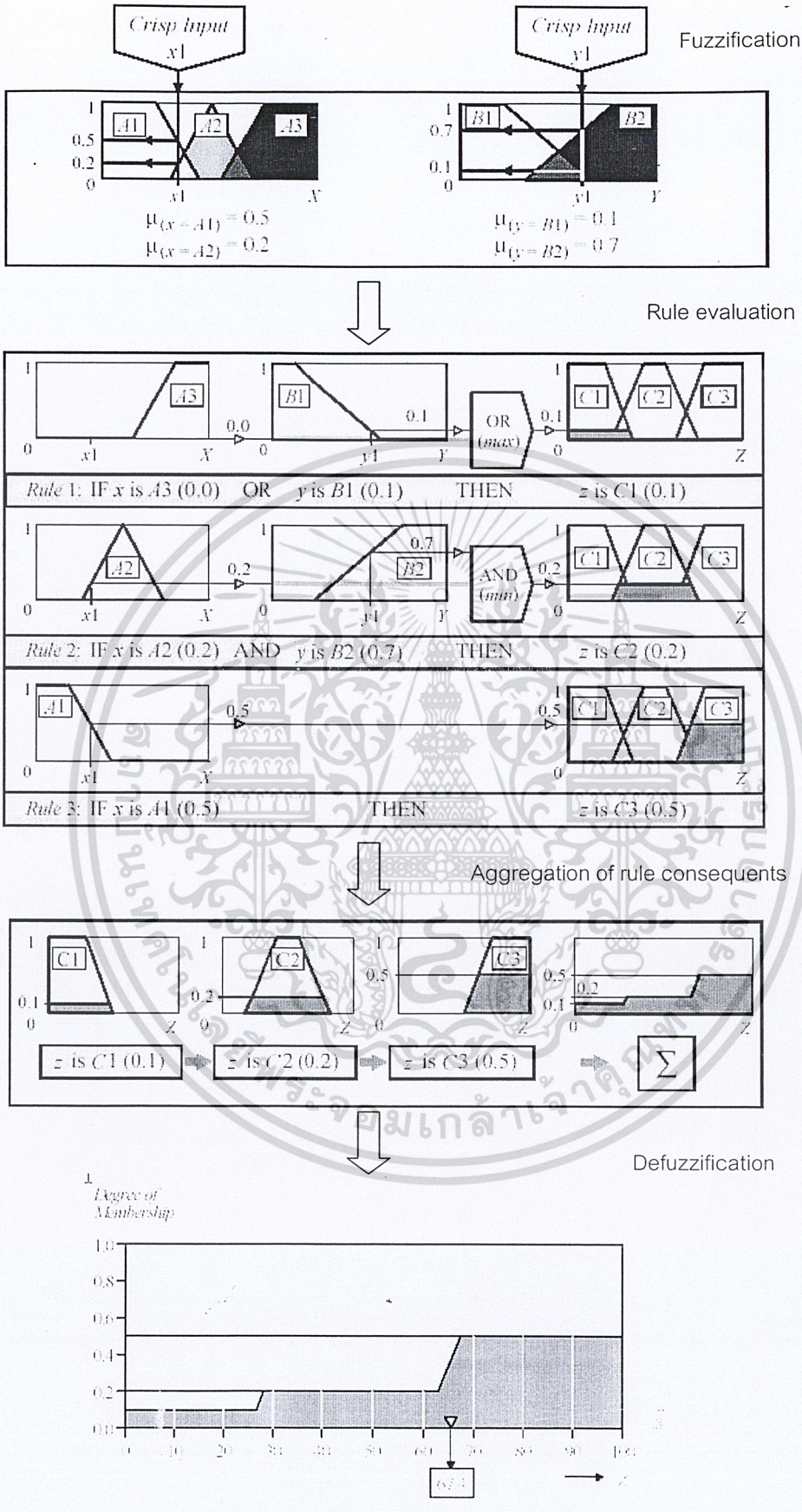
การทำฟัซซี่ทำให้สามารถคำนวณค่ามาจากกฎได้ อย่างไรก็ตามท้ายที่สุดแล้วผลที่ได้ก็ออกมาเป็นตัวเลขที่แน่นอนเพียงตัวเดียว เราเรียกวิธีนี้ว่า ดีฟัซซี่

โดยมีการหาค่า centre of gravity (COG) ดังนี้

$$COG = \frac{\int_a^b \mu_A(x)\, x\, dx}{\int_a^b \mu_A(x)\, dx}$$

ดังนั้นสามารถคำนวณค่าได้ดังนี้

$$COG = \frac{(0+10+20)*0.1+(30+40+50+60)*0.2+(70+80+90+100)*0.5}{0.1+0.1+0.1+0.2+0.2+0.2+0.2+0.5+0.5+0.5+0.5} = 67.4$$



ปที่ 4-4 แสดงลำดับขั้นตอนการทำงานของ Mandani-style

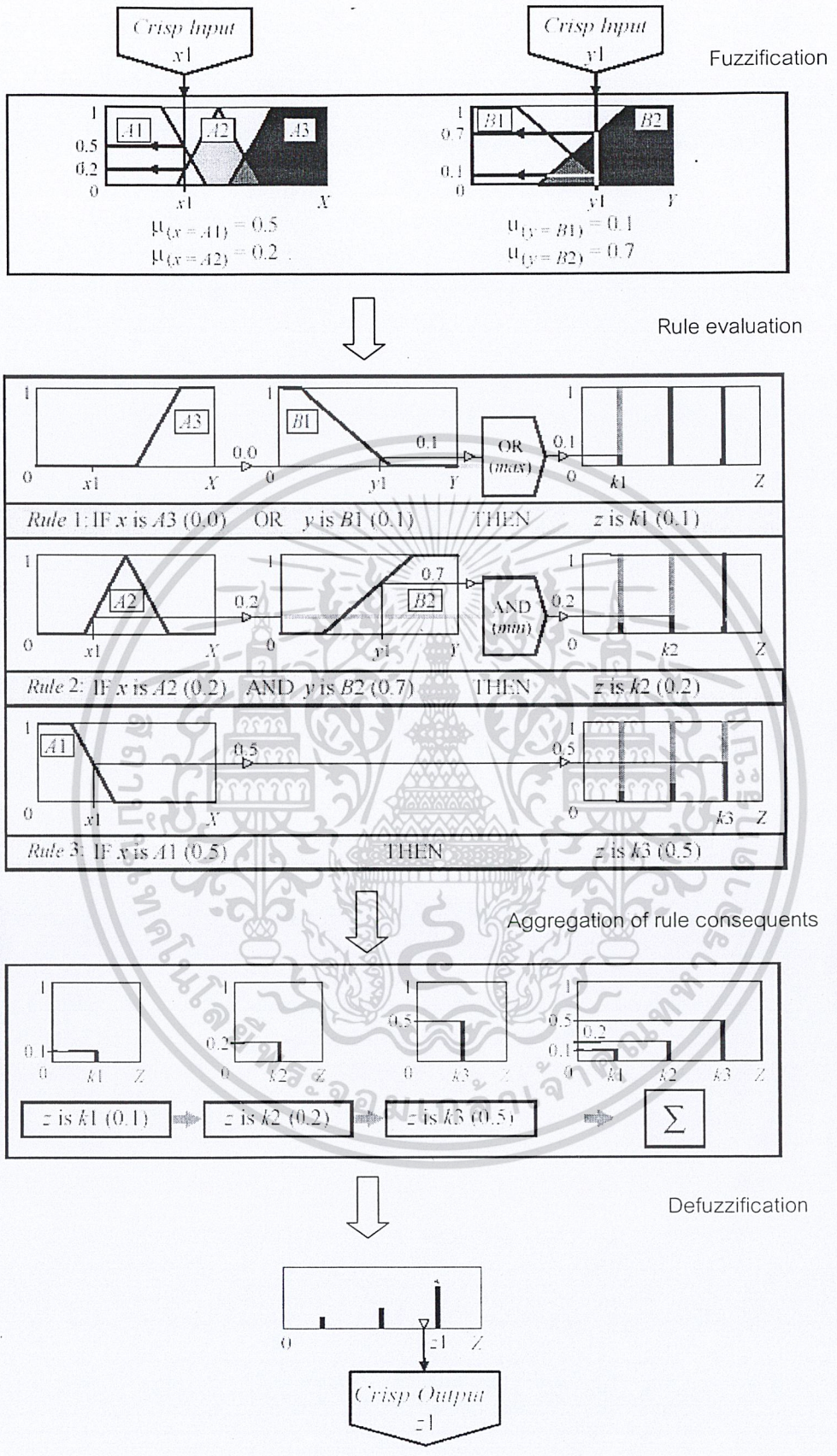
2) สูจิโน สไตล์ (Sugeno-style) มีขั้นตอนการทำงานเหมือนกับ Mandani-style จะแตกต่างกันในส่วนของ การสรุปผลต่างๆออกเป็นเอาท์พุต โดยในการคำนวณเราจะหาค่าออกมาเป็นค่าน้ำหนักเฉลี่ย (weight average) มีหลักการคำนวณดังนี้

$$WA = \frac{\mu(k1)*k1+\mu(k2)*k2+\mu(k3)*k3}{\mu(k1)+\mu(k2)+\mu(k3)} \quad \frac{=0.1*20+0.2*50+0.5*80}{0.1+0.2+0.5} = 65$$

4.1.3 เหตุผลในการเลือกวิธีใช้งานระหว่าง Mandani-style และ Sugeno-style

Mandani-style จะเป็นวิธีการที่นำมาใช้งานคาดเดาสิ่งที่ยังไม่ปรากฏ เช่น ฝนตก น้ำท่วม เป็นต้น รวมถึงความรู้ของมนุษย์

Sugeno-style จะเป็นการคำนวณค่าผลกระทบ และ ค่าการทำงาน ด้วยค่าที่ดีที่สุด หรือการพัฒนาเทคนิค ของการควบคุมปัญหา และแบ่งเป็นส่วนสำหรับการทำค่าที่เคลื่อนไหวโดยไม่เป็นเส้นตรง



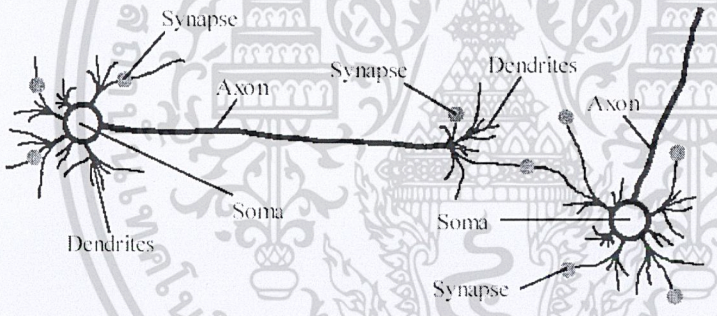
รูปที่ 4-5 แสดงลำดับขั้นตอนการทำงานของ **Sugeno-style**

4.2 การเรียนรู้

อุปกรณ์ต่างๆ ที่นำคอมพิวเตอร์เข้าไปใช้งานเพื่อการเรียนรู้นั้น จะเรียนรู้ได้โดยอาศัยประสบการณ์ ตัวอย่าง และ อนาล็อก (analog) การเรียนรู้ทำให้อุปกรณ์เพิ่มระดับความสามารถให้มากขึ้น โดยวิธีการที่นิยมในการนำมาใช้ในการเรียนรู้ คือ นิวรอนเน็ตเวิร์ค (Neural Network) และ เจนเนติก (Genetic algorithms) ในวิทยานิพนธ์ฉบับนี้จะขอกล่าวเฉพาะในส่วนของนิวรอนเน็ตเวิร์ค

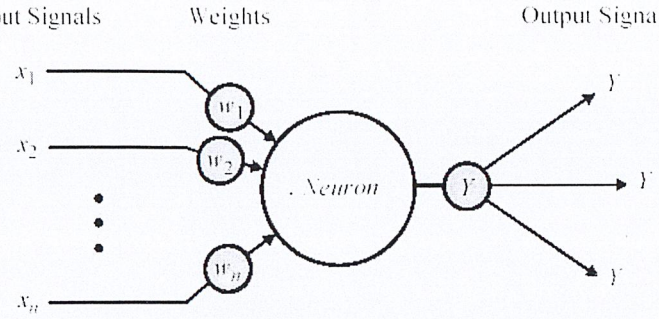
4.2.1 นิวรอนเน็ตเวิร์ค (Neural Network)

นิวรอนเน็ตเวิร์ค เป็นการจำลองการทำงานของสมองมนุษย์มาเป็นรูปแบบการประมวลผลข้อมูล ปกติแล้วในสมองของมนุษย์จะประกอบด้วยเซลล์ประสาทจำนวนมาก ซึ่งทำงานอย่างอิสระ แต่ละเซลล์จะมีการเชื่อมต่อระหว่างกันเรียกว่า สาแนฟ (Synapse) มีส่วนที่เป็นอินพุตของเซลล์ เรียกว่า ดิไรด์ (Dendride) และส่วนที่เป็นเอาท์พุตเรียกว่า เอซอน (Axon) ซึ่งการติดต่อทำได้โดยการสร้างความต่างศักย์ (Voltage) ซึ่งถือว่าเซลล์ประสาทเหล่านี้เป็นหน่วยย่อยที่สุดในการทำงานของสมองซึ่งการทำงานจะเป็นไปในลักษณะที่สัญญาณประสาทถูกส่งต่อกันไปเป็นทอดๆ จากเซลล์หนึ่งไปยังอีกเซลล์หนึ่ง โดยแต่ละเซลล์จะทำหน้าที่ในการคำนวณในส่วนย่อย และมีการเชื่อมต่อถึงกันที่ซับซ้อนเพื่อให้ได้ผลการคำนวณที่ต้องการ



