ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

Intrusion Detection System

นายสุเทพ แก้ววงค์

นายเสือ อินจ๋อย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

Intrusion Detection System

โดย

นายสุเทพ แก้ววงค์

นายเสือ อินจ๋อย

อาจารย์ที่ปรึกษา

อ.ธนา หงษ์สุวรรณ

อ.อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจจับผู้บุกรุกคอมพิวเตอร์

Intrusion Detection System

ผู้จัดทำ

นาย สุเทพ แก้ววงค์ รหัสประจำตัว 43015389(3P)

นาย เสือ อินจ๋อย รหัสประจำตัว 43015393(3P)

อาจารย์ที่ปรึกษา

(นาย ธนา หงษ์สุวรรณ)

อาจารย์ที่ปรึกษา

(นายอัครเดช วัชระภูพงษ์)

ระบบตรวจจับผู้บุกรุกคอมพิวเตอร์

นาย สุเทพ แก้ววงค์ 43015389(3P)
นาย เสือ อินจ๋อย 43015393(3P)

อ.ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อ.อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา

บทคัดย่อ

เป็นที่รู้กันดีว่าระบบคอมพิวเตอร์และเครือข่ายมีการออกแบบที่ไม่ค่อยปลอดภัยมากนัก แม้ว่ามีไฟร์วอล (Firewall) ก็ตามแต่ก็ไม่สามารถป้องกันการบุกรุก 100 เปอร์เซ็นต์ เนื่องจากไฟร์วอลเป็นการป้องกันระหว่างระบบคอมพิวเตอร์ภายในเครือข่าย (Internal networks) กับระบบภายนอก (Internet) การมีไฟร์วอล เปรียบเสมือนการมีรั้วกั้นรอบบ้าน และระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์เปรียบเสมือนการติดตั้งกล้องวีดีโอคอยตรวจสอบสิ่งผิดปกติทำให้การป้องกันระบบทำได้ดียิ่งขึ้น

โครงงานนี้จัดทำระบบตรวจจับผู้บุกรุกคอมพิวเตอร์บนระบบปฏิบัติการลีนุกซ์ซึ่งเป็นระบบที่ช่วยผู้ดูแลระบบคอมพิวเตอร์ตรวจสอบหาผู้บุกรุกซึ่งอาจเข้ามาทำความเสียหายระบบคอมพิวเตอร์ วิธีการทำงานเป็นการตรวจสอบการบุกรุกบนฝั่งโฮสต์ (Host-based IDS) ทำงานโดยตรวจสอบซีสเต็มคอลของผู้ใช้งานกับฐานข้อมูลผู้ใช้งานปกติว่ามีการใช้งานที่ผิดปกติหรือไม่ (Misuse detection) อาศัยหลักการที่ว่าปกติแล้วโปรแกรมจะมีรูปแบบการทำงานของซิสเต็มคอลอยู่รูปแบบหนึ่งซึ่งไม่แตกต่างไปจากเดิมถ้าเราสามารถเก็บรูปแบบการทำการดังกล่าวได้ครบ แล้วนำซีสเต็มคอลของโปรแกรมที่ทำงานตามลำดับที่เป็นปกติมาเก็บลงฐานข้อมูลเป็นฐานข้อมูลผู้ใช้งานปกติ จากนั้นนำข้อมูลลำดับของซีสเต็มคอลที่ต้องการตรวจสอบของผู้ใช้งานเข้ามาเปรียบเทียบ โดยใช้ Hamming Distance คำนวณหาค่าความอันตราย พร้อมทั้งนับจำนวนความแตกต่าง ถ้าพบความแตกต่างเกินกว่าที่ยอมรับได้ก็จะแจ้งเตือนผู้ดูแลระบบ

Intrusion Detection System

Mr. Suthep Kaewwong
Mr. Sua Injoy
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor

ABSTRACT

As we know, Computer system and network usually are insecure designed. Although the system has a firewall to prevent but it's not 100% safe because firewall is a prevention between Internal network and Internet. Firewall is like a fence and intrusion detection system is like a video camera to watch out for any wrong that could happen. So intrusion detection system can make a system to be more secured.

Intrusion detection system on Linux is the system that helps an administrator to detect any intrusion that could destroy the computer system. Intrusion detection system will detect the invasion on host (Host-based IDS) by inspect the user's system call and database compare with the usual system call.

If there is some different occurs, intrusion detection system will inform to an administrator.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จสมบูรณ์ได้ถ้าไม่ได้รับคำแนะนำจาก อาจารย์ ธนา หงษ์สุวรรณ และ อาจารย์ อัครเดช วัชระภูพงษ์ คณะผู้จัดทำขอขอบพระคุณยิ่งสำหรับคำแนะนำที่ได้รับจากท่านทั้งสอง และขอบพระคุณอาจารย์ทุกท่านในสถาบันแห่งนี้ที่ได้สั่งสอนคณะผู้จัดทำจนมีความรู้ความสามารถจนถึงทุกวันนี้

ขอบคุณภาควิชาคอมพิวเตอร์ โดยเฉพาะห้องวิจัยและพัฒนาการรักษาความปลอดภัยข้อมูล (ISAG) ที่ได้เอื้อเฟื้อสถานที่ให้คณะผู้จัดทำได้ทำการวิจัยและช่วยอำนวยความสะดวกต่าง ๆ ขอขอบคุณพี่ ๆ เพื่อนๆ น้อง ๆ ชาว ISAG ที่ให้ความช่วยเหลือในการทำงานตลอดเวลา เป็นที่ปรึกษายามมีปัญหา รวมทั้งให้ยืมทรัพยากรที่จำเป็นต่าง ๆ

ที่สำคัญและขาดมิได้ คือ ต้องขอขอบคุณบิดา มารดาที่ได้ให้กำเนิด คอยสั่งสอน และให้การสนับสนุนการศึกษา นับเป็นพระคุณอย่างยิ่ง ทางคณะผู้จัดทำขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

สารบัญ

สารบัญภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เป็นที่รู้กันดีว่าระบบคอมพิวเตอร์และเครือข่ายมีการออกแบบที่ไม่ค่อยปลอดภัยมากนักทำให้ผู้บุกรุกมีโอกาสที่จะบุกรุกระบบได้ง่ายแม้ว่ามีไฟร์วอล (Firewall) อยู่แล้วก็ตามแต่ก็ไม่สามารถป้องกันการบุกรุก 100 เปอร์เซ็นต์ เนื่องจากไฟร์วอลเป็นการป้องกันระหว่างระบบคอมพิวเตอร์ภายในเครือข่าย (Internal networks) กับระบบภายนอก (Internet) หากผู้บุกรุกเป็นคนภายในไฟร์วอลเอง ก็ไม่สามารถป้องกันได้ ดังนั้นการมีไฟร์วอล เปรียบเสมือนการมีรั้วกั้นรอบบ้าน และระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์เปรียบเสมือนการติดตั้งกล้องวีดีโอคอยตรวจสอบสิ่งผิดปกติทำให้การป้องกันระบบทำได้ดียิ่งขึ้นอีกทั้งระบบปฏิบัติการลีนุกซ์เป็นระบบปฏิบัติการที่เป็นที่รู้จักทั่วโลกซึ่งมีอัตราการเจริญเติบโตสูงและมีแนวโน้มที่จะถูกนำมาใช้อย่างแพร่หลายใน แต่ว่าลีนุกซ์ยังมีปัญหาด้านความน่าเชื่อถือของระบบเนื่องมาจากเป็นซอฟต์แวร์ฟรีและเปิดเผยซอร์สโค้ดที่สามารถดาวน์โหลดได้ทางอินเทอร์เน็ต ดังนั้นลีนุกซ์จึงยังมีความปลอดภัยไม่เพียงพอสำหรับการใช้งานในองค์กรที่ต้องการความปลอดภัยสูง

จากสาเหตุที่กล่าวมานี้ ผู้จัดทำโครงงานจึงคิดทำระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์ซึ่งทำงานอยู่บนระบบปฏิบัติการลีนุกซ์นี้ขึ้น

1.2 วัตถุประสงค์

1. เพื่อให้เข้าใจการทำงานและการเขียนแอพลิเคชันบนระบบปฏิบัติรการลีนุกซ์
2. เพื่อสร้างระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์บนระบบปฏิบัติการลีนุกซ์ ใช้วิธีการตรวจจับผู้บุกรุกโดยตรวจสอบลำดับการทำงานของซีสเต็มคอล เพื่อตรวจจับผู้บุกรุกระบบคอมพิวเตอร์ได้ โดยทำงานแบบ Misuse detection

1.3 ขอบเขตของงานวิจัย

1 สร้างระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์บนระบบปฏิบัติการลีนุกซ์เพื่อให้สามารถตรวจจับหาผู้บุกรุกได้

2 สามารถแจ้งเตือนผู้บุกรุกได้ทันทีเมื่อมีการตรวจพบการบุกรุก

3 สามารถทำงานได้บนระบบปฏิบัติการลีนุกซ์เคอร์เนลเวอร์ชั่น 2.4.18 และเวอร์ชั่นที่มีรูปแบบของซิสเต็มคอลเหมือนกัน

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาแนวทางในการพัฒนาระบบตรวจจับผู้บุกรุก วิธีการตรวจจับผู้บุกรุกและศึกษาข้อดีข้อเสียความสามารถของระบบตรวจจับผู้บุกรุกที่ใช้งานกันอยู่ในปัจจุบัน