รูปที่ 4-6 แสดงรูปทางชีววิทยาของสมอง

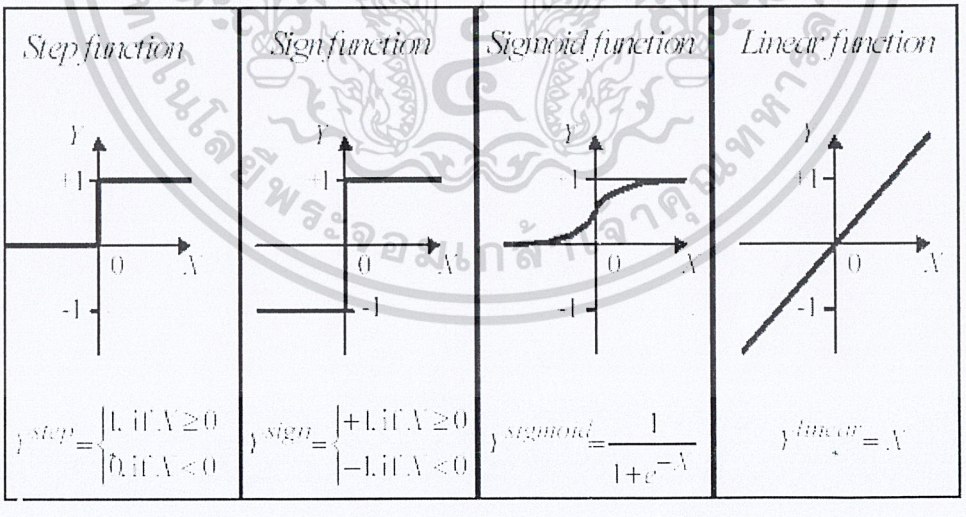
ข้อดี ของการทำงานแบบนี้ คือ เราไม่จำเป็นต้องทราบถึงรูปแบบการคำนวณที่แน่นอน แต่ปล่อยให้เป็นหน้าที่ของกระบวนการเรียนรู้ ที่จะทำหน้าที่สร้างรูปแบบการคำนวณที่ใกล้เคียงกับการคำนวณค่าที่ถูกต้องขึ้นมา โดยอาศัยการปรับค่าที่อยู่ภายในหน่วยย่อย หรือ ค่าที่บริเวณจุดเชื่อมต่อระหว่างหน่วยย่อยเหล่านี้ว่าจะให้อินพุต ใดมีความสำคัญเท่าไร เมื่อนำมาใช้ในทางคอมพิวเตอร์ เราจึงสร้างหน่วยประมวลผลซึ่งมีลักษณะคล้ายเซลล์ประสาท โดยให้แต่ละหน่วยประกอบด้วยส่วนอินพุต ที่มีค่าน้ำหนัก (Weight) อยู่เพื่อระบุค่าความสำคัญของอินพุตนั้นๆ ภายในจะมีการประมวลผลโดยส่วนแรกจะนำค่าอินพุตทั้งหมดมารวมกัน จากนั้นจึงนำไปแทนค่าในแอ็คติเวชัน ฟังก์ชัน (Activation Function) แล้วได้ออกมาเป็นเอาท์พุต



รูปที่ 4-7 แสดงไดอะแกรมของนิวรอนเน็ตเวิร์ค

แอ็คติเวชัน ฟังก์ชัน (Activation Function) นำมาใช้ในการแยกประเภทการจดจำของงานนั้น โดยมีรูปแบบดังนี้

- สะเท็ป (Step function) และ ซายฟังก์ชัน (Sign function) ถูกเรียกว่า ฮาร์ดลิมิตฟังก์ชัน (Hard-limit function) คือ มักจะถูกใช้ในการตัดสินใจของนิวรอนเน็กเวิร์ค ในการแยกประเภท และการจดจำรูปแบบ
- ซิกมอย ฟังก์ชัน (Sigmoid function) ใช้กับอินพุตที่มีค่าระหว่างค่าบวก ถึงอินฟินิตี้ (α)และมีค่าเอาท์พุตอยู่ระหว่าง 0-1 ฟังก์ชันนี้จะถูกใช้ใน แบล็กโพพาเกชัน นิวรอนเน็กเวิร์ค (Back Propagation Network)
- ลีเนียแอ็คติเวชัน (linear activation function) ใช้กลับค่าที่มีเอาท์พุตเท่ากับค่าน้ำหนัก (Weight) ของนิวรอนในส่วนอินพุต มักจะใช้ในจัดการค่าประมาณ

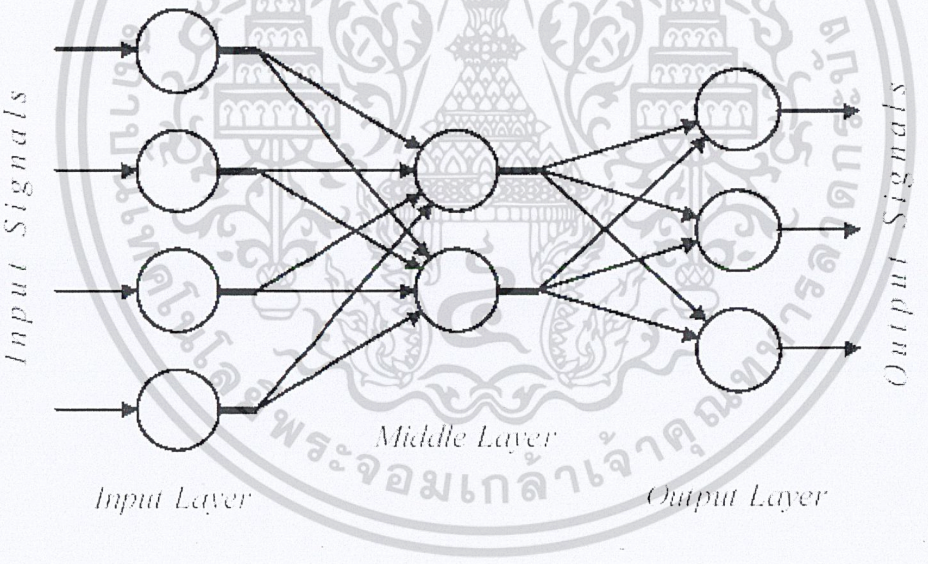


รูปที่ 4-8 แสดงแอ็คติเวชัน ฟังก์ชัน (Activation Function)

4.2.2 รูปแบบการใช้งานนิวรอนแบบต่างๆ

มัลติ เลเยอร์ เพอเซบทอน เน็ตเวิร์ค (Multi-layered perceptron Network) เน็ตเวิร์คแบบนี้ จะมีลักษณะแบ่งออกเป็นชั้น (Layer) ซึ่งประกอบไปด้วย อินพุต ยูนิกต์ (Input units) เอาท์พุต ยูนิกต์ (Output units) ยูนิกต์ และ ไฮเดน ยูนิกต์ (Hidden units) ซึ่งจำนวนชั้นของ ไฮเดน (Hidden units) จะแตกต่างกันไปตามความซับซ้อนของการคำนวณที่ใช้กล่าวคือ ยิ่งการคำนวณซับซ้อนมาก ก็ต้องมีชั้นของ ไฮเดน ยูนิกต์ (Hidden units) มากขึ้น ข้อมูลที่รับเข้ามาทาง อินพุตยูนิกต์ นั้นจะถูกผ่านไปให้แต่ละหน่วยประมวลผลที่อยู่ในชั้นเดียวกันเพื่อทำการประมวลผล และสัญญาณเหล่านี้จะถูกส่งผ่านต่อไปเรื่อยๆ สู่ชั้นถัดไป ซึ่งจะมีการให้ค่าความสำคัญ (Weight) ของอินพุตที่รับเข้ามาจากหน่วยต่างๆ แตกต่างกันไป และสัญญาณจะถูกส่งต่อระหว่างชั้นไปเรื่อยๆ จนกระทั่งไปถึงเอาท์พุตยูนิกต์ซึ่งจะส่งผลลัพธ์ที่ได้ออกมาโดยค่าน้ำหนักของอินพุตต่างๆ นั้น จะถูกปรับโดยวิธีการเรียนรู้ (Learning) โดยวิธีการเรียนรู้ ก็คือ การนำข้อมูลที่ทราบผลลัพธ์ แล้วมาใช้เป็นอินพุต และเปรียบเทียบค่าเอาท์พุตที่ได้กับผลลัพธ์หาค่าความผิดพลาด

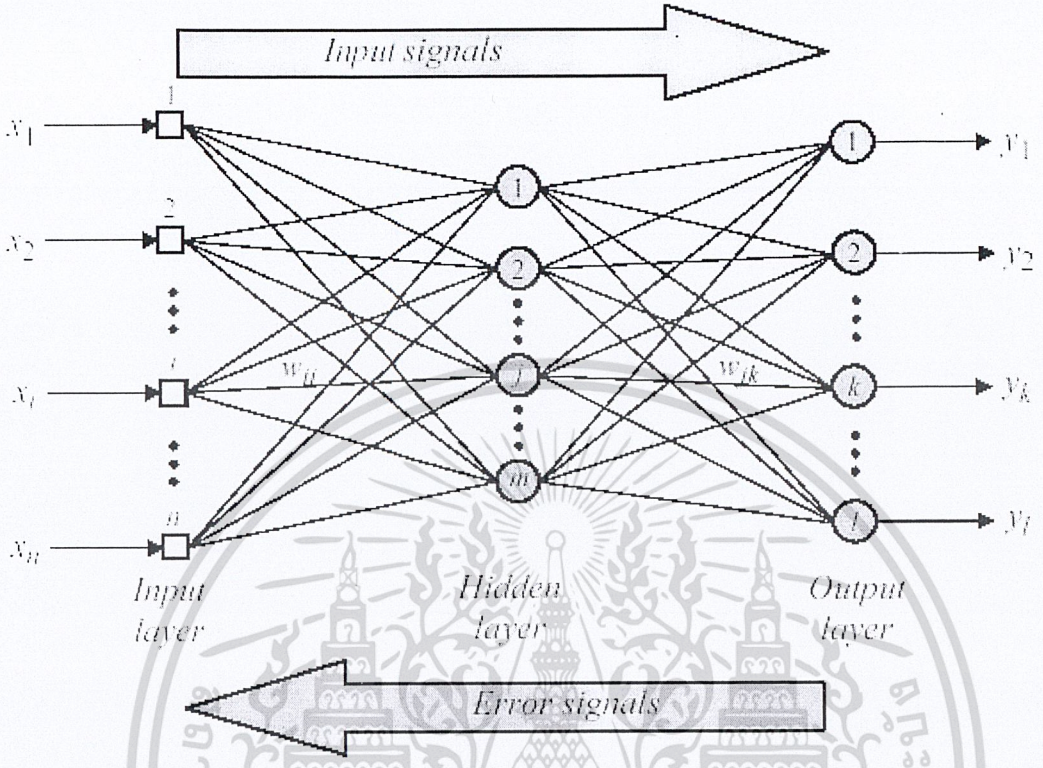
จากนั้นจึงนำค่าความผิดพลาดที่ได้มาทำการปรับน้ำหนัก (Weight) ภายใน Neural Network ซึ่งวิธีการปรับค่านั้นขึ้นอยู่กับวิธีการเรียนรู้ที่เลือกใช้ ซึ่งวิธีการเรียนรู้นั้นมีหลายรูปแบบขึ้นกับการออกแบบโครงสร้างภายในของ นิวรอน เน็ตเวิร์คและประเภทของงานที่นำไปใช้



รูปที่ 4-9 แสดงการทำงานหลายเลเยอร์

แบ็กโพพาเกชัน (Back-propagation neural network) เป็นวิธีการเรียนรู้รูปแบบหนึ่ง ซึ่งเป็นที่นิยมใช้ โดยอาศัยหลักการกระจายความรับผิดชอบต่อความผิดพลาดออกไปยังหน่วยย่อยทุกหน่วยใน เน็ตเวิร์ค เท่าๆ กัน โดยจะกระจายความผิดพลาดลงไปในแต่ละระดับชั้น ซึ่งจะสังเกตได้ว่าใน มัลติเลเยอร์ เพอเซปตอน (Multi-layered perceptron Network) นั้น ค่าที่ได้จากแต่ละระดับชั้นจะส่งผลกระทบถึง

ระดับชั้นถัดไป ในลักษณะของการยกกำลัง ดังนั้นการกระจายความผิดพลาดจากระดับชั้นหลังไปสู่ระดับชั้นต้นๆ นั้นจำเป็นต้องใช้หลักการอินทิเกรต เพื่อให้ได้อัตราความผิดพลาดที่เกิดขึ้นในระดับชั้นนั้น



รูปที่ 4-10 แสดงการทำงานของแบล็กโพพาเกชัน **(Back-propagation neural network)**

วิธีการเรียนรู้ของแบ็กโพพาเกชัน นิวรอนเน็ตเวิร์ค (Back-propagation neural network)

1. สร้างค่าน้ำหนัก (Weight) และค่ากำหนดขอบเขต (Threshold) โดยค่าทั้งสองจะเป็นค่าที่สุ่มขึ้นมาระหว่าง

$$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i} \right)$$

โดยค่า F คือจำนวนอินพุตของนิวรอน I ในเน็ตเวิร์ค

2. ประมวลผล

คำนวณค่าเอาท์พุตจริงของนิวรอนในไฮเดนเลเยอร์

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) \cdot w_{ij}(p) - \theta_j \right]$$

โดยที่ n คือ จำนวนอินพุต j ในไฮเดนเลเยอร์ และ sigmoid

คำนวณค่าเอาท์พุตจริงของนิวรอนในเอาท์พุตเลเยอร์

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p)\cdot w_{jk}(p) - \theta_k\right]$$

โดยที่ m คือ จำนวนอินพุตของนิวรอน k ใน เอาท์พุตเลเยอร์

3. เรียนรู้น้ำหนัก

ปรับค่าน้ำหนักใน แบ็กโพพาเกชัน เน็ตเวิร์ค ค่าผิดพลาดจะเกี่ยวข้องกับเอาท์พุตนิวรอน คำนวณค่าผิดพลาดแกเดียน (error gradient) จากนิวรอนในเอาท์พุตเลเยอร์

$$\delta_k(p) = y_k(p)\cdot[1 - y_k(p)]\cdot e_k(p)$$

โดย

$$e_k(p) = y_{d,k}(p) - y_k(p)$$

คำนวณค่าผิดพลาดแกเดียน (error gradient) จากนิวรอนในไฮเดนเลเยอร์

$$\delta_j(p) = y_j(p)\cdot[1 - y_j(p)]\cdot\sum_{k=1}^{l}\delta_k(p)\,w_{jk}(p)$$

คำนวณหาค่าน้ำหนักจริง

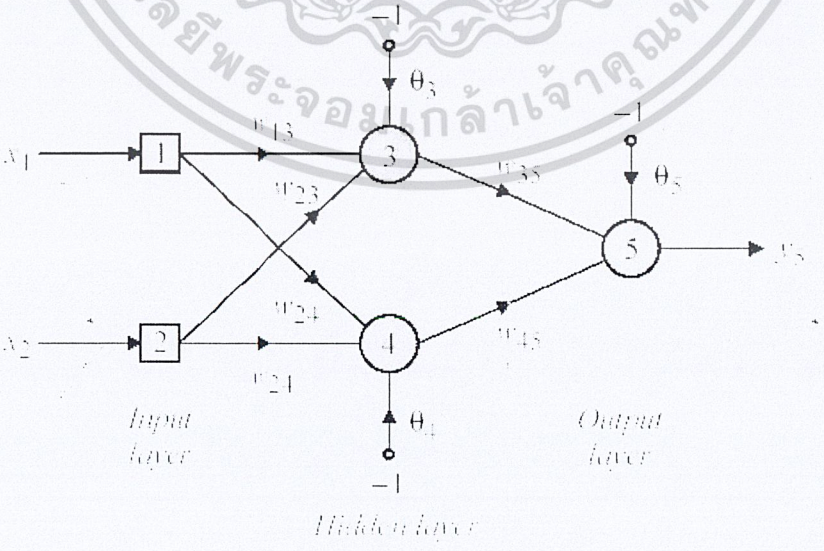
$$\Delta w_{ij}(p) = \alpha\cdot x_i(p)\cdot\delta_j(p)$$

ปรับค่าน้ำหนักที่ไฮเดน นิวรอน

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

4. ทำซ้ำ

เพิ่มค่า p ขึ้นอีกหนึ่ง และกลับไปเริ่มทำลำดับที่ 2 ใหม่ ทำจนกระทั่งค่าผิดพลาด พอใจ



รูปที่ 4.11 ตัวอย่างการทำงาน โดยใช้ 3 เลเยอร์

อัตราความเร็วของการเรียนรู้ขึ้นอยู่กับตัวแปร α ซึ่งหากมีค่ามากจะทำให้ค่าภายในนิวรอนเน็ตเวิร์ค เปลี่ยนแปลงไปอย่างรวดเร็วระหว่างการเรียนรู้ แต่ก็มีข้อเสียคือจะทำให้ค่าที่ได้ไม่ละเอียดซึ่งทำให้ผลการทำงานของนิวรอนเน็ตเวิร์คผิดพลาด ไม่เที่ยงตรงได้ ในกรณีที่เราใช้ค่า α ไม่มากนัก อาจทำให้ในการทำแบ็กโพพาเกชันพี เพียงรอบเดียวไม่สามารถปรับค่าให้ถูกต้องได้ ดังนั้นส่วนใหญ่ในการเรียนรู้แต่ละครั้งนั้น จะเป็นการทำ Back Propagation หลายๆ รอบกับข้อมูลกลุ่มเดิม โดยยิ่งค่า α ต่ำจำนวนรอบในการทำงานก็ยิ่งต้องเพิ่มมากขึ้น นอกจากนี้ปริมาณและการกระจายตัวของข้อมูลตัวอย่างที่ใช้ในการเรียนรู้ (Training Set) นั้นก็เป็นสิ่งสำคัญที่มีผลต่อการเรียนรู้มาก ข้อมูลที่นำมาใช้เป็นตัวอย่างในการเรียนรู้นั้นต้องเป็นข้อมูลที่เป็นมาตรฐานตามปกติ ไม่ใช่ข้อมูลที่มีรูปแบบโครงสร้างที่ผิดไปจากข้อมูลปกติ นอกจากนี้ต้องมีจำนวนมากพอที่ จะทำให้มีรูปแบบที่หลากหลายพอที่จะครอบคลุมถึงลักษณะปกติของข้อมูลที่จะทำการวิเคราะห์ได้หมด

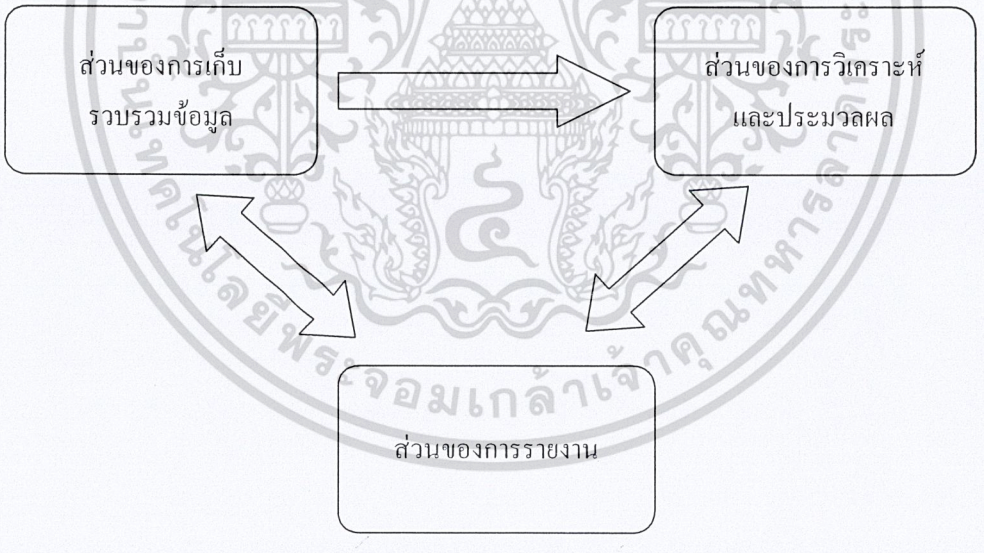
บทที่ 5
การคำนวณ การสร้างและการออกแบบ

5.1 ระบบโดยรวม

ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์ที่ออกแบบแบ่งออกเป็น 3 ส่วน คือ

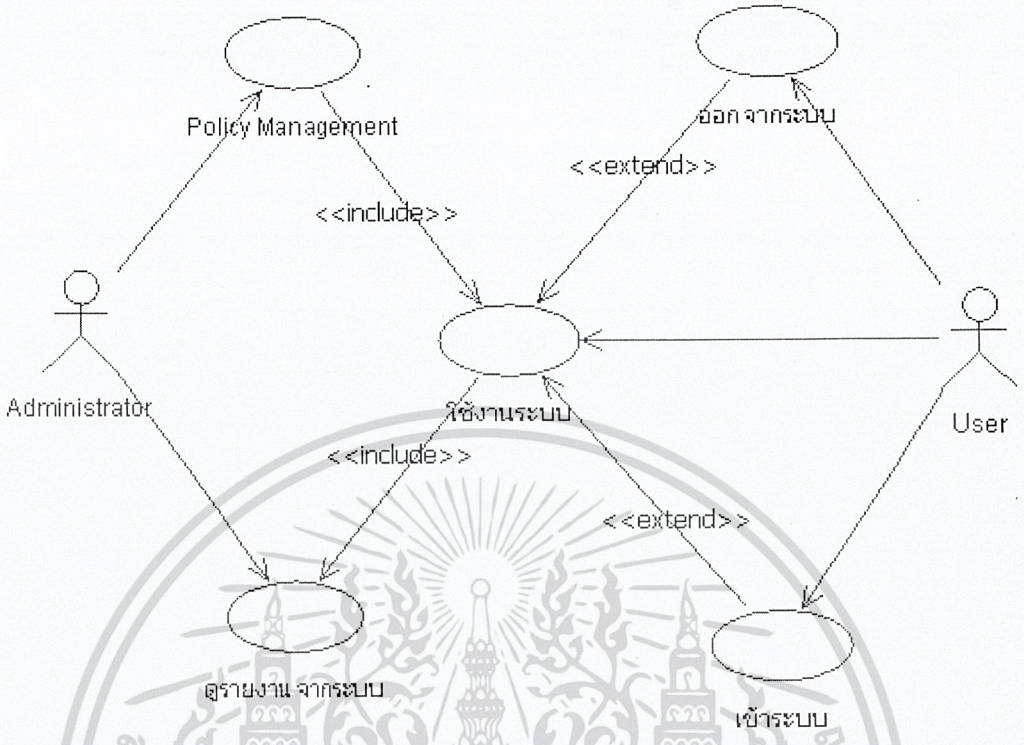
1) **ส่วนของการเก็บรวบรวมข้อมูล** จะมีการเก็บข้อมูลพฤติกรรมการใช้งานโพรเซสต่างๆ ของผู้ใช้งานระบบแต่ละคนให้เป็นรูปแบบที่พร้อมสำหรับการวิเคราะห์
2) **ส่วนการวิเคราะห์และประมวนผล** จะมีการนำฟัซซี่ลอจิกและ นิวรอนเน็ตเวิร์คมาใช้ในการวิเคราะห์และประมวลผล
3) **ส่วนของการรายงานผลไปยังผู้ดูแลระบบ** เมื่อพบพฤติกรรมที่ผิดปกติ จะรายงานโดยการส่งเมล์ไปยังผู้ดูแลระบบและเก็บลงล็อกไฟล์ของโปรแกรม

ในส่วนของการเก็บรวบรวมข้อมูล จะทำงานโดยใช้การเขียนเชลล์สคริปต์ในการดักจับพฤติกรรมและแปลงข้อมูลให้อยู่ในรูปแบบที่พร้อมสำหรับการวิเคราะห์ ส่วนของการวิเคราะห์และประมวลผล จะมีการเขียนโดยภาษา C++



รูปที่ 5-1 แสดงรูปแบบการทำงานของระบบตรวจจับผู้บุกรุก

Use Case Diagrams

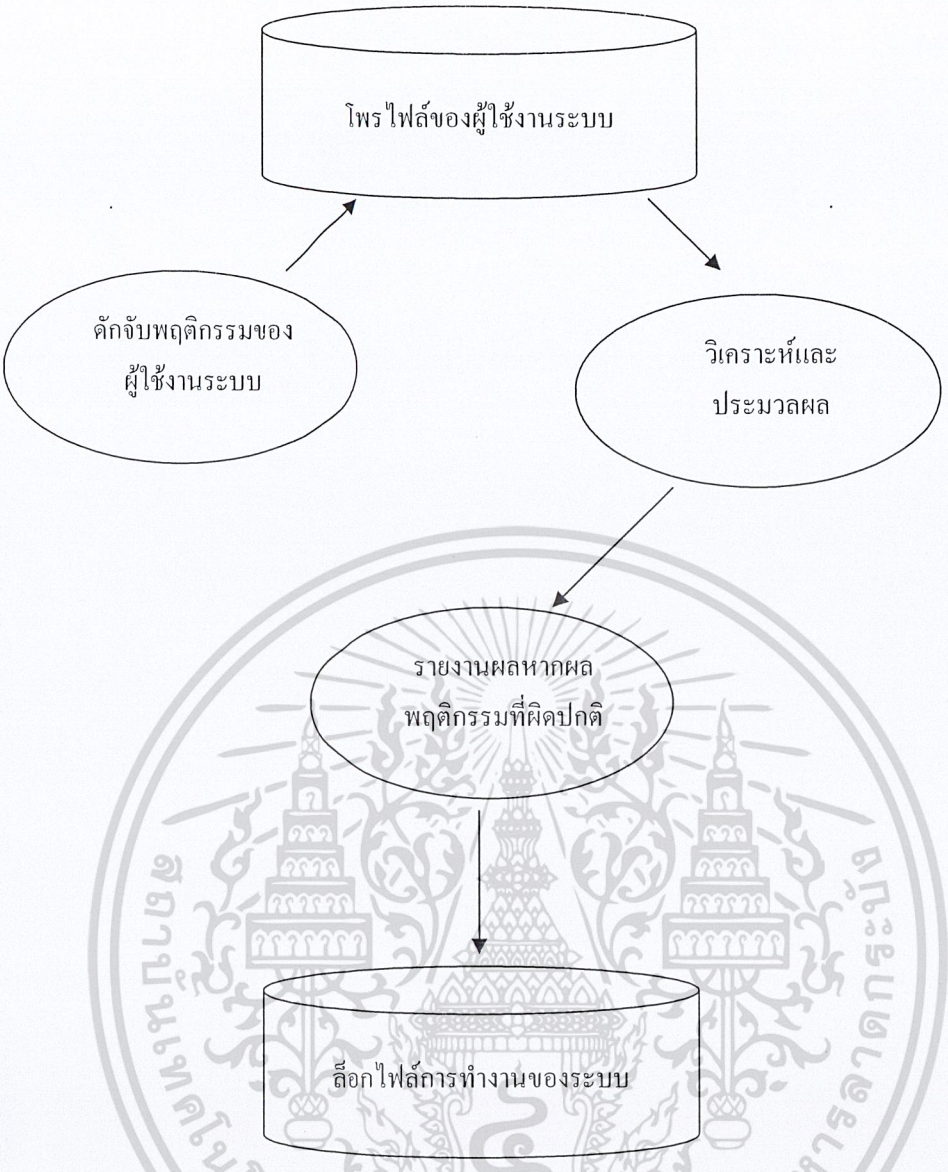


รูปที่ 5-2 ยูสเคสไดอะแกรม (Use Case Diagrams) แสดงภาพโดยรวมของระบบ

จากรูปผู้ใช้งานระบบสามารถล็อกอินเข้ามาใช้งานระบบโดยระบบจะมีการเก็บโพรไฟล์การใช้งานไว้ ส่วนผู้ดูแลระบบสามารถที่จะเข้ามาดูพฤติกรรมของผู้ใช้แต่ละคนได้ตลอดเวลาโดยดูจากล็อกไฟล์ของระบบ นอกจากนี้ยังสามารถปรับแต่งระบบได้โดยการระบบชื่อคำสั่งที่ต้องการตรวจจับหรือน้ำหนักค่าฟัซซี่ (Fuzzy) ของแต่ละคำสั่ง

5.2 ลักษณะการทำงานของระบบตรวจจับผู้บุกรุกที่มีการนำปัญญาประดิษฐ์มาใช้งาน

เมื่อระบบตรวจจับผู้บุกรุกเริ่มทำงาน ระบบจะทำการรวบรวมข้อมูลการทำงานของแต่ละผู้ใช้งานระบบแต่ละคน เก็บไว้ในโพรไฟล์ของผู้ใช้งานระบบแต่ละคน เพื่อนำไปใช้ในการวิเคราะห์ และประมวลผลพฤติกรรมของผู้ใช้แต่ละคน หากพบความผิดปกติ ระบบจะทำการรายงานผลการตรวจสอบและส่งข้อมูลภายในล็อกไฟล์ ไปยังผู้ดูแลระบบ