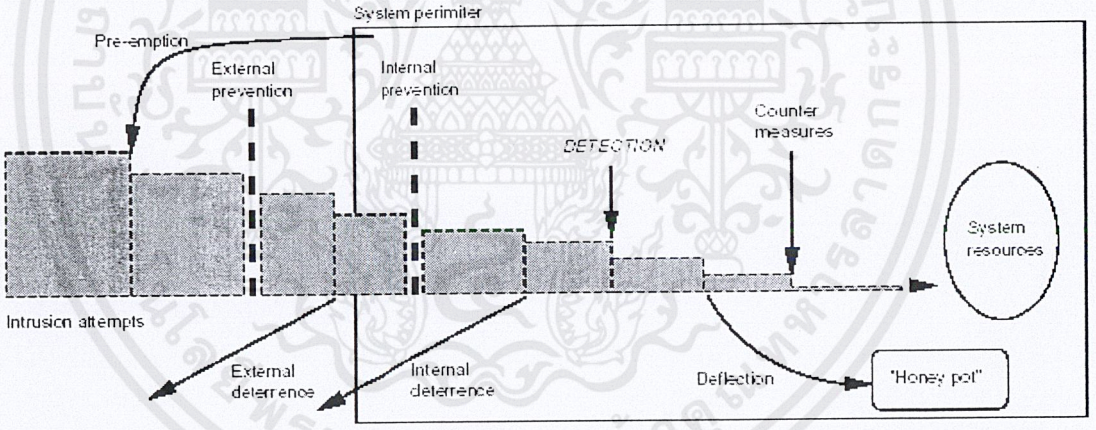
2. ศึกษาการเขียนโปรแกรมบนเคอร์เนล(Kernel)เพื่อตรวจจับการทำงานของซีสเต็มคอล
3. กำหนดรายละเอียด (Specification) ของระบบทั้งหมดและทำการออกแบบระบบ
4. เขียนโปรแกรม
5. ทดสอบระบบและแก้ไขข้อผิดพลาด

บทที่ 2
ความรู้เบื้องต้นของการตรวจจับผู้บุกรุก

เพื่อให้มองเห็นภาพของการตรวจจับผู้บุกรุก ให้คิดถึงภาพของระบบป้องกันขโมยทั่ว ๆ ไป เมื่อเทียบกับระบบคอมพิวเตอร์หรือระบบเครือข่ายในแบบเดียวกัน ก็คือการมีการตรวจจับการฝ่าฝืนนโยบายทางด้านการรักษาความปลอดภัยที่เป็นน่าจะเกิดขึ้น จะส่งสัญญาณเตือนไปให้ผู้มีอำนาจหรือผู้ที่รับผิดชอบ (ในที่นี้หมายถึง SSO ซึ่งย่อมาจาก Site Security Officer) ระบบทั้งสองแบบนี้ก็มีบางรูปแบบของปัญหาที่คล้ายกัน เช่น การส่งสัญญาณเตือนที่ผิดพลาด การหลบหลีกการตรวจจับของระบบเตือนภัย

2.1 รูปแบบต่าง ๆ ของระบบตรวจจับการบุกรุกในขอบเขตที่กว้างขึ้น

มีหลายวิธีที่จะใช้ในการปกป้องระบบคอมพิวเตอร์ หรือระบบเครือข่ายจากการถูกโจมตี การจำกัดขอบเขตบริเวณการใช้งานก็เป็นวิธีหนึ่งในรูปแบบที่ใช้กัน ซึ่งรูปแบบของเทคนิคของการต่อต้านการบุกรุก (Anti intrusion) สามารถแบ่งออกได้เป็น 6 รูปแบบอย่างไม่เฉพาะเจาะจงดังนี้



รูปที่ 2.1 สรุปเทคนิคในการต่อต้านผู้บุกรุก

Prevention (การป้องกัน) การสร้างอุปสรรคหรือการขัดขวางไม่ให้เกิดการบุกรุกได้สำเร็จ ตัวอย่างเช่น การที่ตัดสินใจไม่ทำการเชื่อมต่อเครือข่ายเข้ากับอินเตอร์เน็ทถ้ากลัวจะถูกใช้ในหารโจมตี แต่โชคไม่ดีที่ทำแบบนี้สามารถทำให้เกิดค่าใช้จ่ายที่เพิ่มขึ้นและเกิดความล่าช้าขึ้นได้

Preemption (การจัดการก่อน) เป็นการโจมตีกลับไปยังการคุกคาม ที่จะเกิดขึ้นก่อนที่มันจะมีโอกาสสร้างการโจมตีให้เกิดขึ้นได้ นั้นหมายถึงการพยายามค้นหาการบุกรุกรูปแบบใหม่ ๆ ที่จะเกิดขึ้นก่อนที่จะมีผู้บุกรุกนำมาใช้ในการบุกรุกระบบจริง ๆ ซึ่งถ้าทำเทคนิคนี้ไปประยุกต์ใช้กับระบบทั่วไปจะทำให้เกิดอันตรายได้ อาจจะมีความผิดทางกฎหมายด้วย

Deterrence (การยับยั้ง) เป็นการทำให้ผู้โจมตี ยั้งรอการโจมตีนั้นไว้ หรือหยุดการโจมตีที่กำลังดำเนินอยู่ โดยทั่วไปจะใช้โดยการเพิ่มมาตรการลงโทษ ผลที่ผู้บุกรุกจะได้รับ (ถ้าเกิดจับหรือตรวจสอบได้) แต่ก็แน่นอน ถ้าทรัพยากรที่ปกป้องมีค่าหรือมีความหมายมากแล้ว การตัดสินใจของผู้โจมตีก็อาจจะเป็น "ไม่เกรงกลัว" ได้ง่าย ๆ การยับยั้งภายในระบบสามารถทำได้โดยการใส่คำเตือนบทลงโทษหากมีการละเมิด ไว้ในข้อความตอนที่มีการเข้าใช้ระบบ (login) บทลงโทษที่เฉียบขาดต่อผู้ที่ทำการโจมตีทั้งจากภายนอกและภายใน จะทำให้เขาเหล่านั้นฉุดคิด ส่วนการยับยั้งภายนอกอาจจะทำได้โดยการตั้งกฎของระบบให้เกี่ยวข้องกับกฎหมายอาชญากรรมคอมพิวเตอร์ และทำให้มีผลบังคับใช้

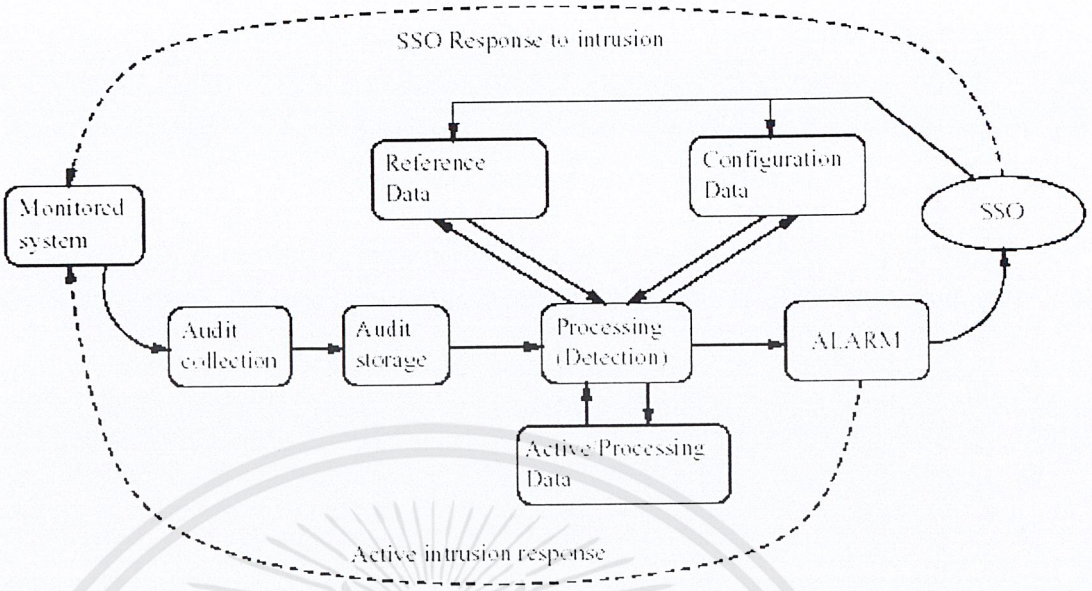
Deflection (การหักเห) เป็นการหลอกล่อให้ผู้ที่บุกรุกเข้าไปในระบบคิดว่าสามารถบุรุกเข้าได้สำเร็จแล้วแต่ในความจริงแล้ว เขาจะถูกแยกออกหรือส่งการทำงานไปยังระบบที่ผู้บุกรุกไม่สามารถทำความเสียหายจริง ๆ ได้ ซึ่งปัญหาหลักของเทคนิคนี้คือใช้จัดการได้กับผู้โจมตีที่ยังมีประสบการณ์น้อยอยู่เท่านั้น

Detection (การตรวจจับ) การตรวจจับมุ่งไปยังการค้นหาการพยายามที่จะบุกรุกและสร้างการตอบสนองต่อเหตุการที่จะเกิดขึ้น แต่ก็มีบ่อยครั้งที่มีการแจ้งเตือนจากผู้ใช้ระบบที่มีสิทธิ์การใช้งานถูกต้อง ซึ่งรูปแบบปัญหาก็มี เรื่องความยากลำบากในการป้องกันการโจมตีจากผู้บุกรุกที่สามารถหลบหลีกออกจากระบบได้อย่างรวดเร็วหลังจากสำเร็จเป้าหมายนั้นแล้ว (Hit and Run) , การส่งสัญญาณเตือนที่ผิดพลาด , หรือการไม่มีสัญญาณเตือนส่งออกมาเมื่อมีผู้บุกรุกสามารถ ปกปิดและซ่อนร่องรอยได้เข้ามาหรือการพยายามที่จะบุกรุกเข้ามา