รูปที่ 5-3 แสดงลักษณะการทำงานของระบบตรวจจับผู้บุกรุก

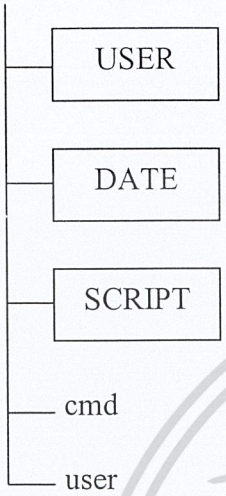
5.3 ขั้นตอนการทำงานของระบบตรวจจับผู้บุกรุกที่มีการนำเอาปัญญาประดิษฐ์มาใช้งาน

ลำดับแรกในการทำงานระบบจำเป็นต้องมีการเรียนรู้พฤติกรรมของผู้ใช้งานระบบเป็นเวลาอย่างน้อย 2 สัปดาห์ เพื่อสามารถนำพฤติกรรมดังกล่าวมาเปรียบเทียบในการใช้งาน ณ เวลาปัจจุบันได้ หลังจากนั้นระบบจะมีการวิเคราะห์และประเมินผลเป็นเวลาทุกวัน รวมถึงมีการเรียนรู้พฤติกรรมเพิ่มเติมของแต่ละผู้ใช้งานระบบไปพร้อมกันด้วย หากพบว่ามีพฤติกรรมที่ผิดปกติระบบจะทำการรายงานผลไปยังผู้ดูแลระบบพร้อมทั้งบันทึกลงล็อกไฟล์

รูปที่ 5-4 แสดงขั้นตอนการทำงานของระบบตรวจจับผู้

5.4 การเก็บรวบรวมข้อมูลการทำงานของผู้ใช้งานแต่ละคน

ส่วนการเก็บรวบรวมข้อมูลการทำงานของผู้ใช้ มีหน้าที่ในการเก็บรวบรวมข้อมูลความถี่ในการใช้งานโพรเซสต่างๆ ของระบบแยกเป็นโพรไฟล์ของผู้ใช้แต่ละคนโดยจะเก็บข้อมูลเป็นรายวันเพื่อให้ได้ข้อมูลมากเพียงพอต่อการวิเคราะห์โดยใช้นิวรอนเน็ตเวิร์คต่อไป โครงสร้างการเก็บข้อมูลดังนี้



เก็บโพรไฟล์ของผู้ใช้แต่ละคน โดยเป็นขอมูลที่พร้อมสำหรับการวิเคราะห์

เก็บไฟล์ที่ระบุข้อมูลการใช้งานโพรเซสต่างแยกแต่ละวัน

เก็บไฟล์สคริปต์ที่ใช้ในการเก็บและแปลงข้อมูล

เก็บชื่อคำสั่งหรือชื่อโพรเซสที่ต้องการวิเคราะห์ผล

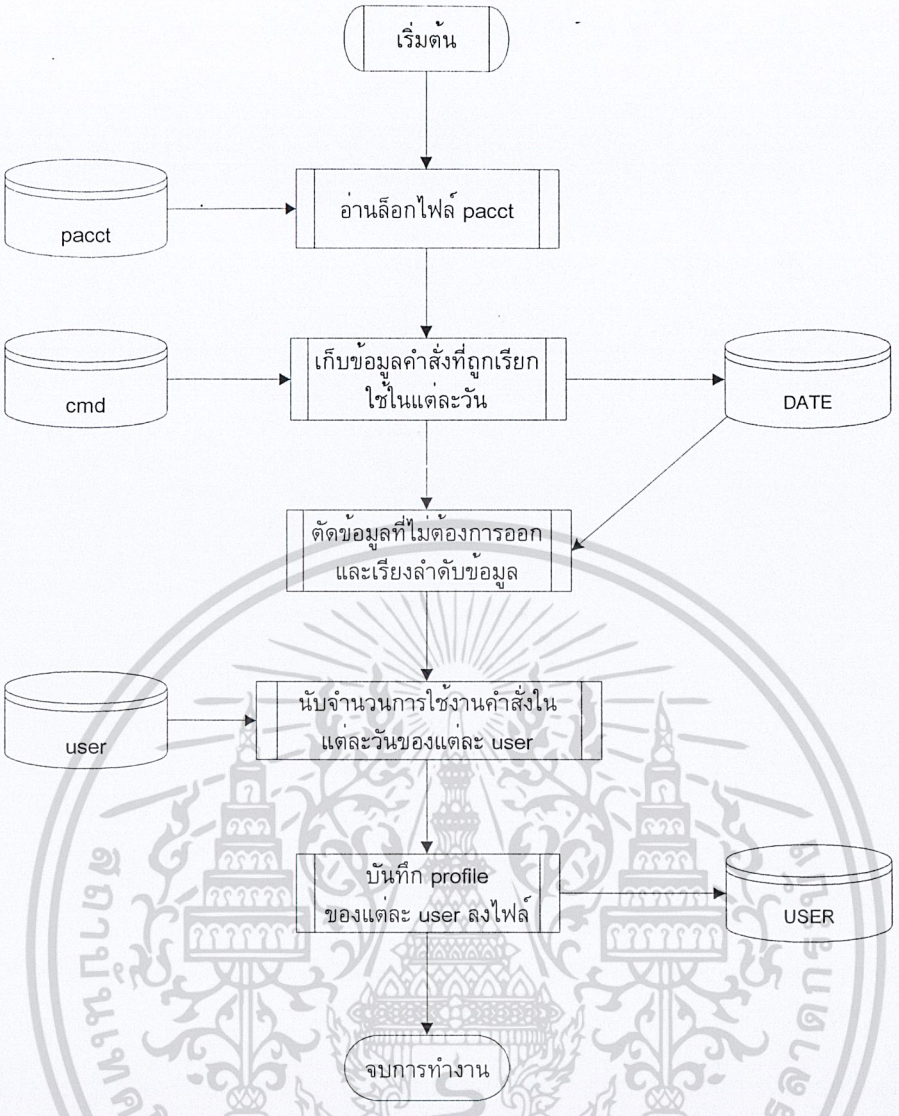
เก็บชื่อผู้ใช้งานระบบที่ต้องการวิเคราะห์ ซึ่งโดยปรกติแล้วจะเป็นผู้ใช้ที่ใช้งานเป็นประจำ

การทำงานของสคริปต์ทั้งหมดอ้างอิงกับไฟล์ user และ cmd เพื่อให้สามารถแก้ไขไฟล์ดังกล่าวได้ในภายหลัง โครงสร้างการเก็บข้อมูลในไฟล์ cmd และ user จะเก็บเป็น 2 คอลัมน์คือ หมายเลข(ID) และชื่อคำสั่ง โดยเหตุที่ต้องมีการเก็บเป็น id ก็เพราะว่าสามารถอ้างอิงและปรับปรุงแก้ไขได้ง่าย

ID	User name
1	s2010176
2	s2010169
3	s2010159

ID	Command
1	ls
2	vi
3	pico

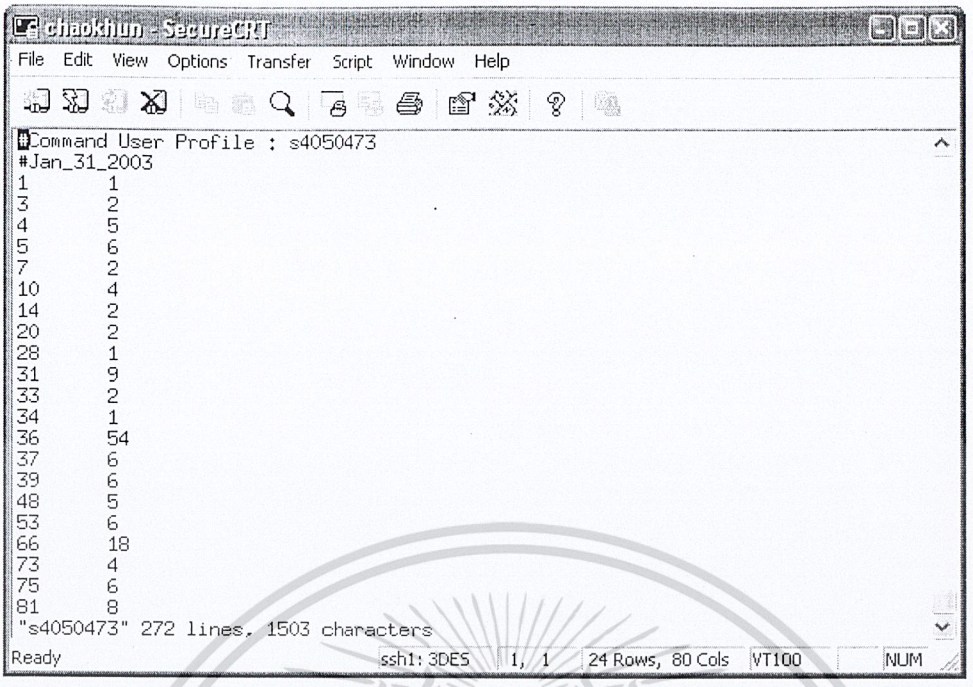
*ตารางที่ 5-1 แสดงโครงสร้างการเก็บข้อมูลในไฟล์ **cmd** และ **user***



รูปที่ 5-5 แสดงขั้นตอนการทำงานของการรวบรวมข้อมูลผู้ใช้งาน

ขั้นตอนการทำงานของการเก็บรวบรวมข้อมูล มีดังนี้

1. กำหนดชื่อผู้ใช้งานที่อยู่ในระบบและใช้งานบ่อยครั้งลงในไฟล์ user
2. กำหนดชื่อคำสั่งหรือชื่อโพรเซสที่ต้องการตรวจจับลงในไฟล์ cmd
3. เรียกใช้งานคำสั่ง lastcomm เพื่อดูข้อมูลในไฟล์ pacct ที่ระบบเก็บไว้ที่ /var/account จะได้ไฟล์ข้อมูลการใช้งานคำสั่งต่างๆภายในวันนั้น ในโฟล์เดอร์ DATE โดยมีเฉพาะคำสั่งที่ระบบในไฟล์ cmd เท่านั้น
4. นำข้อมูลการใช้งานคำสั่งรายวันมาตัดข้อมูลที่ไม่ต้องการออก จากนั้นเรียงลำดับและนับจำนวนการเรียกใช้งานโพรเซสพร้อมทั้งแยกข้อมูลเก็บลงไฟล์ของผู้ใช้แต่ละคนในโฟล์เดอร์ USER ซึ่งชื่อไฟล์จะเป็นชื่อผู้ใช้ที่ระบบในไฟล์ user โดยมีโครงสร้างภายในไฟล์คือคอลัมน์ที่1 เป็นหมายเลขของคำสั่งซึ่งอ้างอิงจากไฟล์ cmd และคอลัมน์ที่2เป็นจำนวนที่ถูกเรียกใช้งาน ดังรูป



รูปที่ 5-6 แสดงข้อมูลโพรไฟล์การใช้งานคำสั่งของ User แต่ละคน

ซึ่งข้อมูลในไฟล์โพรไฟล์ของผู้ใช้นี้จะเก็บไว้เพียง 30 วันย้อนหลังเท่านั้น ที่เหลือจะตัดแล้ว zip เก็บไว้

5.5 การติดตั้งระบบ

ในส่วนนี้ต้องมีการกำหนดค่าเริ่มต้นหรือปรับแต่งค่าต่างๆเพื่อให้ระบบตรวจจับได้แม่นยำมากที่สุด มีสิ่งที่ต้องกำหนดดังนี้

- การตั้งค่าคำสั่งที่ต้องการตรวจจับในไฟล์ “cmd” ซึ่งควรจะเป็นคำสั่งที่มีการเรียกใช้งานเป็นประจำ
- ชื่อผู้ใช้ที่ต้องการวิเคราะห์ในไปในไฟล์ “user” ซึ่งควรจะเป็นชื่อผู้ใช้ที่มีการเข้ามาใช้งานบ่อยครั้ง
- กำหนดช่วงของการเก็บข้อมูลมาทำการเรียนรู้โดยผ่านนิวรอนเน็ตเวิร์ค
- ตั้งค่าเวลาที่ต้องการให้โปรแกรมทำการวิเคราะห์ระบบและเรียนรู้โดยใช้คำสั่ง crontab โดยต้องมีการทดลองเพื่อหาค่าที่เหมาะสมที่สุด คือใช้เวลาในการเรียนรู้น้อยและได้ผลที่ถูกต้องแม่นยำ

5.6 การวิเคราะห์และประมวลผล

ในส่วนการวิเคราะห์และประมวลผลนั้นระบบตรวจจับผู้บุกรุกได้มีการนำปัญญาประดิษฐ์มาใช้งานร่วมด้วย กล่าวคือมีการนำฟัชชี่ลอจิกมาใช้ในการแบ่งระดับการความถี่ในการใช้งานและนิวรอนเน็ตเวิร์คมาใช้เพื่อให้ระบบมีความยืดหยุ่นและสามารถเรียนรู้ได้ โดยมีหลักการดังนี้

การทำงานในส่วนของการเก็บรวบรวมข้อมูลการทำงานของผู้ใช้งานแต่ละคนจะส่งผลมาให้กลับส่วนของการวิเคราะห์และประมวลผลในรูปของโพไฟล์ของแต่ละคน โดยมีตัวอย่างการทำงานดังนี้

กำหนดให้มี User 5 คน มี command ที่ใช้ 5 คำสั่ง

คำสั่ง	User A	User B	User C	User D	User E
ls	50	9	20	30	50
cat	0	30	25	19	20
vi	10	26	0	40	20
man	2	0	16	10	25
pico	9	80	0	30	95

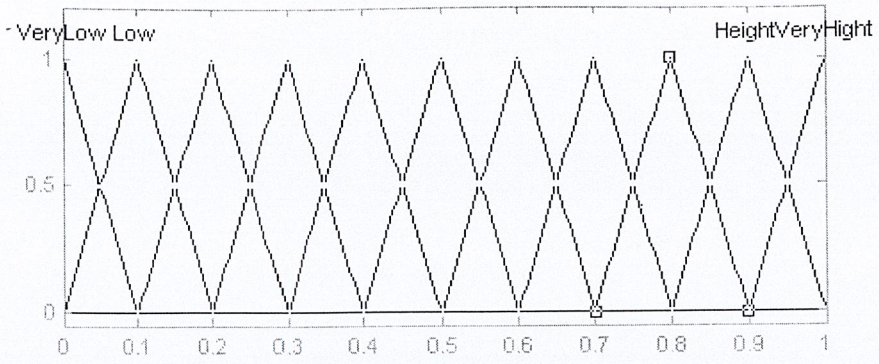
ตารางที่ 5-2 แสดงตัวอย่างการเก็บรวบรวมพฤติกรรมของการใช้ของผู้ใช้งานแต่ละคน

1) นำค่าที่ได้จากส่วนของการเก็บรวบรวมข้อมูลเข้ามาประมวลผลในฟัซซี่ ลอจิก

โดยในการเก็บค่าการใช้งานคำสั่งแต่ละคำสั่งนั้นเราจะทำการเก็บเป็นเวลาอย่างน้อย 2 สัปดาห์ จากนั้นเราจะนำค่าจำนวนของการใช้งานแต่ละคำสั่ง มาเข้าฟัซซี่ ลอจิกเพื่อจะหาค่าระดับการใช้งานมากน้อยเพียงใดในแต่ละคำสั่งก็จะมีค่า ช่วงของการคิดค่ามากน้อยต่างกัน เช่นในส่วนของ command ls จะมีค่าในช่วง 0-50 แต่ command vi จะมีค่าในช่วง 0-30 สมมตินำ User A เข้า ฟัซซี่ ลอจิก ค่าที่ได้ของ ls จะมีค่าเป็น 1.0 และ vi จะมีค่าเป็น 0.2

Linguistic variable : Intrusion Level		
Linguistic value	**Notation**	**Numerical range (normallsed)**
Low	L	[0, 0.1]
Quite Low	QL	[0.1, 0.15, 0.2]
. .	. .	. .
Medium	M	[0.8, 0.85, 0.9]
High	H	[0.9,1]

ตารางที่ 5-3 แสดงการแบ่งค่า Linguistic variable



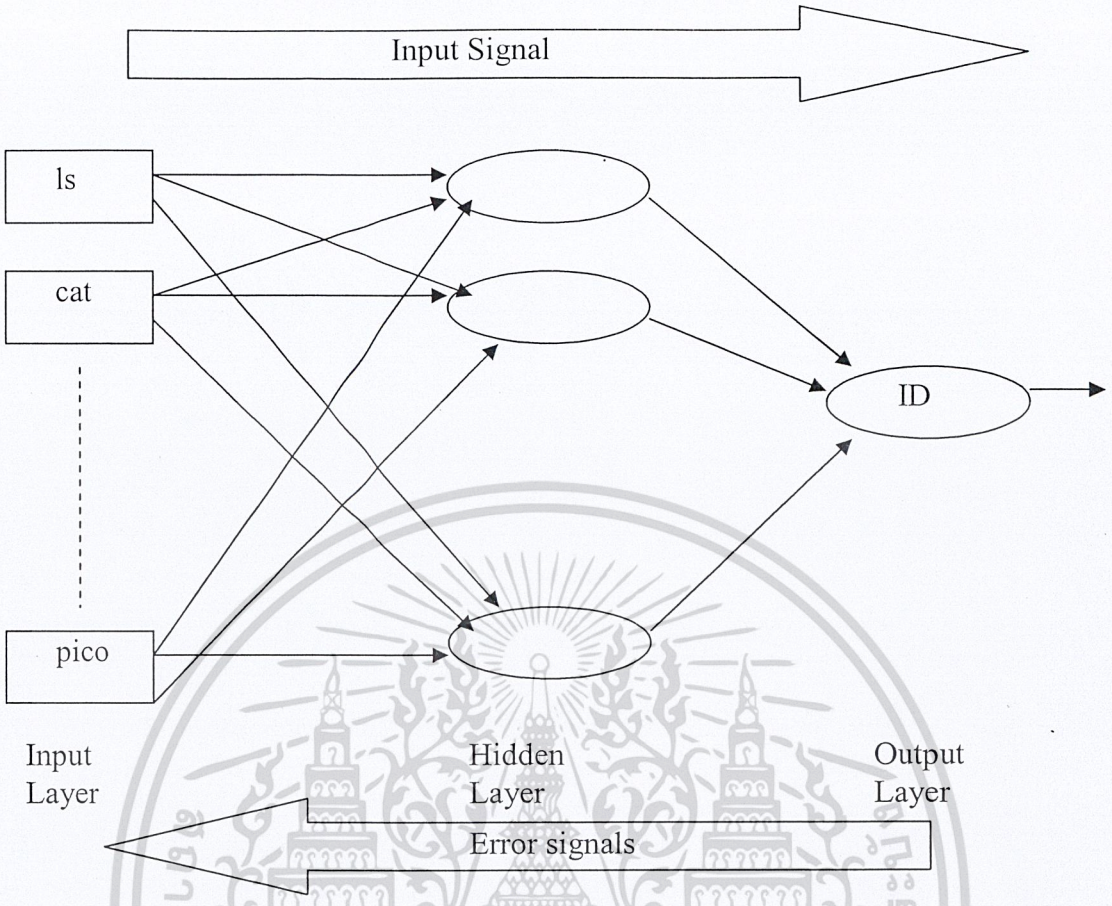
รูปที่ 5 -7 แสดงฟัซซี่เซต

คำสั่ง	User A
ls	1.0
cat	0
vi	0.2
man	0.08
cd	0.105

ตารางที่ 5-4 แสดงผลที่ได้เมื่อผ่านฟัซซี่ ลอจิก

2) นำค่าที่ได้จากส่วนของฟัซซี่ลอจิก มาเข้าสู่กระบวนการการเรียนรู้

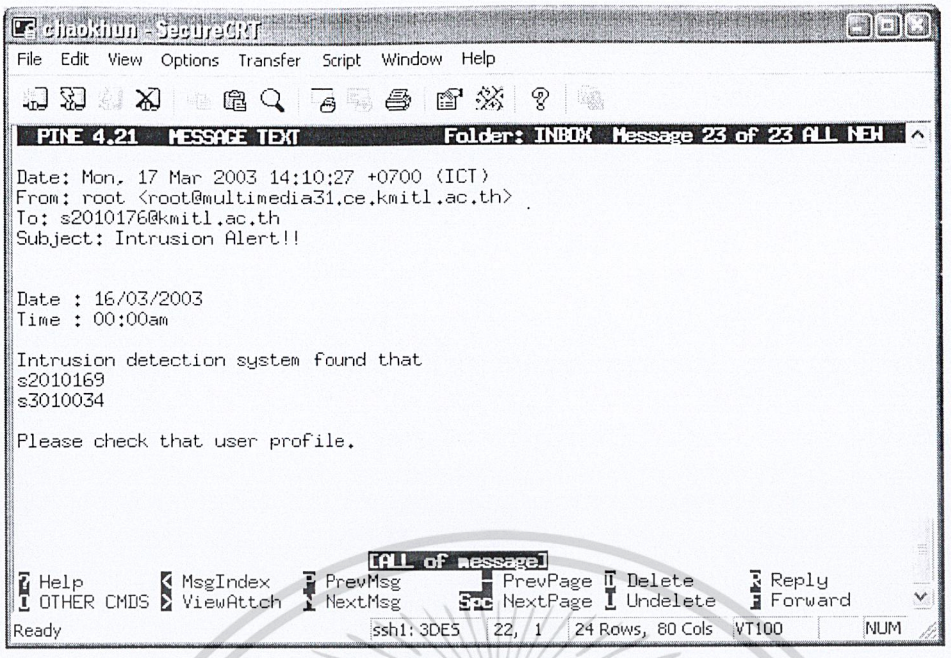
นิวรอนที่นำมาใช้ในวิทยานิพนธ์ฉบับนี้คือ แบกโพพาเกชันนิวรอน (Back Propagation Neural) จะมีการเรียนรู้และเพิ่มน้ำหนักในแต่ละเส้นทาง ในการทำงานเพื่อหาค่าน้ำหนัก และของเขต เพื่อใช้ในการประมวลผลเมื่อนำระบบไปใช้งานจริง คือจะนำค่าทั้งสองไปใช้ในการวิเคราะห์ว่าการใช้งานของผู้ใช้งานลักษณะนี้เป็นการใช้งานของหมายเลขรหัสประจำตัวอะไร หากการทำงานที่เข้ามากับในส่วนของข้อมูลที่เก็บบันทึกพฤติกรรมไว้ไม่ตรงกัน แสดงว่าผู้ใช้งานผู้นั้นมีอัตราเสี่ยงต่อการเป็นผู้บุกรุก



รูปที่ 5-8 แสดงรูปแบบการทำงานของนิวรอนเน็ตเวิร์คของระบบตรวจจับผู้บุกรุก

5.7 ส่วนแจ้งเตือนและรายงานผลการตรวจสอบ

เมื่อระบบทำการตรวจสอบแล้วพบความผิดปกติระบบจะทำการแจ้งไปยังผู้ดูแลระบบโดยการส่งอีเมล์ไปยังผู้ดูแลระบบ รวมถึงการเก็บข้อมูลการวิเคราะห์ลงล็อกไฟล์ เพื่อให้ผู้ดูแลระบบสามารถทำการตรวจสอบการทำงานได้โดยง่าย



รูปที่ 5-9 แสดงผลการแจ้งเตือนผู้ดูแลระบบทางอีเมล์

บทที่ 6

ผลการทดลอง

หลังจากได้ออกแบบตามแนวคิด ที่ได้กำหนดไว้ในบทที่ 5 แล้ว ก็ได้ทดลองปรับเปลี่ยนค่าพารามิเตอร์ต่างๆ ที่มีผลต่อการทดลอง และส่วนแสดงผลการทำงานของโปรแกรม

6.1 การวัดประสิทธิภาพของระบบ

การวัดประสิทธิภาพ มีการวัด 2 แบบ คือ

6.1.1 เวลา

เวลา คำนวณจากจำนวนครั้งที่ใช้ในการเรียนรู้ซึ่งถ้าจำนวนครั้งที่ใช้ในการเรียนรู้มาก เวลาจะมาก และถ้าจำนวนครั้งที่ใช้ในการเรียนรู้น้อยจะถือว่าประสิทธิภาพมาก

6.1.2 ความถูกต้อง

ความถูกต้อง เป็นการวัดความถูกต้องของระบบโดยวัดทั้งแบบที่เกิดการบุกรุกแล้วตรวจพบว่าเกิดการบุกรุกได้กับแบบที่ไม่เกิดการบุกรุกแล้วตรวจพบว่าไม่เกิดการบุกรุก รวมถึงมีการตรวจสอบว่ามีมีการดักจับผิดเป็นจำนวนเท่าไร โดยแบ่งเป็น 2 ประเภทคือ มีการบุกรุกแต่ระบบไม่สามารถดักจับได้ (False positive) และ ไม่มีการบุกรุกแต่ระบบมีการตรวจสอบว่าเป็นการบุกรุก (False negative) โดยได้ทำการทดลองจากข้อมูลจริงที่ทราบแล้ววัดผลความถูกต้องออกมาเป็นเปอร์เซ็นต์

ตัวอย่าง การทดลองวัดประสิทธิภาพด้านความถูกต้อง

- จะนำค่า **False Negative** (ตรวจพบว่าเป็นการโจมตีจากข้อมูลทั่วไป) และ **False Positive** (ตรวจไม่พบการโจมตี จากผู้ที่เข้ามาโจมตี) มาใช้ในการวัดประสิทธิภาพ
- กำหนดให้มีผู้เข้ามาใช้งานทั้งหมดในระบบเป็นจำนวน 30 คน มีการดักจับโพรเซสที่เข้ามาใช้งานเพื่อนำไปใช้ในการเรียนรู้ ทั้งหมด 85 คำสั่ง เวลาที่ใช้ในการเรียนรู้ 14 วัน

ผลการทดลองที่ 1 จากการสังเกตการทำงานของผู้ใช้งาน A มีการใช้งานดังนี้

Jan_30_2003		Jan_29_2003		Jan_25_2003		Jan_24_2003		Jan_20_2003		Jan_17_2003	
cat	2	cat	1	cat	1	as	1	cat	3	cat	2
ksh	8	ksh	1	ksh	3	cat	3	clear	3	finger	1
ls	2	ls	1	ls	3	cc1	1	finger	5	ksh	6
mail	2	mail	1	mail	1	cpp	1	ksh	8	ls	11
man	3	quota	1	pico	1	finger	1	ls	9	mail	2
quota	2	stty	2	quota	1	gcc	1	mail	3	pico	2
stty	4	who	1	stty	2	ksh	9	mesg	2	quota	2
				who	1	ld	1	pico	2	stty	6
						lex	2	pine	1	talk	4
						ls	8	quota	3	tty	2
						mail	3	stty	6	who	4
						pico	4	talk	1		
						quota	3	tput	3		
						stty	6	who	9		
						who	1	write	1		
						write	1				

Jan_16_2003		Jan_13_2003		Jan_11_2003		Jan_10_2003		Jan_09_2003		Jan_08_2003	
a.out	1	a.out	2	as	16	cat	2	cat	1	cat	2
acomp	2	acomp	17	cat	9	ksh	2	ksh	1	cp	1
cat	1	as	1	cc1	19	mail	2	mail	1	date	1
cc	2	cat	9	col	3	pico	2	quota	1	ksh	15
ksh	2	cc	20	comm	6	quota	2	stty	3	ls	5
ld	2	cc1	1	cpp	19	stty	6	tty	1	mail	2
ls	3	clear	4	deroff	6	tty	2			mkdir	2
mail	1	cpp	2	finger	5					quota	2
pico	1	finger	6	gcc	25					rmdir	1
quota	1	gcc	2	ksh	42					stty	6
stty	3	ksh	28	ld	16					tty	2
tty	1	ld	11	ls	20					whoami	1
		lex	5	mail	3						
		ls	68	man	6						
		mail	7	mesg	2						
		pico	23	more	4						
		quota	7	mv	3						
		rm	11	nroff	3						
		stty	21	pico	12						
		tput	4	pine	1						
		tty	7	quota	3						
		uname	2	rm	6						
		vi	2	sed	24						
		who	6	sendmail	4						
		write	2	sh	16						
				sort	6						
				spell	24						
				spellpro	12						
				stty	9						
				talk	7						
				tbl	2						
				tee	6						
				tr	6						
				tty	3						

ตารางที่ 6-1 แสดงโพรไฟล์ของผู้ใช้ชื่อ A

พฤติกรรมที่นำมาตรวจสอบ มีดังนี้

#Jan_31_2003	
comp	1
as	7
cat	3
cc	1
cc1	8
clear	1
cpp	8
gcc	9
ksh	11
ld	8
lex	7
ls	32
mail	3
pico	13
quota	3
rm	9
rmdir	2
stty	6
talk	3
top	4
tput	1
who	4