Countermeasures (การโต้ตอบไปยังการคุกคาม) จะทำงานอยู่ตลอดเวลาและเป็นอิสระในการโต้ตอบกลับไปยังผู้บุกรุกที่พยายามจะโจมตี ซึ่งสามารถกระทำได้โดยไม่ต้องมีการตรวจจับก่อน เนื่องจาก Countermeasure ไม่มีความจำเป็นต้องรู้ลักษณะแตกต่างระหว่างผู้ใช้ที่ได้สิทธิ์ถูกต้องที่กระทำ ผิดพลาดไปกับผู้ที่บุรุก (ถึงแม้ว่าการแยกแยะได้จะเป็นการที่ดีกว่าก็ตาม) ซึ่งเทคนิคที่ว่านี้อาจจะเรียกว่ากับดักก็ได้

2.2 รูปแบบในเชิงสถาปัตยกรรมทั่วไปของระบบตรวจจับผู้บุกรุก

Audit collection เป็นข้อมูลที่ถูกรวบรวมไว้สำหรับตรวจสอบ และให้ส่วนตรวจจับการบุกรุกใช้ในการกำหนดการตัดสินใจ ซึ่งมีเก็บรวมรวมแหล่งข้อมูลจากการตรวจและดูแลการทำงานของระบบในแต่ละส่วนที่แตกต่างกันออกไป เช่น การรับอินพุตจากคีย์บอร์ด ล็อกไฟล์ที่บันทึกการใช้งานคำสั่งหรือใช้โปรแกรมต่าง ๆ แต่อย่างไรก็ดี โดยทั่วไปแล้วข้อมูลที่ถูกใช้จะเป็นไฟล์ที่ใช้บันทึกกิจกรรม ต่าง ๆ ที่เกิดขึ้นในระบบเครือข่ายหรือ ล็อกไฟล์ที่เกี่ยวกับระบบรักษาความปลอดภัยของโฮสต์นั้น ๆ (หรือ ใช้ทั้งสองอย่าง)



รูปที่ 2.2 ส่วนประกอบทั่วไปของระบบตรวจจับการบุกรุก

Audit Storage ข้อมูลที่ใช้ในการตรวจสอบจะถูกจัดเก็บในที่ใดก็ได้ ที่สามารถอ้างอิงได้ในภายหลัง หรือข้อมูลที่ได้จัดเก็บไว้ชั่วคราเพื่อรอการประมวลผล ปริมาณของข้อมูลส่วนมากจะมีขนาดใหญ่มาก (ปัญหาของการเก็บข้อมูลให้เพียงพอ แต่ไม่เก็บข้อมูลที่มากจนเกินความจำเป็น) นั่นคือ ส่วนประกอบในระบบตรวจจับการบุกรุกทุกแบบ ที่ทำให้นักวิจัยต้องมาศึกษาค้นคว้าเพิ่มเติมเพื่อแก้ไขปัญหาในการลดขนาดของการจัดเก็บข้อมูล

Processing ส่วนการประมวลผลเป็นหัวใจสำคัญของระบบตรวจจับผู้บุกรุก ซึ่งมีการทำงานที่หลากหลายวิธี และหลากหลายรูปแบบในการที่จะค้นหาหลักฐานของพฤติกรรมที่เป็นที่ต้องสงสัยจากข้อมูลการตวราสอบที่รวบรวมมาทั้งหมด

Configuration data นี่เป็นขั้นตอนที่มีผลกระทบต่อกระบวนการของระบบตรวจจับการบุกรุก นั่นคือ จะจัดเก็บข้อมูลที่รวบรวมมาได้อย่างไร และที่ไหน เราจะส่งสัญญาณโต้ตอบกับการบุกรุกต่าง ๆ ได้อย่างไรด้วยเหตุนี้จึงเป็นหน้าที่หลักของ SSO (Site Security Officer) ในการควบคุม ระบบการตรวจจับผู้บุกรุก ข้อมูลนี้สามารถเพิ่มขนาดและความซับซ้อน สำหรับการติดตั้งระบบตรวจจับการบุกรุกที่ใช้งานจริง ๆ และยิ่งไปกว่านั้นระบบค่อนข้างจะมีความละเอียดอ่อน เนื่องจากการเข้าถึงข้อมูลจะให้รายละเอียดผู้บุกรุกเกี่ยวกับลู่ทางว่าจะไม่สามารถตรวจจับได้

Reference data การอ้างอิงไปยังส่วนจัดเก็บข้อมูลจะให้รายละเอียดเกี่ยวกับ signature ของการบุกรุก หรือ/และ รูปแบบของพฤติกรรมที่ปกติ ซึ่งในกรณีหลังนี้จะต้องมีส่วนของการสังเกตและประมวลผลเพื่อปรับปรุง เรียนรู้รูปแบบใหม่ ๆ ของพฤติกรรมที่เกิดขึ้น การปรับปรุงข้อมูลจะถูกกระทำอยู่บ่อย ๆ ในช่วงเวลาการใช้งานปกติ ซึ่งจะเป็นแบบแบตซ์ (batch oriented) การจัดเก็บ signature ของการบุกรุก

ต้องมีการปรับปรุงอยู่บ่อยโดย SSO ในขณะที่เมื่อ signature ของการบุกรุกใหม่เป็นที่รู้จัก มากขึ้น

การวิเคราะห์การบุกรุกรูปแบบแปลกใหม่เป็นงานที่ต้องใช้ความสามารถสูงบ่อยครั้งการใช้งานจริง ๆ ของระบบการตรวจจับการบุกรุก SSO จะต้องไปสมัครเป็นสมาชิกกับกลุ่มที่ช่วยกันค้นหาและจัดทำ signature ของการบุกรุกแบบต่าง ๆ แต่มันก็เป็นการยากที่จะทำให้ระบบการตรวจจับผู้บุกรุกทำงานร่วมกับ signature ที่มาจากแหล่งข้อมูลต่าง ๆ

Active/processing data คือ ความถี่ของการประมวลผล คือต้องมีการสะสมผลลัพธ์ไปเรื่อย ๆ ตัวอย่างเช่น รายละเอียดเกี่ยวกับการตรวจสอบ signature ต่าง ๆ ของการบุกรุกกับการทำงานแต่ละส่วนซึ่งต้องการเนื้อที่มากในการจัดเก็บข้อมูลที่ถูกใช้งานอยู่

Alarm คือ ส่วนหนึ่งของระบบซึ่งจัดการเอาต์พุตทั้งหมดที่มาจากระบบ ซึ่งระบบสามารถตอบสนองอัตโนมัติต่อกิจกรรมที่ต้องสงสัย หรือ ในแบบธรรมดาที่สุดคือการส่งสัญญาณเตือนไปยัง SSO

ในระบบผสมนั่นคือ การรวมเอา ส่วน Anomaly และ policy based detection ซึ่งในระบบผสมนี้จะต้องมีส่วนการประมวลผลสองแห่ง และต้องมีที่เก็บการติดตั้งค่าต่าง ๆ ของระบบและข้อมูลที่ประมวลผลอยู่สองแห่ง ในส่วนของสัญญาณเตือน (Alarm) จะต้องสามารถตัดสินใจบนฐานของเอาท์พุตที่มาจากส่วนของการตรวจจับทั้งสองรูปแบบหรือมากกว่า

ส่วนที่ได้กล่าวถึงไปแล้วในรูปที่ 2.2 ส่วนการประมวลผลเป็นส่วนที่ต้องได้รับการค้นคว้าและการศึกษาอย่างละเอียด ส่วนอื่น ๆ จะไม่ค่อยให้ความสำคัญมากนัก และจะต้องเน้นย้ำอยู่ในส่วน data collection (ข้อมูลอะไรที่จะเก็บแล้วสามารถทำให้แน่ใจได้ว่า มีการบุกรุกเกิดขึ้น จะสามารถนำข้อมูลมาทำให้เกิดประโยชน์สูงสุดได้อย่างไร) และการเก็บข้อมูลอย่างไรให้มีประสิทธิภาพ อีกคำถามหนึ่งที่ยังไม่ได้คำตอบแน่ชัดคือ เราจะจัดการกับผู้บุกรุกอย่างไร โดยเฉพาะการทำงานระหว่างส่วนของ สัญญาณเตือนและ SSO ที่รับผิดชอบ

2.3 รูปแบบเบื้องต้นที่นำมาใช้ในการตรวจจับผู้บุกรุก

2.3.1 Anomaly base intrusion detection จะแสดงปฏิกิริยาโต้ตอบต่อรูปแบบพฤติกรรมที่ผิดแปลกไปจากปกติ คำว่า "ปกติ" ได้ถูกให้คำจำกัดความในเชิงความสัมพันธ์ที่เกี่ยวข้องกับพฤติกรรมหลัก ๆ (subject behavior) ที่ถูกติดตามมาก่อนหน้านี้ และมีการเรียนรู้พฤติกรรมใหม่ ๆ ต่อเนื่องไปด้วย การปรับปรุงเพื่อเรียนรู้พฤติกรรมเพิ่มขึ้นจะทำเป็นคาบเวลาและอัตโนมัติ นั่นก็คือระบบจะมีการเรียนรู้พฤติกรรมของผู้ใช้ตลอดเวลา ให้สังเกตพฤติกรรมหลัก ๆ นี้อาจถูกตีความอย่างกว้าง ๆ ได้ ไม่จำเป็นจะต้องเป็นพฤติกรรมของผู้ใช้เท่านั้น ยังอาจหมายถึง พารามิเตอร์ของโฮสต์และระบบเครือข่ายด้วย ซึ่งสามารถถูกตรวจสอบเพื่อให้มีข้อมูลมากขึ้น