ทดลองนำพฤติกรรมในวันที่ตรวจจับซึ่งจะเห็นได้ว่ามีความใกล้เคียงกับพฤติกรรมที่มีการเรียนรู้ไปมาทดลองให้ระบบประมวลผล ผลที่ได้ ไม่มีการแจ้งเตือนว่าเป็นผู้บุกรุกไปยังผู้ดูแลระบบ แสดงว่าการเรียนรู้พฤติกรรมการใช้งานของ A ถูกต้อง

ผลการทดลองที่ 2 จากการสังเกตการทำงานของผู้ใช้งาน B มีการใช้งานดังนี้

Jan_30_2003		Jan_29_2003		Jan_28_2003		Jan_27_2003		Jan_26_2003	
bash	12	bash	7	bash	2	bash	4	bash	5
cat	9	cat	2	cat	1	cat	2	cat	3
grep	3	ftp	1	ipop3d	8	grep	3	ipop3d	11
ipop3d	3	ipop3d	7	ksh	1	ipop3d	3	ksh	3
ksh	6	ksh	2	mail	1	ksh	2	ls	3
ls	15	ls	5	pine	1	ls	3	mail	4
mail	6	mail	2	quota	1	mail	2	more	1
more	2	more	3	sshd1	1	quota	2	pine	3
mv	1	pico	1	stty	3	ssh1	10	quota	3
pine	4	pine	2	tty	1	sshd1	2	sendmail	7
ping.sun	2	quota	2	w	2	stty	6	sh	2
quota	14	rm	2			tty	2	sshd1	3
ssh1	7	sshd1	2					stty	9
sshd1	6	stty	6					tty	3
stty	18	tty	2					w	1
tty	6	w	1					xauth	1
w	2								

Jan_25_2003	Jan_24_2003	Jan_23_2003	Jan_22_2003	Jan_21_2003
bash 2	bash 3	bash 5	bash 1	bash 10
cat 1	cat 2	cat 5	cat 1	cat 2
imapd 3	ipop3d 4	grep 1	ipop3d 9	chmod 1
ipop3d 16	ksh 2	ipop3d 6	ksh 1	clear 1
ksh 1	mail 2	ksh 5	mail 1	col 1
ls 10	pine 3	ls 3	pine 1	gdb 2
mail 1	quota 2	mail 5	quota 1	ipop3d 6
pine 2	sendmail 4	man 1	sshd1 1	ksh 1
quota 1	sshd1 2	more 7	stty 3	ls 6
sshd1 1	stty 6	pine 6	tty 1	mail 2
stty 3	tty 2	quota 5	w 1	man 1
tty 1		sendmail 4		more 2
w 1		sh 1		mv 1
		sshd1 5		nroff 1
		stty 15		ping.sun 1
		tty 5		quota 2
		w 5		sh 3
				ssh1 4
				sshd1 2
				stty 6
				tput 1
				tty 2
				w 1

Jan_20_2003	Jan_19_2003	Jan_18_2003	Jan_17_2003	Jan_16_2003
bash 5	ipop3d 6	bash 5	bash 4	bash 1
cat 2		cat 3	cat 2	cat 1
col 3		col 1	ipop3d 22	ipop3d 24
finger 2		ipop3d 9	ksh 2	ksh 1
grep 1		ksh 3	ls 1	mail 1
ipop3d 5		ls 1	mail 2	pine 2
ksh 2		mail 3	more 4	quota 1
last 1		man 3	quota 2	sh 1
ls 4		more 1	sh 3	sshd1 1
mail 1		mv 1	ssh1 2	stty 3
man 4		nroff 1	sshd1 2	tty 1
more 4		pine 1	stty 6	w 1
mv 3		quota 3	tty 2	xauth 1
nroff 3		sh 4		
ping.sun 8		sshd1 3		
quota 1		stty 9		
sh 9		tty 3		
ssh1 41		uname 1		
sshd1 1		w 1		
stty 3		xauth 1		
talk 3				
telnet 5				
tty 1				
w 4				

ตารางที่ 6-2 แสดง โพรไฟล์ของผู้ใช้ชื่อ B

พฤติกรรมที่นำมาตรวจสอบ มีดังนี้

#Jan_31_2003

as	16
cat	9
cc1	19
col	3
comm	6
cpp	19
deroff	6
finger	5
gcc	25
ksh	42
ld	16
ls	20
mail	3
man	6
mesg	2
more	4
mv	3
nroff	3
pico	12
pine	1
quota	3
rm	6
sed	24
sendmail	4
sh	16
sort	6
spell	24
spellpro	12
stty	9
talk	7
tbl	2
tee	6
tr	6
tty	3
uname	1
what	2
who	4
write	12

ทดลองนำพฤติกรรมของ B มาใช้ในวันที่ตรวจจับซึ่งจะเห็นได้ว่ามีความแตกต่างกับพฤติกรรมที่มีการเรียนรู้ไปมาทดลองให้ระบบประมวลผล ผลที่ได้มีการแจ้งเตือนว่า B เป็นผู้บุกรุกไปยังผู้ดูแลระบบแสดงว่าการเรียนรู้พฤติกรรมการใช้งานของ B ถูกต้อง

เปอร์เซ็นต์ความผิดพลาด

a) ตรวจพบว่าเป็นการโจมตีจากข้อมูลทั่วไป (False negative)

ผลจากการทำงานของระบบเมื่อระบบผ่านการเรียนรู้จากการเรียนรู้ทั้งหมด 30 คน แล้วลองนำพฤติกรรมที่ปกติของผู้ใช้งานแต่ละคนซึ่งมีความเปลี่ยนแปลงในแต่ละวันบ้างเล็กน้อยตามความเป็นจริงในการใช้งานมาทดสอบ ปรากฏว่าระบบตรวจจับผิดพลาด (False negative)

เป็นจำนวนทั้งสิ้น 7 คน คิดเป็นประมาณ 23.33 % ทั้งนี้อาจเกิดจากข้อมูลที่เรียนรู้มีปริมาณน้อยเกินไป

b) ความผิดพลาดที่เกิดจากการตรวจไม่พบการโจมตี จากผู้ที่เข้ามาโจมตี (False Positive)

ผลจากการทำงานของระบบเมื่อระบบผ่านการเรียนรู้จากการเรียนรู้ทั้งหมด 30 คน แล้วทดลองนำพฤติกรรมที่ผิดปกติมาทำการทดสอบกับผู้ใช้งานแต่ละคน ผลที่ได้ปรากฏว่าควรมีการตรวจสอบว่าเป็นผู้บุกรุกทุกคนได้เกือบทั้งหมด คือมีที่ตรวจสอบไม่ได้เพียง 1 คน คิดเป็นประมาณ 3 %

6.2 ผลการทดลองปรับเปลี่ยนค่าพารามิเตอร์ต่างๆ ที่มีผลต่อการทดลอง

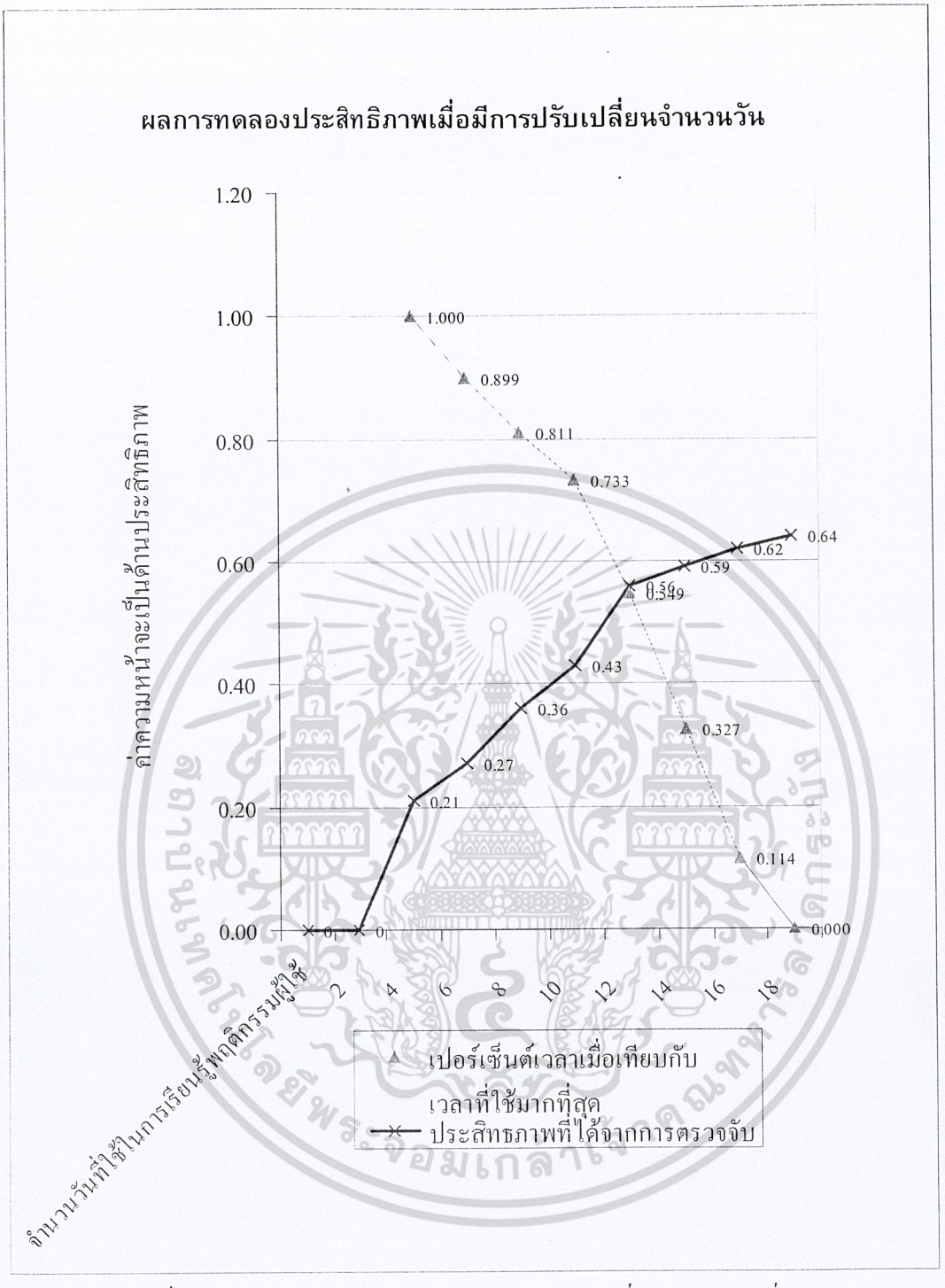
โดยการปรับเปลี่ยนค่าพารามิเตอร์นั้น ก็เพื่อต้องการหาค่าที่ดีที่สุดในการทำงาน คือ มีความสามารถในการตรวจจับผู้บุกรุก และใช้เวลาในการทำงานน้อย

6.2.1 การกำหนดระยะเวลาในการเก็บพฤติกรรมของแต่ละผู้ใช้งานระบบ

ในการกำหนดระยะเวลาในการเก็บพฤติกรรมของผู้ใช้งานนั้น หากกำหนดเวลาที่ใช้น้อยเกินไปก็ไม่สามารถจะระบุพฤติกรรมบุคคลนั้นได้ แต่ถ้าหากกำหนดมาเกินไปช่วงเวลาที่ใช้ในการจัดเก็บที่มากขึ้นก็จะทำให้มีความเสี่ยงในการบุกรุกมากขึ้นด้วย

จำนวนวันที่ใช้ในการเรียนรู้พฤติกรรมผู้ใช้	จำนวนครั้งที่ใช้ในการเรียนรู้	เปอร์เซ็นต์เวลาเมื่อเทียบกับเวลาที่ใช้มากที่สุด	ประสิทธิภาพที่ได้จากการตรวจจับ
2	ไม่สามารถแยกแยะได้	-	0
4	ไม่สามารถแยกแยะได้	-	0
6	6.342	1.000	0.21
8	7.877	0.899	0.27
10	9.249	0.811	0.36
12	11.136	0.733	0.43
14	62.854	0.549	0.56
16	92.365	0.327	0.59
18	101.736	0.114	0.62
20	112.421	0.000	0.64

ตารางที่ 6-3 แสดงผลการทดลองประสิทธิภาพเมื่อมีการปรับเปลี่ยนจำนวนวัน



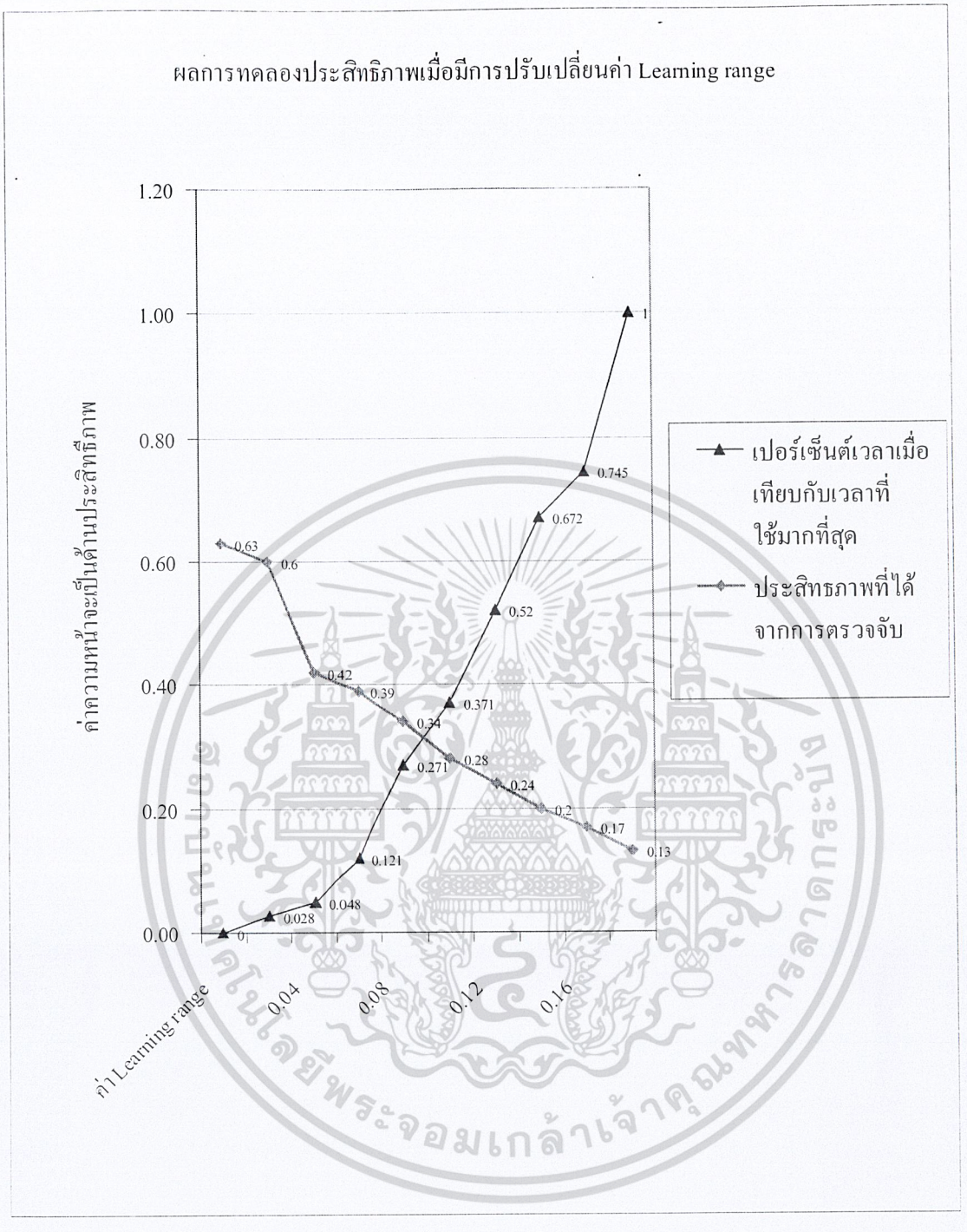
รูปที่ 6-1 แสดงแผนภูมิผลการทดลองประสิทธิภาพเมื่อมีการปรับเปลี่ยนจำนวนวัน