2.3.2 signature based intrusion detection ระบบพยายามที่จะค้นหาหลักฐานในรูปแบบของข้อมูลที่ตรงกับ signature ของการบุกรุกที่เป็นที่รู้จัก หรือพฤติกรรมที่น่าสงสัย signature เหล่านี้ถูกสร้างขึ้นภายหลังจากที่มีการค้นพบการบุกรุกรูปแบบใหม่ สังเกตว่า แม้ว่า signature เหล่านี้สามารถแทน

รูปแบบพฤติกรรมที่น่าสงสัย และยังไม่ได้รับการพิสูจน์หลักฐานให้เป็น รูปแบบการบุกรุกที่เป็นที่รู้จักกันได้ แต่มันยังแตกต่างจากการตรวจสอบแบบ Anomaly ดังที่ได้กล่าวมาแล้วข้างต้น เนื่องจากในระบบนี้จะไม่มีส่วนที่ทำการเรียนรู้พฤติกรรมด้วยตัวเอง

2.3.3 specification based intrusion detection เป็นรูปแบบพิเศษของ signature base intrusion detection ซึ่งระบบสามารถกำหนด signature ให้กับตัวเองได้ ซึ่ง signature เหล่านี้ไม่ได้แทนพฤติกรรมของการบุกรุก แต่แทนพฤติกรรมที่ถูกต้อง ทุก ๆ การใช้งานที่ต่างไปจากรูปแบบเดิมจะถูกให้สัญญาณเพื่อบ่งชี้ความพยายามที่จะบุกรุก

จากรูปแบบข้างต้นลดการแบ่งรูปแบบเป็น 2 รูปแบบเท่านั้น

- **Anomaly based detection** เหมือนที่ได้กล่าวไปข้างต้น
- **Policy based detection** เป็นรูปแบบของการตรวจจับบนพื้นฐานของนโยบายทางด้านความปลอดภัยภายนอกระบบ ในกรณีที่นโยบายถูกระบุเจาะจงถึงสิ่งที่อนุญาตให้ทำได้ หลักการตรวจจับก็จะเหมือนกันกับ signature based detection แต่ในกรณีที่นโยบายเหล่านี้ระบุถึงสิ่งที่ไม่ได้ให้กระทำ หลักการตรวจจับก็จะกลายเป็นแบบ specification based

2.4 การแบ่งประเภทของระบบตรวจจับการบุกรุก

ระบบตรวจจับการบุกรุกที่ได้สำรวจและรวบรวมมานั้นสามารถแบ่งประเภทได้ตามคุณลักษณะหลาย ๆ อย่าง แตกต่างกันออกไป ที่เป็นที่ชัดเจนก็คือการแบ่งตามหลักในการตรวจจับดังที่ได้กล่าวมา

2.4.1 Anomaly detection (การตรวจจับสิ่งผิดปกติ) ระบบจะแสดงการโต้ตอบพฤติกรรมที่ผิดปกติ โดยให้ความหมายคือ ประวัติของการใช้งานบางส่วนที่ถูกเฝ้าสังเกตเป็นพฤติกรรมก่อนหน้า หรือโดยกำหนดเป็น profile ก่อนหน้า (profile หรือพฤติกรรมที่ได้กล่าวมานั้นสามารถหมายถึง ผู้ใช้ โฮสต์ เครือข่าย และอื่น ๆ) เพื่อที่จะทำให้รูปแบบการตรวจจับสิ่งผิดปกติมีลักษณะแตกต่างจากแบบตรวจจับตามนโยบาย นั่นคือระบบต้องมีการเรียนรู้จากตัวอย่างในอดีตโดยอัตโนมัติ ถ้าผู้ใช้งานมีการใช้งานที่เหมือนเดิมจากข้อมูลในอดีตและประมวลผลความรู้ที่ได้ไปเป็นกฎในระบบผู้เชี่ยวชาญ

2.4.2 Policy base detection (การตรวจจับตามนโยบาย) ระบบจะแสดงการโต้ตอบเมื่อมีการฝ่าฝืนนโยบายบางส่วน ซึ่งนโยบายที่ว่านี้เป็นไปได้ทั้งแบบ อนุญาตให้ทำ (default permit) หรือ ละเว้นการกระทำ (default deny) นั่นคือ SSO จะต้องเลือกระบุ signature ต่าง ๆ ที่อธิบายถึง พฤติกรรมการใช้งานที่ผิด หรือจะระบุเป็น การใช้งานที่ผิดปกติหรือเป็นไปในทางที่ดีต่อระบบถ้ามีการเปลี่ยนแปลงจากสภาวะที่เป็นปกติจะถูกเฝ้ามองว่าเป็นการพยายามที่จะบุกรุกโดยระบบการตรวจจับการบุกรุก

2.4.3 Hybrid (แบบผสม) ดังที่ได้กล่าวไว้แล้ว นักวิจัยส่วนใหญ่จะเชื่อว่ารูปแบบทั้งสองนั้นควรจะนำมาให้รวมกันในการออกแบบระบบตรวจจับการบุกรุก เพื่อที่จะได้รับข้อดีทั้งสองแบบ

นอกจากนั้นยังมีการแบ่งกลุ่มระหว่างระบบต่าง ๆ จากแต่ละรูปแบบของระบบตรวจจับการบุกรุกในส่วนของข้อมูลออดิต จากการศึกษาอย่างละเอียดทำให้แบ่งกลุ่มได้อย่างชัดเจน ดังนี้

Time of detection (ช่วงเวลาในการตรวจจับ) สามารถจำแนกได้เป็นกลุ่มใหญ่ ๆ คือแบบมีการตรวจจับในเวลาจริง (real-time) หรือ ใกล้เวลาจริง (near real-time) และแบบที่มีการประมวลผลข้อมูลแบบออดิตเลื่อนออกไป รูปแบบอันหลังนี้จะทำให้การตรวจจับล่าช้าออกไป ระบบที่ได้ทำการสำรวจนั้นอยู่ในประเภท real-time จะต้องสามารถทำงานในแบบ off-line กับข้อมูลการออดิตในอดีตโดยไม่มีข้อยกเว้นใด ๆ ซึ่งจะทำให้การทำงานในส่วนการตรวจสอบง่ายขึ้นมาก เช่นเดียวกันกับการพัฒนาระบบ แต่ก็แน่นอนที่บางครั้งจะเป็นประโยชน์กับความสามารถของระบบในการทำงานกับข้อมูลที่ได้รับไว้ก่อนแล้ว สร้างกรณีวิเคราะห์ทางด้านความปลอดภัยในอดีต

Granularity of data-processing (ส่วนย่อยของการประมวลผลข้อมูล) เป็นการแบ่งประเภทโดยเทียบความแตกต่างของระบบในการประมวลผลข้อมูลติดต่อกันไปเรื่อย ๆ กับระบบซึ่งมีการประมวลผลข้อมูลแบบแบตช์ (batch) ขณะอยู่ในช่วงเวลาตามปกตินั้น การแบ่งประเภทนี้นั้นจะ เชื่อมโยงกันแบบ ช่วงเวลาในการตรวจจับ (Time of Detection) แต่ให้สังเกตทั้งสองจะไม่มีส่วนที่เหลื่อมล้ำกัน เนื่องจากระบบจะสามารถประมวลผลข้อมูลอย่างต่อเนื่องด้วยเวลาที่ต้องเลื่อนไปนิดหน่อย (บางกรณี) หรือประมวลผลข้อมูลแบบแบตช์ (ขนาดเล็ก) ในแบบ real-time

Source of audit data (แหล่งที่มาของข้อมูลการออดิต) ข้อมูลการออดิตของระบบที่ได้สำรวจและรวบรวมนั้นมาจาก 2 แหล่งใหญ่คือ ข้อมูลเครือข่าย (network data) ในแบบธรรมดาก็จะอ่านเอาจากข้อมูลที่กระจายออกมาจากเครือข่าย (Ethernet) และจากล็อกไฟล์ทางด้านความปลอดภัยจากบน โฮสต์ (host base) ซึ่งล็อกไฟล์ที่อยู่บนโฮสต์นี้จะรวมล็อกไฟล์ของเคอร์เนล (kernel) ในระบบปฏิบัติการ การใช้งานโปรแกรม, อุปกรณ์เครือข่าย (เช่น router และ firewall) และอื่น ๆ

Response to detected intrusion (การตอบสนองต่อการบุกรุกที่ถูกตรวจจับ) พาสซีฟ (passive) กับ แอ็กชัน (action) ระบบที่เป็นแบบพาสซีฟนั้นจะมีการตอบสนองโดยการแจ้งไปยังผู้ที่มีอำนาจเกี่ยวข้องเท่านั้น ระบบตจะไม่มีการแก้ไขความเสียหายที่เกิดขึ้นเอง และไม่กระทำการใด ๆ ไปหายังผู้โจมตี ส่วนระบบแบบแอ็กชันนั้นสามารถแบ่งย่อยได้เป็น 2 ลักษณะ คือ

- พวกที่ควบคุมการปฏิบัติไปยังระบบที่ถูกโจมตี คือจะมีการแก้ไขสภาวะของระบบที่ถูกทำการโจมตีเพื่อขัดขวางหรือบรรเทาผลกระทบที่เกิดจากการโจมตี การควบคุมเช่นนี้อาจจะเป็นในรูปแบบของการตัดการเชื่อมต่อเครือข่าย , การเพิ่มการเก็บล็อกไฟล์เกี่ยวกับความปลอดภัย, หยุดการทำงานของโปรเซส
- พวกที่ควบคุมการปฏิบัติไปยังระบบที่ทำการโจมตี คือ มีการโจมตีกลับไปยังผู้(ระบบ)ที่ทำการโจมตีเพื่อที่จะพยายามที่จะกำจัดการกระทำของผู้บุกรุก เนื่องจากในรูปแบบนี้เป็นอะไรที่ใช้ในแง่ของกฎหมายได้ยาก ซึ่งไม่ได้ให้ความสนใจมาเท่าไร ในระบบที่ไม่ใช่ของรัฐบาลหรือเกี่ยวข้องกับทางการทหาร