6.2.2 การปรับเปลี่ยนค่าความสามารถในการเรียนรู้

หากมีค่า α น้อยระบบก็จะมีความสามารถแยกประเภทของผู้ใช้งานได้ดีแต่ ก็ทำให้เวลาที่ใช้ในการเรียนรู้เพิ่มมากขึ้นด้วย ในทำนองเดียวกันหากมีค่า α มากระบบก็จะมีความ สามารถในการแยกประเภทของผู้ใช้งานแต่ละคนได้ไม่ดีเท่าที่ควร แต่ทำให้เวลาในการเรียนรู้น้อยลง

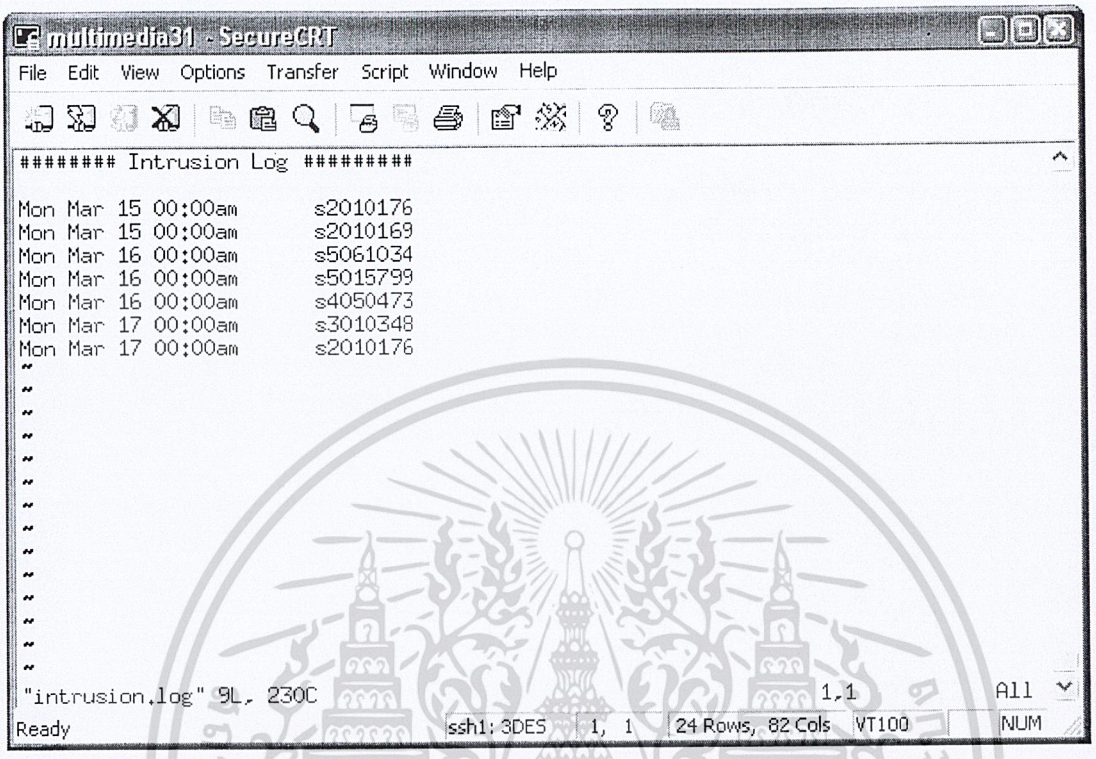
ค่า Learning range	จำนวนครั้งที่ใช้ในการเรียนรู้	เปอร์เซ็นต์เวลาเมื่อเทียบกับเวลาที่ใช้มากที่สุด	ประสิทธิภาพที่ได้จากการตรวจจับ
0.02	95,362	0	0.63
0.04	72,365	0.028	0.6
0.06	65,824	0.048	0.42
0.08	52,136	0.121	0.39
0.1	38,698	0.271	0.34
0.12	29,723	0.371	0.28
0.14	16,238	0.52	0.24
0.16	9,621	0.672	0.2
0.18	7,856	0.745	0.17
0.2	5,325	1	0.13

ตารางที่ 6-4 แสดงผลการทดลองประสิทธิภาพเมื่อมีการปรับเปลี่ยนค่า Learning rate



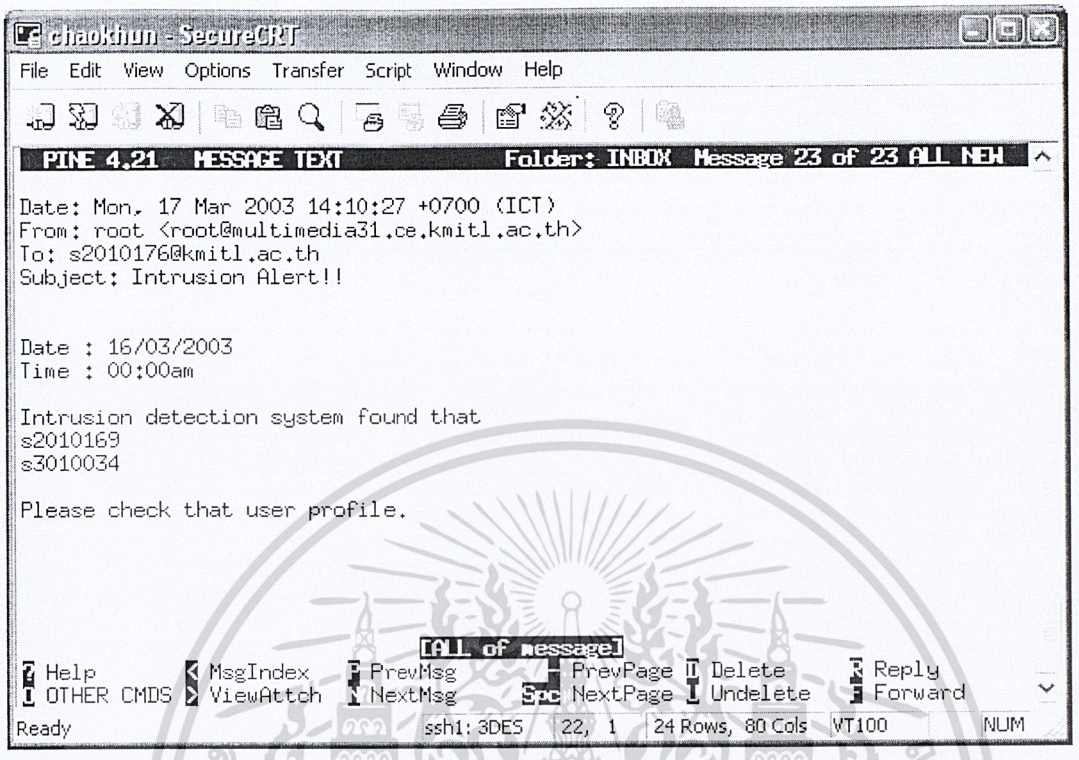
รูปที่ 6-2 แสดงแผนภูมิผลการทดลองประสิทธิภาพเมื่อมีการปรับเปลี่ยนค่า Learning rate

6.3 ส่วนแสดงผลการทำงานของโปรแกรม



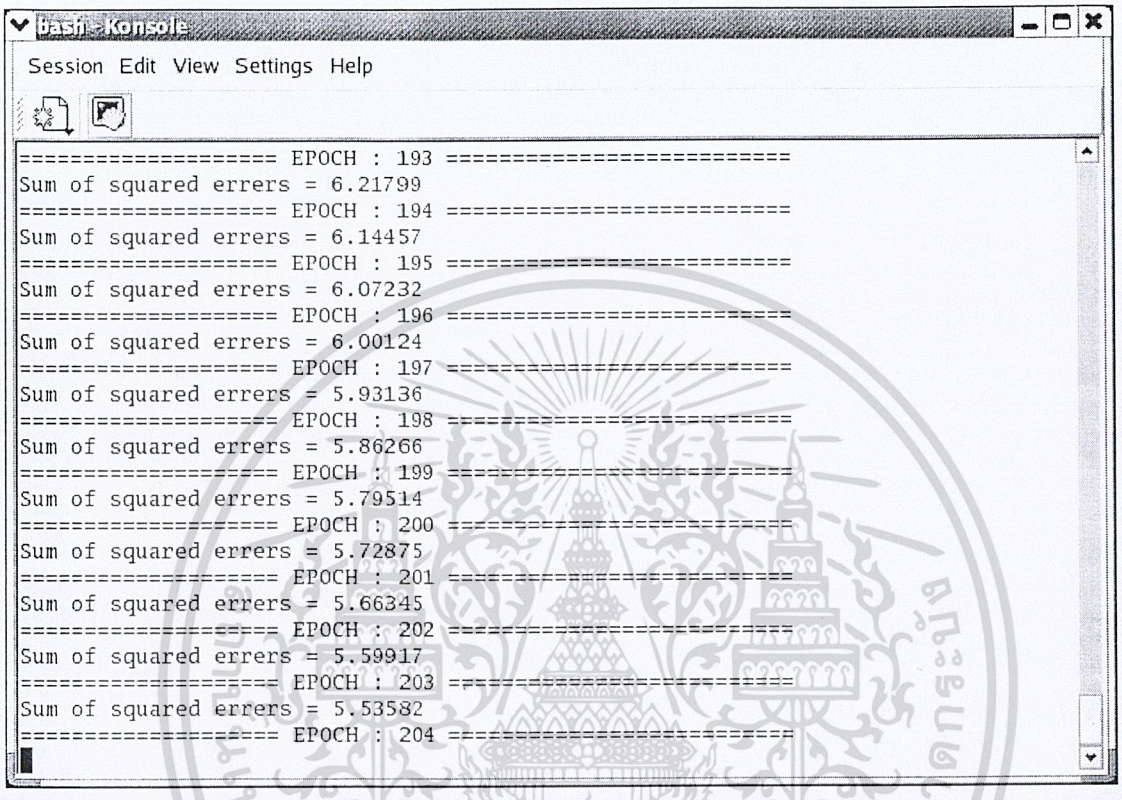
รูปที่ 6-3 แสดงผล log file ของโปรแกรมที่เก็บการทำงานของโปรแกรม

6.3.2 การส่งอีเมล์แจ้งเตือนผู้ดูแลระบบ ระบบจะส่งอีเมล์ไปยังผู้ดูแลระบบเมื่อตรวจพบการบุกรุก



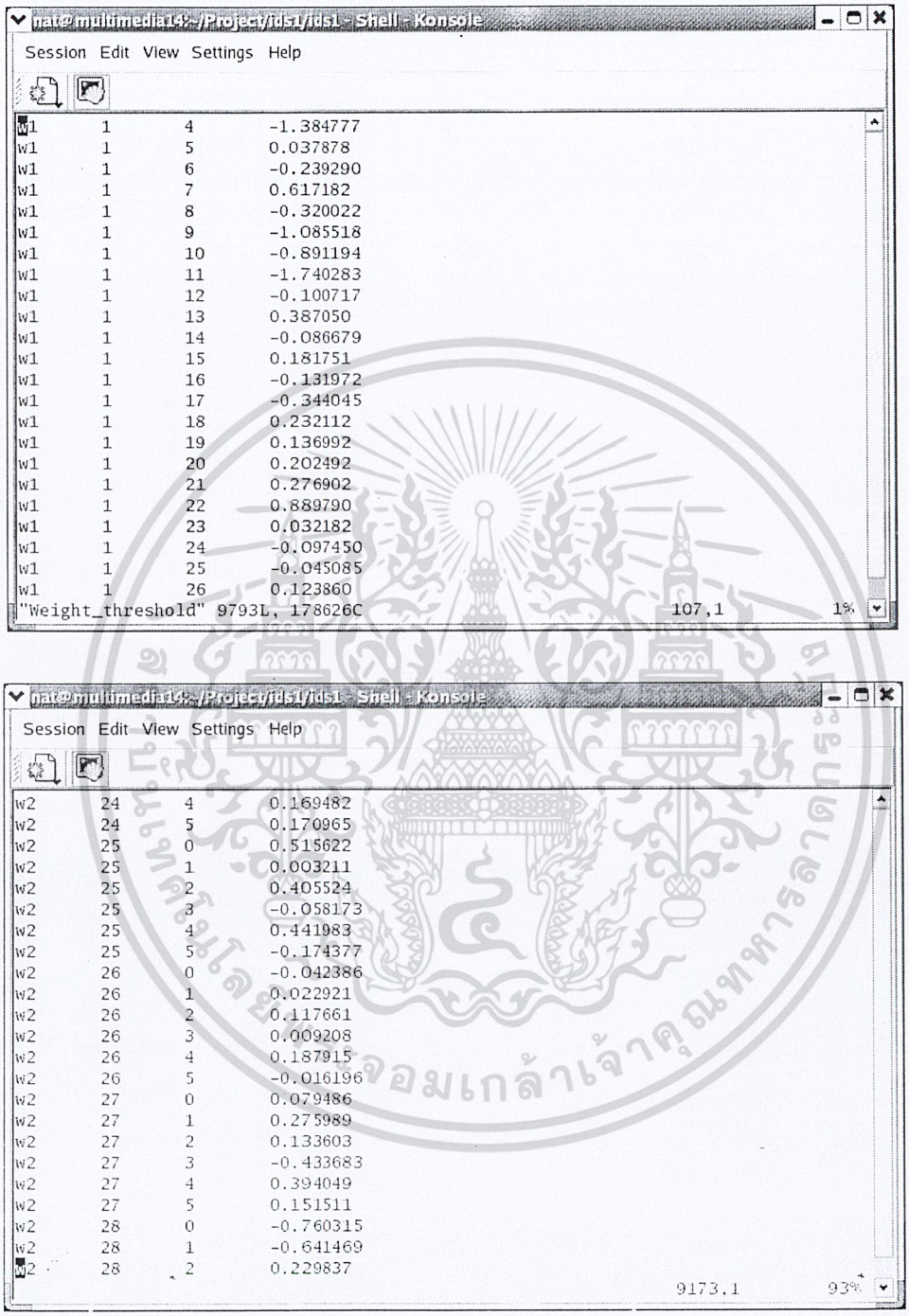
รูปที่ 6-4 แสดงผลการส่งรหัสที่ตรวจจับได้ไปยังผู้ดูแลระบบ

6.3.3 ส่วนแสดงผลการเรียนรู้ขณะที่โปรแกรมทำงาน เมื่อระบบเริ่มทำงานระบบจะแบ่งการทำงานเป็น 2 ส่วนคือส่วนที่ตรวจจับผู้บุกรุกโดยใช้ค่า weight และ threshold ที่ได้เรียนรู้มาแล้วมาทำงาน แต่จากนั้นก็จะทำการปรับค่า weight และ threshold ใหม่เพื่อได้ข้อมูลที่นำมาวิเคราะห์เป็นปัจจุบันที่สุด

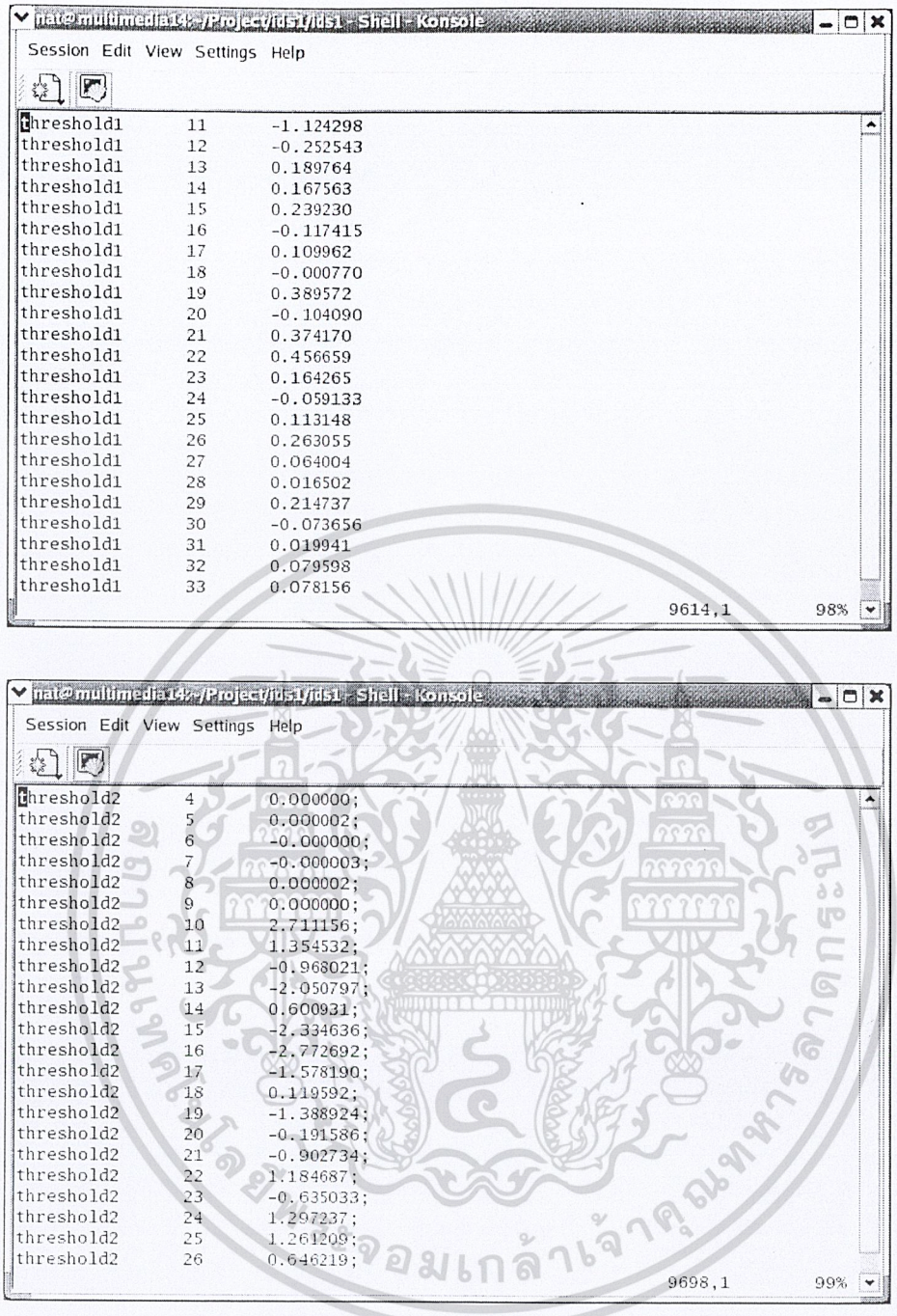


รูปที่ 6-5 แสดงผลขณะที่โปรแกรมทำการเรียนรู้

6.3.4 ส่วนแสดงผลเมื่อโปรแกรมทำการเรียนรู้เสร็จแล้ว เมื่อทำการเรียนรู้เรียบร้อยแล้วค่าที่ได้คือค่า weight และ threshold โดยมีผลดังนี้



รูปที่ 6-6 แสดงผลค่า Weight ที่ได้จากการเรียนรู้แล้ว



รูปที่ 6-7 แสดงผลค่า Threshold ที่ได้จากการเรียนรู้แล้ว

หมายเหตุ : ค่า Weight และ Threshold ที่ได้จะนำไปใช้ในการตรวจจับผู้บุกรุก

บทที่ 7
วิเคราะห์ผลการทดลองและสรุป

7.1 วิเคราะห์ผลการทดลอง

จากการทดลองพบว่าระบบตรวจจับผู้บุกรุกคอมพิวเตอร์ที่มีการนำปัญญาประดิษฐ์มาใช้งานสามารถตรวจจับการบุกรุกที่มีพฤติกรรมการใช้งานผิดไปจากปรกติมากๆได้ และในกรณีที่มีพฤติกรรมที่เบี่ยงเบนจากพฤติกรรมเดิมไม่มากนักก็จะยังไม่ถือว่าเป็นการบุกรุก ทั้งนี้เป็นข้อดีของการใช้ นิวรอนเน็ตเวิร์กที่ทำให้ระบบมีความยืดหยุ่นมากยิ่งขึ้น แต่อย่างไรก็ตามระบบสามารถตรวจพบรูปแบบการบุกได้ยังไม่ครอบคลุมทั้งหมด เช่น กรณีที่ผู้บุกรุกมีการปรับเปลี่ยนพฤติกรรมเพียงเล็กน้อย ทำให้ระบบไม่สามารถตรวจพบได้ หรือ ในกรณีที่เป็นการบุกรุกผ่านทางเน็ตเวิร์ก เป็นต้น

จากการทดลองมาสามารถเปรียบเทียบประสิทธิภาพกับระบบตรวจจับผู้บุกรุกแบบอื่นได้ เนื่องจากระบบตรวจจับผู้บุกรุกที่สร้างขึ้นนี้ สร้างขึ้นเพื่อทดลองนำเอาทฤษฎีด้านปัญญาประดิษฐ์มาประยุกต์ใช้งานกับระบบตรวจจับผู้บุกรุกเท่านั้น ทำให้ผลที่ได้ยังไม่ครอบคลุมการบุกรุกมากนัก แต่สามารถนำไปพัฒนาเสริมกับระบบอื่นได้

7.2 สรุปผล

ระบบตรวจจับผู้บุกรุกคอมพิวเตอร์ที่มีการนำปัญญาประดิษฐ์มาใช้งานสามรถตรวจจับผู้บุกรุกได้ดีในระดับที่น่าพอใจ และมีความยืดหยุ่นในการใช้งานมากขึ้น โดยสามารถเรียนรู้พฤติกรรมของผู้ใช้และทำการตรวจจับพฤติกรรมที่ผิดปรกติไปจากเดิมได้ แต่ในกรณีที่มีความผิดปรกติเพียงเล็กน้อยจะไม่ระบุว่าเป็นพฤติกรรมที่ผิดปรกติซึ่งก็ตรงตามความเป็นจริงที่บางทีอาจมีการใช้งานผิดแปลกไปจากเดิมบ้างแต่อย่างไรก็ตามก็ไม่เกินขอบเขตที่ที่สามารถยอกรับได้ แต่อย่างไรก็ตาม ระบบตรวจจับผู้บุกรุกนี้ยังไม่สามารถนำไปใช้งานได้จริงเพราะระบบไม่ได้ครอบคลุมการบุกรุกในแบบอื่นๆอีกหลายแบบทั้งนี้ในโครงงานนี้มีจุดประสงค์เพียงต้องการ สร้างนำเอาทฤษฎีด้านปัญญาประดิษฐ์มาประยุกต์ใช้งานกับระบบตรวจจับผู้บุกรุกเท่านั้น ดังนั้นหากต้องการนำไปใช้จริงต้องมีการนำไปพัฒนาให้ทำงานร่วมกับส่วนอื่นเพื่อให้ครอบคุลมการบุกรุกมากขึ้นต่อไป

7.3 แนวทางในการพัฒนาต่อสำหรับผู้สนใจในอนาคต

ประสิทธิภาพของโปรแกรมอยู่ในเกณฑ์ที่พอใช้ได้ แต่ยังมีข้อเสียอยู่บางประการ ทั้งนี้ด้วยข้อจำกัดทางด้านเวลาในการพัฒนาโปรแกรม ดังนั้นหากมีเวลามากกว่านี้น่าจะลองปรับปรุงโปรแกรมโดยการทดลองหาสมการและวิธีการของนิวรอนเน็ตเวิร์กที่เหมาะสมมากยิ่งขึ้น เพื่อให้ผลที่ออกมามีความแม่นยำมากยิ่งขึ้น

- ทดลองกับ Neural แบบอื่นๆเพื่อหาแบบที่เหมาะสมที่สุด

- เพิ่มรูปแบบของ Input แทนที่จะเป็น คำสั่งหรือโพรเซสอย่างเดียวเป็นการวิเคราะห์ปริมาณการใช้งาน I/O, ปริมาณการใช้งาน CPU, การเรียกใช้ System call หรือทรัพยากรอื่นๆเป็นต้น
- เพิ่มเติมการตรวจจับการบุกรุกโดยเปรียบเทียบกับพฤติกรรมผู้ใช้ในรูปแบบที่รู้จัก โดยอาจนำเอา ระบบผู้เชี่ยวชาญ (Rule-base expert system) มาร่วมพัฒนา ทำให้ผลที่ออกมามีความแม่นยำมากยิ่งขึ้น
- พัฒนาให้ระบบเป็นแบบเรียลไทม์ (Real-time) มากขึ้น เพื่อให้เตือนผู้ดูแลระบบได้ทันท่วงที
- พัฒนาให้สามารถระบุเปอร์เซ็นต์ความน่าจะเป็นการบุกรุกได้ด้วย โดยอาจนำฟัซซี่ลอจิกมาใช้กับผลที่ออกมาจากนิวรอนเน็ตเวิร์กอีกทีหนึ่ง
- เพิ่มเติมการเข้ารหัสของโพรไฟล์ของผู้ใช้แต่ละคน เพื่อป้องกันการลอกเลียนพฤติกรรม

บรรณานุกรม

หนังสืออ้างอิง

[1] Rebecca Gurley Bace. Intrusion Detection. McMillan Technical Publishing, Indianapolis, 2000.

[2] William Stallings, Ph.D. Operating Systems Internals and Design Principles. Practices-Hall, New Jersey, 2001.

[3] Stuart J. Russell and Peter Norring. Artificial Intelligence: A Modern Approach. Practices-Hall, New Jersey, 1995.

[4] Michael Negnvitsky. Artificial Intelligence: A Guide to Intelligent Systems. Addison-Wesley, Essex, 2002.

[5] (Editor) วศิน เพิ่มทรัพย์ม สิคารัศมี ธำรงสมบัติสกุล : "คู่มือติดตั้งและใช้งาน Linux ", โครงการลีนุกซ์ภาษาไทย บริษัท ไกวัล ซอฟท์แวร์ จำกัด,2542

[6] นุกูล กระจาย : "การเขียนโปรแกรมในดอสและวินโดวส์ด้วยบอร์แลนด์C++5.0", บริบัท ซีเอ็ดยูเคชัน จำกัด(มหาชน),2540

[7] มนตรี พจนารถถลาวัณย์ : "การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี", บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน), 2540

[8] สันติ ศรีลาศักดิ์, วรวุฒิ เที่ยงธรรม : "เจาะประเด็นงานเขียนโปรแกรมบนลีนุกซ์", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน),2540

[9] ภัทรพงศ์ น้อยเรือง, ประภาพร ช่างไม้ : คู่มือการใช้งาน Linux ฉบับ Admin, บริษัทดวงกลมสมัย จำกัด

เว็บไซต์อ้างอิง

[10] http://www.linuxsecurity.com

[11] http://www.sans.org

[12] http://plg.uwaterloo.ca/~itbowman/CS746G/a1/

[13] http://www.tldp.org

[14] http://www.ieee.org

[15] http://thaicert.nectec.or.th

[16] http://www.freeos.com/guides/lsst/

[17] http://www.netti.hu/doc/LinuxShellScript/

[18] http://ieee-nns.org/

[19] http://www.inns.org/

[20] http://www.emsl.pnl.gov:2080/proj/neuron/