Locus of data-processing (ตำแหน่งการประมวลผลข้อมูล) ข้อมูลการออดิตสามารถถูกประมวลผลในส่วนกลาง (central) โดยต้องไม่คำนึงถึงว่าข้อมูลเกิดมาจากที่ใด -อาจจะมาจากแหล่งเดียวกัน —หรืออาจถูกรวบรวมและเรียบเรียง (ประมวลผล) มาจากแหล่งที่ต่าง ๆ กันออกไปในรูปแบบกระจาย (distributed)

Locus of data-collection (ตำแหน่งของการรวบรวมข้อมูล) ข้อมูลออดิตสำหรับการประมวลผลหรือการตรวจจับ ถูกรวบรวมมาจากหลาย ๆ แหล่งข้อมูลต่าง ๆ กัน ในรูปแบบกระจาย (distribute) หรือรูปแบบของส่วนกลาง ซึ่งรวบรวมข้อมูลจากจุดเดียว

Security (ความปลอดภัย) ความสามารถที่จะต่อต้านการโจมตีต่อระบบตรวจจับการบุกรุกของตัวมันเอง ซึ่งนี่เป็นการศึกษาส่วนเล็กเท่านั้น การแบ่งประเภทที่ง่ายคือ ความปลอดภัย สูง – ต่ำ

Degree of interoperability (ระดับของความสามารถในการทำงานที่มีเหตุการณ์เกิดขึ้นพร้อมกัน) คือ ระดับความสามารถที่ระบบจะทำงานพร้อม ๆ กันกับระบบตรวจจับการบุกรุกระบบอื่น ๆ เช่น การรับข้อมูลออดิตจากแหล่งอื่น ๆ เป็นต้น หมายเหตุว่าไม่เหมือนกันกับความสามารถในการทำงานได้ในหลายแพลตฟอร์มที่ต่างกัน

การแบ่งประเภทดังที่กล่าวมาแล้วทั้งหมดนี้ไม่สามารถแสดงการแบ่งแยกระบบที่มีอยู่จริงได้ อย่างไรก็ตามก็ยังเชื่อว่าหลาย ๆ ระบบที่ได้รวบรวมมานั้นเพียงพอที่จะแสดงการแบ่งแยกให้เห็น

2.5 การแยกประเภทของระบบตรวจจับผู้บุกรุกที่สำรวจมา (A classification of the surveyed systems)

เมื่อใช้การแบ่งตามรูปแบบข้างต้นนั้นกับระบบที่ได้สำรวจมา จะแบ่งประเภทต่าง ๆ ได้ ดังแสดงในตาราง 2.1

2.6 ข้อดีข้อเสียของระบบตรวจจับผู้บุกรุกในรูปแบบต่างๆ

การค้นคว้าก่อนหน้านี้ได้ทำให้เห็นคุณลักษณะหลาย ๆ อย่างของ 2 รูปแบบหลัก ๆ ในการตรวจจับการบุกรุก คือ Anomaly based และ Signature based ซึ่งสามารถสรุปข้อดีข้อเสียได้ดังนี้

Anomaly detection

ข้อดี การทำงานไม่ต้องการคอนฟิกตัวระบบ มันจะเรียนรู้พฤติกรรมทั่ว ๆ ไป โดยอัตโนมัติ ทำให้สามารถปล่อยให้ทำงานได้เองโดยไม่ต้องมาคอยดู เนื่องจากที่มันไม่รู้ความแตกต่างทำให้มีบางคนบอกว่าจะทำให้เกิดความเสียหายในแง่ที่ว่าจะทำอย่างไรให้เปิดเผยหรือแสดงการบุกรุก ที่เกิดขึ้น ซึ่งมีโอกาสเป็นไปได้ที่จะจับการบุกรุกแบบแปลกใหม่ได้ เช่นเดียวกันกับการบุกรุกที่รู้จักกันมาแล้ว

ข้อด้อย จากคำจำกัดความของรูปแบบนี้ จะเป็นการทำงานสังเกตพฤติกรรมที่ผิดปกติไป โดยที่ไม่ได้เน้นลักษณะพฤติกรรมที่มีการหลบซ่อนในตัวมันเอง ซึ่งอาจเป็นปัญหาเมื่อรูปแบบพฤติกรรมทั้ง 2 ชนิดไม่มีส่วนที่ซ้อนกัน ระบบนั้นจะเรียนรู้ที่จะยอมรับพฤติกรรมที่เป็นอันตรายเหมือนเป็นรูปแบบปกติของผู้ใช้ที่เจาะจงลงไป โดยจะเปลี่ยนแปลงรูปแบบพฤติกรรมไปอย่างช้า ๆ ตามเวลา ทำให้ไม่สามารถค้นหาการเปลี่ยนแปลงไปจากธรรมดาได้ เมื่อผู้ใช้คนนั้นสิ้นสุดการโจมตีแล้ว ในการ update profile ของผู้ใช้และการเทียบเคียงพฤติกรรมที่กำลังทำอยู่ในช่วงเวลานี้เข้ากับ profile นั้นจะเป็นภาระที่ใช้ในการประมวลผลข้อมูลอย่างเต็มที่ซึ่งจะทำให้สิ้นเปลืองทรัพยากรการประมวลผลอย่างหนัก

Name of system	Publ. Year	Detection principle	Time of detection	Granularity	Audit source	Type of response	Data-processing	Data-collection	Security	Inter - oper.
Haystack[51]	1988	Hybrid	Non-real	Batch	Host	Passive	Centralised	Centralised	Low	Low
MIDAS[50]	1988	Hybrid	Real	Continuous	Host	Passive	Centralised	Centralised	Low	Low
LDES[41]	1988	Anomaly[a]	Real	Continuous	Host	Passive	Centralised	Distributed	Low	Low
W&S	1989	Anomaly	Real	Continuous	Host	Passive	Centralised	Centralised	Low	Low[b]
Comp-Watch[11]	1990	Anomaly[c]	Non-real	Batch	Host	Passive	Centralised	Centralised	Low	Low
NSM[21]	1990	hybrid[d]	Real	Continuous	network[e]	Passive	Centralised	Centralised[h]	Low	Low[g]
NADIR[25]	1991	policy	Non-real	Continuous	host[h]	Passive	Centralised	Distributed	Low	Low
Hyperview[7]	1992	hybrid	Real	Continuous	host	Passive	Centralised	Centralised	Low	Low
DIDS[52]	1992	hybrid	Real	Continuous	both[i]	Passive	Distributed	Distributed	Low	Low[j]
ASAX[18]	1992	policy	real[k]	continuous[l]	host	Passive	Centralised	Centralised	Low	Higher[m]
USTAT[24]	1993	policy	real	continuous	host	Passive	Centralised	Centralised	Low	Low[n]
DPEM[30]	1994	policy[o]	real	batch	host	Passive	Distributed	Distributed	Low	Low
IDIOT[34]	1994	policy	real[p]	continuous	host	Passive	Centralised	Centralised	Low	higher
NIDES[1]	1995	hybrid	real[q]	continuous	host[r]	Passive	Centralised	Distributed	Low[s]	higher[i]
GrIDS[53]	1996	hybrid[u]	non-real	batch	both[v]	Passive	Distributed	Distributed	Low	Low
CSM[58]	1996	policy	real	continuous	host	active[w]	Distributed	Distributed	Low	Low
Janus[17]	1996	policy	real	continuous	host	active[x]	Centralised	Centralised	Low	Low
JiNao[15]	1997	hybrid	real	batch	"host"[y]	Passive	Distributed	Distributed	Low	Low
EMERALD[47]	1997	hybrid	real	continuous	both	Active	Distributed	Distributed	Monderate	high
Bro[40]	1998	policy	real	continuous	network	Passive	Centralised	Centralised	higher	Low

ตารางที่ 2.1 การแยกประเภทของระบบตรวจจับผู้บุกรุกที่สำรวจมา

Signature detection

ข้อดี ระบบแบบนี้จะ "รู้" รูปแบบจริง ๆ ของทั้งพฤติกรรมที่เป็นที่น่าสงสัยหรือวิธีแสดงว่าพฤติกรรมนั้นเป็นปกติ มีการประมวลผลข้อมูลออดิตที่ง่ายและมีประสิทธิภาพ และมีอัตราส่วนการทำงานที่ผิดพลาดต่ำอีกด้วย

ข้อด้อย การระบุ signature ของการบุกรุกนั้นต้องผ่านการตรวจสอบอย่างละเอียดและเป็นงานที่ต้องใช้เวลา มันไม่ได้ใช้อะไรที่การทำงานปกติของระบบควรจะทำ การขึ้นอยู่กับซึ่งปัจจัยข้างต้นของการระบุ signature นั้นจะทำให้ไม่สามารถตรวจจับเหตุการณ์ที่เปลี่ยนแปลงไปของการบุกรุกได้ แน่นอนวิธีการนี้มีข้อจำกัด มันไม่สามารถจับการบุกรุกที่แปลกใหม่ได้ โดยเฉพาะอย่างยิ่งการบุรุกประเภทใหม่ที่เกิดขึ้น

2.7 แนวโน้มและสิ่งคงที่ในการวิจัยเกี่ยวกับระบบตรวจจับการบุกรุก

2.7.1 แนวโน้ม

จากโฮสต์ไปสู่เครือข่าย (From host to network)

การเปลี่ยนจากการตรวจจับการบุกรุกบนโฮสต์ไปอยู่บนเครือข่าย เป็นเหมือนกับการเปลี่ยนระบบจากระบบหลายผู้ใช้เดียว (single multi-user system) ไปเป็นเครือข่ายสถานีงาน (network of workstation) แต่อย่างไรก็ตามเทคโนโลยีด้านเครือข่ายในปัจจุบัน (เครือข่ายสวิตช์ , ความเร็วที่เพิ่มขึ้นในการติดต่อสื่อสารบนเครือข่าย) เป็นสิ่งที่ทำให้การเฝ้าดูเครือข่ายเพื่อหาข้อมูลออดิตเป็นไปได้ลำบากขึ้น ยิ่งไปกว่านั้นยังมีปัญหาที่ว่าจะทำอย่างไรกับข้อมูลที่ถูกเข้ารหัสบนเครือข่าย ยังคงมีปัญหาให้แก้ไขอยู่เสมอ ๆ รูปแบบต่อ ๆ ไปก็น่าจะเป็นแบบผสม

จากส่วนกลางสู่การกระจาย (From centralise to distrbuted)

การเปลี่ยนแปลงอีกอันที่เกี่ยวพันกับการเปลี่ยนจากระบบหลายผู้ใช้เดียวไปยังเครือข่ายสถานีงานก็คือ การเปลี่ยนจากการตรวจจับแบบส่วนกลางไปเป็นการตรวจจับการบุกรุกการกระจาย

ในกรณีการเก็บรวบรวมข้อมูล (data collection) นั้นเราได้เห็นแนวโน้มที่ชัดเจนว่าการเก็บล็อกเกี่ยวกับความปลอดภัยบนโฮสต์นั้นต้องกระจายไปเพื่อเก็บล็อกสิ่งที่เกิดขึ้นในเครื่องเวิร์กสเตชันที่อยู่บนเครือข่าย ในกรณีข้อมูลของเครือข่าย ก็มีความคิดที่ให้มีการเฝ้าดูการใช้งานของเครื่องเวิร์กสเตชันที่อยู่บนเครือข่ายจากจุดรวมของเครือข่าย และที่จริงมีแต่ระบบ 2 ระบบจากที่ได้ไปสำรวจมาที่เป็นการเฝ้าดูการใช้งานเครือข่ายอย่างแท้ ๆ ที่ใช้รูปแบบนี้ อย่างไรก็ตามระบบที่เหลือนั้นมีการเฝ้าดูการใช้งานทางล็อกไฟล์บนโฮสต์ด้วย บนจุดเชื่อมต่อเครือข่ายด้วย ด้วยเหตุนี้จึงไม่ได้เกิดจุดสนใจที่เรื่องนี้อย่าง แท้จริง

เรื่องของการประมวลผลข้อมูล (data processing) แนวโน้มที่จะไปทางการกระจาย นั้นดูเหมือนยังล้าหลังอยู่ แต่เนื่องจากมันน่าจะเป็นเพียงวิธีที่จะแก้ปัญหาในเรื่องการกำหนดขนาดหรือสัดส่วนของระบบตรวจจับการบุกรุก จึงเป็นรูปแบบที่น่าจะพิจารณาไว้และในปัจจุบันก็ได้เริ่มมีการพยายามสร้างขึ้นมาแล้ว อีกสิ่งที่น่าสนใจคือให้สังเกตว่าแม้จะมีการประมวลผลแบบกระจายเกิดขึ้นจริง แต่ความเชื่อถือของ SSO ต่อแบบส่วนกลางยังคงมีต่อไป

ในเรื่องของความสามารถที่ระบบจะทำงานพร้อม ๆ กันกับระบบตรวจจับการบุกรุกระบบอื่น (Towards interoperability)

ในขณะที่ระบบพัฒนามาแล้วส่วนใหญ่นั้นจะยึดติดอยู่กับแฟลตฟอร์มที่ระบุเฉพาะไปเลย แนวโน้มในปัจจุบันเคลื่อนสู่รูปแบบที่จะทำให้ระบบตรวจจับผู้บุกรุกทำงานร่วมกันได้มากขึ้นเรื่อย ๆ ผลดีที่เห็นได้ก็คือ การลบล้างการยึดติดของวิธีการที่แตกต่างของผู้จำหน่าย ทำให้เกิดประโยชน์กับจุดแข็งและจุดอ่อนของระบบแต่ละแบบ และยังสามารถที่จะทำให้ระบบตรวจจับการบุกรุกทำงานได้ในสภาวะแวดล้อมที่มีลักษณะแตกต่างกัน

ความต่อต้านต่อการโจมตี (More resistant to attack)

ความต่อต้านต่อการโจมตีของระบบตรวจจับการบุกรุกเองนั้นเป็นหัวข้อที่ ซึ่งงานวิจัยในก่อนนั้นไม่ได้อยู่ในช่วงของการใช้งาน ดังนั้นแนวโน้มจึงชัดเจนที่ระบบนั้นจะต้องสามารถต่อต้านการโจมตีเช่นเดียวกับระบบเป้าตรวจจับ ระบบอีกระบบหนึ่งในปัจจุบันที่พยามยามที่จะอยู่ในแนวโน้มนี้

อย่างไรก็ดีมีการศึกษาเล็ก ๆ น้อย ๆ ของลักษณะทั่วไปของการโจมตีระบบตรวจจับผู้บุกรุกที่สามารถต้านทานได้จริง ๆ ดังเช่น เอกสารทางวิชาการฉบับหนึ่งในปัจจุบันที่มีประเด็นเกี่ยวกับการพิจารณาการหลบหลีกการตรวจจับ [49] แต่ในฉบับอื่นๆ ยังไม่มีการพูดถึงมากนัก

สิ่งที่คงที่ (Constants)

การผสมระหว่างแบบ Anomaly และ Policy

ดูราวกับว่ายังเป็นข้อตกลงทั่ว ๆ ไปอยู่ในการสร้างระบบตรวจจับการบุกรุกที่มีประสิทธิภาพ นั้นคือระบบจะต้องใช้ทั้งวิธี anomaly base และ policy base

เป็นที่น่าสนใจที่จะสังเกตความสัมพันธ์ของการเปลี่ยนแปลง — ในการรวบรวมงานวิจัยใหม่ ๆ เน้นไปยังการตรวจสอบจากนโยบายอยู่บ่อยครั้ง และเป็นการสิ้นเปลืองสิ่งต่าง ๆ ในการตรวจสอบแบบการใช้ที่ผิดปกติ เหตุผลที่เหมาะสมน่าจะเป็นดังนี้ ในเมื่อเป็นการยากที่จะเอามาทดสอบแล้วลงความเห็นอะไรที่แน่ชัดที่ครอบคลุมเทคนิคแบบ anomaly base และต้องพูดว่าการพิจารณาส่วนครอบคลุมยังค่อนข้างจะขาดบางส่วนไปในรูปแบบที่ทำงานบน policy base ด้วยเหมือนกัน

การตรวจจับขณะช่วงเวลาจริง (Real-time Detection)

จากการวิจัยช่วงก่อน ๆ ระบบจะทำการตรวจจับในช่วงที่ไม่ใช่เวลาจริง (non-real time) ซึ่งจะทำให้มองเห็นได้เลยว่าเป็นความไม่สมบูรณ์ของระบบ ซึ่งถูกบังคับโดยข้อจำกัดทางเทคโนโลยีในสมัยนั้น

ในขณะที่ชัดเจนเลยว่าการตรวจจับในช่วงเวลาจริงเป็นคุณสมบัติที่ควรจะต้องมี แต่ก็ไม่ควรจะตัดส่วนที่เป็นประโยชน์ของการตรวจจับอยู่ในช่วงเวลาจริงไปเสียหมด ยังคงมีหลาย ๆ กรณีที่หลังจากที่มีการประเมินสถานการณ์จากข้อเท็จจริงผ่านไปแล้ว ยังสามารถที่จะแสดงเหตุการณ์ที่เกิดขึ้นได้ ในบางโอกาสก็เหมาะสมกว่าที่จะมีการส่งสัญญาณเตือนแบบทันทีทันใดที่บางอย่างอาจผิดพลาดได้

ดังเช่น สถานการณ์ที่เกิดขึ้นในกรณีที่ต้องใช้กฎข้อบังคับที่ซึ่งความถูกต้องและการสามารถที่จะสืบได้ว่ามาจากไหนนั้นในเหตุการณ์นั้นมีความสำคัญมากกว่าการทำงานในเวลาจริง มีอีกกรณีที่คล้ายกัน

คือเมื่อนโยบายทางด้านความปลอดภัยขององค์กร ดังเช่น เหตุฉุกเฉินทางด้านการแพทย์ เจ้าหน้าที่ทางการแพทย์ต้องมีการจำกัดการเข้าถึงให้เร็วที่สุดเท่าที่เป็นไปได้ การรักษาความปลอดภัยก็เป็นผลดีที่สะท้อนที่มีความเกี่ยวข้องในภายหลัง

จากที่ได้กล่าวไว้แล้วว่ายังมีความสัมพันธ์ระหว่างการแบ่งกลุ่ม ระหว่างช่วงเวลาในการตรวจจับ (time of detection) และ ส่วนย่อยของการประมวลผลข้อมูล สามารถเข้าใจได้จากตารางที่ 2.1 ที่ว่าการตรวจจับในเวลาจริงที่เกี่ยวพันกับการประมวลผลข้อมูลที่ต่อเนื่อง และการตรวจจับแบบไม่ใช่เวลาจริงเกี่ยวพันกับการประมวลผลข้อมูลแบบแบตช์ ซึ่งยังมีข้อยกเว้นอีกเล็กน้อยสำหรับความสัมพันธ์ดังกล่าวแต่ที่แน่นอน คือมันจะไม่มีการเลื่อมล้ำกันอย่างสมบูรณ์

การตอบสนองแบบแอ็กทีฟที่น้อย (Few active responses)

มีการโต้เถียงบางประการทีอนุญาตให้ระบบตอบสนองแบบแอ็กทีฟมากขึ้น ยกตัวอย่างเช่นการตัดการเชื่อมต่อที่ดูเหมือนว่าเป็นสาเหตุของการโจมตี ตามความคิดเห็นแล้วระบบแอ็กทีฟน่าจะเป็นที่นิยม แต่ในแง่ของการวิจัยแล้ว ในส่วนนี้ยังไม่จริงจังมาเท่าไร ความยากในการพิจารณาปัญหาการตรวจจับที่แม่นยำถูกต้อง . ความเป็นไปได้ที่จะทำให้ระบบถูกโจมตีแบบการปฏิเสธการให้บริการ , สิ่งที่หรือขอบเขตที่ระบบจะต้องรับผิดชอบ ยังคงเป็นปัญหาที่จะต้องได้รับการแก้ไข ปรับปรุง เพื่อให้ระบบตรวจจับการบุกรุกมีความหน้าเชื่อถือในการจะตอบสนองเหตุการณ์ต่าง ๆ โดยตัวระบบเองได้เลย

การใช้งานทรัพยากร (The consumption of resource)

เนื่องด้วยทั้งคอมพิวเตอร์และเครือข่ายต่างก็มีประสิทธิภาพและความเร็วที่เพิ่มมากขึ้น ทำให้เราสามารถประมวลผลข้อมูลการออดิตได้มากขึ้นใน 1 หน่วยเวลา แต่ทว่าทั้งคอมพิวเตอร์และเครือข่ายต่างก็มีการสร้างข้อมูลการออดิตที่เพิ่มมากขึ้น (บางประเภทของการออดิต) ในอัตราที่ไม่แพ้กันเลย เนื่องจากเหตุนี้อัตราส่วนทั้งหมด (total ratio) ของการใช้งานทรัพยากรที่มี จะอยู่ในลักษณะที่ว่า ถ้าไม่คงที่ อย่างน้อยก็จะไม่ลดลงไปพอที่จะทำความเร็วเพิ่มขึ้น ในทางตรงกันข้ามแล้วความเร็วในการติดต่อสื่อสารบนเครือข่าย ดูเหมือนจะเป็นสิ่งเดียวที่เป็นอุปสรรคที่กลุ่มการคนคว้าวิจัยนั้นเห็นว่าไม่มีทางที่จะทำให้ชัดเจนได้ทีเดียว

การศึกษาในส่วนขอบเขตที่ครอบคลุม (Little study coverage)

ยังคงมีการขาดการศึกษาในส่วน ขอบเขตที่ครอบคลุม วงความรู้เกี่ยวกับการบุกรุกระบบ ปัญหานี้อาจทำให้เกิดการผิดพลาดในการแบ่งแยกการทำกิจกรรมที่ถูกต้องไปเป็นการบุกรุกระบบ ซึ่งเรียกว่าการเตือนที่ผิดพลาด (false positive หรือ false alarm) การแบ่งแยกที่ผิดพลาดนี้นำไปสู่ปัญหา ต่าง ๆ อีกมากมาย ส่วนคำว่า "ขอบเขตที่ครอบคลุม" นั้นหมายถึงในส่วนของความน่าเชื่อถือ ซึ่งในที่นี้จะหมายถึงอัตราส่วนชองการแบ่งแยกการบุกรุกที่ถูกต้อง (true positive) ต่อจำนวนการบุกรุกที่ถูกแบ่งแยกผิดพลาด (false negatives) บวกกับจำนวนของการแบ่งแยกที่ถูกต้องจะได้เป็นเศษส่วนของการบุกรุกที่สามารถตรวจจับได้ แม้ว่ามันจะเป็นสิ่งที่ช่วยวัดประสิทธิภาพของระบบตรวจจับการบุกรุก แต่มีอ้างอิงส่วนนี้ลงไปในการตีพิมพ์น้อยมาก

ลักษณะของการบุกรุกทางด้านความปลอดภัยในคอมพิวเตอร์ จากมุมมองของการตรวจจับการบุกรุก (The nature of computer security intrusions, form an intrusion detection perspective)

ไม่มีการประสานงานกันในการศึกษาในส่วนของ ขอบเขตที่ครอบคลุม หรืออาจจะพูดได้ว่าขาดการศึกษาในส่วนของลักษณะการบุกรุกระบบ ซึ่งน่าจะสามารถแยกเป็นประเภทต่าง ๆ และลักษณะของการบุกรุกแบบใดที่ระบบตรวจจับการบุกรุกสามารถต่อต้านได้

บทบาทหน้าที่และความสามารถของ SSO (The role, and capabilities of the SSO)

ความเชื่อมั่นความไว้ใจใน SSO บางคน ที่มีอำนาจการตัดสินใจอย่างเด็ดขาด และการตอบสนองไปยังการบุกรุก บทบาทที่เฉพาะเจาะจงลงไปของ SSO ยังไม่ได้มีการศึกษาให้รอบคอบเท่าไร เล่น ผลลัพธ์อย่างไรที่ควรจะถูกรายงานไปยัง SSO , ความสามารถในการตอบสนอง เป็นต้น แน่นอนถ้าระบบนั้นผิดพลาดในประเด็นเรื่องของจำนวนการเตือนที่ผิดพลาด (false alarm) ในเทอมของความสัมพันธ์แล้วอาจจะมีการเตือนที่ผิดพลาดเป็น 0.5 % แต่ใช้เทอมของการใช้งานจริง 0.5 % นั้นอาจจะเป็นการเตือนที่ผิดพลาดถึง 5000 ครั้ง ดังนั้นซึ่งเป็นความลำบากของการกำหนดหน้าที่ของ SSO ในมุมมองดังกล่าว

บทที่ 3
การบุกรุกและเครื่องมือในการบุกรุก

3.1 รูปแบบของผู้บุกรุก

ในวงความรู้ของการตรวจจับการบุกรุก ได้มีการค้นคว้าศึกษามาเป็นเวลานานกว่า 18 ปีแล้วได้แบ่งรูปแบบของผู้โจมตี (Attackers) ซึ่งเป็นไปได้ของระบบคอมพิวเตอร์ออกเป็น 4 กลุ่มคือ

3.1.1 External penetrator (ผู้บุกรุกจากภายนอกองค์กร) ผู้บุกรุกจากภายนอกองค์กรจะได้รับการเข้ามาถึงยังระบบคอมพิวเตอร์ ซึ่งผู้นั้นไม่ใช่ผู้ที่ได้รับสิทธิ์อย่างถูกต้องในการใช้งาน Anderson ใช้คำจำกัดความรวมถึงผู้ใช้ที่มีลักษณะดังตัวอย่างนี้ด้วย คือ ลูกจ้างขององค์กรที่พวกเขาสามารถเข้าถึง (เดินเข้าไป) ในตึกและอาคารต่าง ๆ ที่เป็นที่ตั้งของระบบคอมพิวเตอร์ โดยที่พวกเขาไม่ได้รับอนุญาตให้ใช้ทรัพยากรนั้นได้

3.1.2 Masquerader (ผู้ใช้ตัวปลอม) ผู้ใช้ตัวปลอมนั้นหมายถึง ผู้ใช้ที่ได้รับอนุญาตให้เข้าถึงระบบ ผู้ใช้ตัวปลอมนี้อาจจะเป็นผู้บุรุกจากภายนอกองค์กร (External penetrator) หรือผู้ที่ได้รับอนุญาตให้ใช้ระบบที่พยายามจะใช้สิทธิ์การใช้งานของผู้ใช้คนอื่น ๆ ผลก็คือผู้บุกรุกจะกลายเป็นผู้ใช้คนนั้นในส่วนของระบบคอมพิวเตอร์ที่เกี่ยวข้องด้วย ซึ่งรูปแบบของการบุกรุกแบบนี้เป็นที่น่าสนใจเนื่องจากไม่มีวิธีที่แน่นอนที่จะแยกแยะความแตกต่างระหว่างผู้ใช้ที่มีสิทธิ์ถูกต้องกับผู้ใช้ตัวปลอม

3.1.3 Misfeasor (ผู้ละเมิด) ผู้ใช้ที่ได้รับสิทธิ์อย่างถูกต้องในการเข้าถึงทรัพยากรของระบบนั้นสามารถเป็นผู้ละเมิดได้ นั่นคือถึงแม้ว่าเขาคนนั้นจะได้สิทธิ์อย่างถูกต้องในการเข้าถึงข้อมูล แต่ใช้ไปในทางที่ไม่ถูกต้อง เกินกว่าสิทธิ์ที่ได้รับ มีการฝ่าฝืนนโยบายทางด้านความปลอดภัยของระบบที่ได้กำหนดไว้

3.1.4 Clandestine user (ผู้ใช้ลึกลับ) ผู้ใช้ลึกลับนี้จะมีงานโดยที่สามารถหลบหลีกกลไกการตรวจสอบแบบธรรมดาได้ (normal auditing meschanisms) ซึ่งในบางทีการเข้าถึงระบบด้วยสิทธิ์ของผู้ดูแลระบบ (Supervisory)

ในขณะที่รูปแบบปัญหาที่เกิดขึ้นไม่ได้ถูกเปิดกว้างในวงความรู้เกี่ยวกับการตรวจจับผู้บุกรุก ผู้ใช้ตัวปลอมเป็นรูปแบบที่น่าสนใจ ด้วยการให้ความหมายของการใช้งานระบบเป็นพิเศษโดยผู้ใช้ที่ไม่ได้รับสิทธิ์ ซึ่งในฐานะการใช้งานแบบนี้แล้วมันน่าจะมีความเป็นไปได้ที่จะตรวจจับกรณีการใช้งานจากการวิเคราะห์บันทึการใช้ที่มีลักษณะดังนี้

- มีการใช้งานเกินกว่าช่วงเวลาปกติ
- ความถี่ของการใช้งานที่ผิดปกติ
- จำนวนการใช้หรืออ้างอิงข้อมูลที่ผิดปกติ
- รูปแบบการอ้างอิงการใช้โปรแกรมหรือข้อมูลที่ผิดปกติ

3.2 ขั้นตอนการบุกรุก

3.2.1 การแกะรอย

การแกะรอยเป็นการรวบรวมข้อมูลของเป้าหมายที่ต้องการ โดยเฉพาะช่องโหว่ที่มีอยู่บนระบบเน็ทเวิร์ค อันจะนำมาซึ่ง โพรไฟล์ (profile) ขององค์กรเป้าหมายที่เชื่อมต่ออยู่กับอินเตอร์เน็ต ทั้งในส่วนของระบบเน็ตเวิร์กภายในหรืออินทราเน็ต และเอ็กซ์ทราเน็ต รวมทั้งการบริการเชื่อมต่อระยะไกล

วิธีการแกะรอยที่เป็นระบบจะช่วยให้ผู้บุกรุกสามารถสร้างโพรไฟล์ที่สมบูรณ์ขององค์กรได้ โดยเฉพาะในด้านเกี่ยวกับระบบรักษาความปลอดภัยและช่องโหว่ต่าง ๆ โดยการใช้เครื่องมือและเทคนิคต่าง ๆ ผสมผสานกันผู้บุกรุกจะได้รับข้อมูลต่าง ๆ เป็นจำนวนมาก แต่เขาจะต้องนำมากลั่นกรองให้เหลือเฉพาะในสโคปที่น่าสนใจอย่างเช่น ชื่อโดเมน, เน็ตเวิร์คบล็อก, และหมายเลขไอพีแอดเดรส ของเครื่องคอมพิวเตอร์ที่เชื่อมต่อโดยตรงกับอินเตอร์เน็ต ถึงแม้เทคนิคเกี่ยวกับการแกะรอยจะมีอยู่มากมาย แต่โดยภาพรวมแล้ว ต่างก็มีเป้าหมายเดียวกันคือ เพื่อให้ได้มาซึ่งข้อมูลที่เกี่ยวข้องกับเทคโนโลยีต่อไปนี้คือ อินเตอร์เน็ท, อินทราเน็ต, การบริการการเชื่อมต่อระยะไกล และ เอ็กซ์ทราเน็ต ตารางที่ 3.1 เป็นการสรุปข้อมูลสำคัญเกี่ยวกับเทคโนโลยีที่ผู้บุกรุกต้องการทราบ

เทคโนโลยี	**ข้อมูลสำคัญเกี่ยวกับเทคโนโลยีนั้น ๆ**
อินเตอร์เน็ต (internet)	◆ ชื่อโดเมน ◆ เน็ตเวิร์กบล็อก ◆ หมายเลขไอพีแอดเดรส ของเครื่องคอมพิวเตอร์ที่เชื่อมต่อโดยตรงกับอินเตอร์เน็ต ◆ เน็ตเวิร์กเซอร์วิสที่ทำงานโดยตรงกับโปรโตคอล TCP และ UDP ของโปรเซสเซอร์ เช่น SPARC, X86 ◆ Access Control List (ACL) ซึ่งเป็นกลไกลการควบคุมการเข้าถึงทรัพยากรต่าง ๆ บนเครื่อง ◆ ระบบป้องกันผู้บุกรุก (Intrusion Detection Systems, IDSes) ◆ การระบุรายละเอียดเกี่ยวกับระบบในแง่ต่าง ๆ (รายชื่อผู้ใช้และกลุ่มต่าง ๆ , แบนเนอร์, ตารางเราท์ติงของเราท์เตอร์, ข้อมูลของ SNMP)
อินทราเน็ต (intranet)	◆ เน็ตเวิร์กโปรโตคอลที่ใช้อยู่ภายใน(เช่น โปรโตคอล IP, IPX, DecNet ◆ ชื่อโดเมนภายใน ◆ เน็ตเวิร์กบล็อก ◆ หมายเลขไอพีแอดเดรสของเครื่องคอมพิวเตอร์ที่เชื่อมต่อโดยตรงกับอิน ◆ เน็ตเวิร์กโปรโตคอลที่ใช้อยู่ภายใน(เช่น โปรโตคอล IP, IPX, DecNet ◆ ชื่อโดเมนภายใน

ตารางที่ 3.1 เทคโนโลยีและข้อมูลสำคัญที่ผู้บุกรุกต้องการค้นหา

	◆ เน็ตเวิร์กบล็อก ◆ หมายเลขไอพีแอดเดรสของเครื่องคอมพิวเตอร์ที่เชื่อมต่อโดยตรงกับอินเทอร์เน็ต ◆ เน็ตเวิร์กเซอร์วิสที่ทำงานบนโปรโตคอล TCP, UDP ประเภทของโปรเซสเซอร์ (เช่น SPARC, X86) ◆ Access Control List (ACL) ซึ่งเป็นกลไกควบคุมการเข้าถึงทรัพยากรต่าง ๆ บนเครื่อง ◆ ระบบป้องกันผู้บุกรุก (Intrusion Detection Systems, IDSes) ◆ การระบุรายละเอียดเกี่ยวกับระบบในแง่ต่าง ๆ (รายชื่อผู้ใช้และกลุ่มต่าง ๆ , แบนเนอร์, ตารางเราท์ติงของเราท์เตอร์, ข้อมูลของ SNMP)
การเชื่อมต่อจากระยะไกล (Remote Access)	◆ เบอร์โทรศัพท์ในระบบอะนาล็อก/ดิจิตอล ◆ ประเภทของเซิร์ฟเวอร์ที่ให้บริการ ◆ กลไกการตรวจสอบผู้ใช้ (Authentication Mechanism)
เอ็กซ์ทราเน็ต (Extranet)	◆ คอนเน็กชันต้นทางและปลายทาง ◆ ประเภทของคอนเน็กชัน ◆ กลไกควบคุมการเข้าถึงทรัพยากร (Access Control Mechanism)

ตารางที่ 3.1 เทคโนโลยีและข้อมูลสำคัญที่ผู้บุกรุกต้องการค้นหา(ต่อ)

ขั้นตอนของการแกะรอย

3.2.1.1 การกำหนดขอบเขตหรือสโคปของการแกะรอย

ผู้บุกรุกจะต้องพิจารณาว่าจะแกะรอยเน็ตเวิร์กทั้งองค์กรหรือสนใจเฉพาะบางส่วนในบางครั้งมันก็เป็นงานที่ท้าทายพอสมควรเหมือนกันในการพิจารณาว่าสิ่งไหนเป็นสิ่งที่เกี่ยวข้องกับองค์กรเป้าหมายแต่โชคดีที่อินเตอร์เน็ตให้แหล่งเครื่องมือมากมายที่ช่วยให้ผู้บุกรุกสามารถจำกัดสโคปให้แคบเข้าและยังให้ความเข้าใจลึกซึ้งกับประเภทและปริมาณของข้อมูลสาธารณะที่เปิดเผยอยู่บนอินเตอร์เน็ต

3.2.1.2 การรวบรวมรายละเอียดต่าง ๆ ของเน็ตเวิร์กเป้าหมาย

ขั้นตอนของการรวบรวมรายละเอียดก็คือ การหาชื่อโดเมนและเน็ตเวิร์กที่เกี่ยวโยงองค์กรเป้าหมาย ชื่อโดเมนจะถือเป็นชื่อบริษัทที่ปรากฏในอินเตอร์เน็ทอย่างชัดเจน

การหาข้อมูลจะหาได้จากการค้นหาจากฐานข้อมูลของเวิร์ฟเวอร์ที่ให้บริการ Whois บนอินเตอร์เน็ต เซิร์ฟเวอร์ Whois จะให้ข้อมูลมากมาย มีหลายวิธีในการสอบถามข้อมูลฐานข้อมูลของ Whois แสดงดังตารางที่ 3.2

เครื่องมีหรือกลไกที่ใช้	แหล่งข้อมูล	แพลตฟอร์ม
เว็บเบราเซอร์	http://www.neworksolution.com/ http://www.arin.net	ทุก ๆ แพลตฟอร์มที่มีเว็บเบราเซอร์
ไคลเอ็นต์ Whois	Whois ได้มากับยูนิกซ์เกือบทุกยี่ห้อ	ยูนิกซ์
WS Ping ProPack	http://www.ipswitch.com	วินโดวส์ 95/NT/2000
Sam Spade	http://www.samspade.org/ssw	วินโดวส์ 95/NT/2000
Sam Spade Web interface	http://www.samspade.org	ทุก ๆ แพลตฟอร์มที่มีเว็บเบราเซอร์
Netscan tools	http://www.nwspw.com	วินโดวส์ 95/NT/2000
Xwhois	http://www.oxygene.500mhz.net/whois/	ยูนิกซ์ที่มี Xwindows และ GTK+GUI Toolkit

ตารางที่ 3.2 เว็บไซต์ที่เปิดให้บริการ Whois

แต่ละวิธีจะให้คำตอบเหมือน ๆ กัน การสอบถามแต่ละประเภทจะได้ข้อมูลแตกต่างกันไป ประเภทของคำถามต่อไปนี้จะให้ข้อมูลสำคัญส่วนใหญ่ที่ผู้บุกรุกจำเป็นต้องใช้ในการโจมตี

- Registra เพื่อแสดงข้อมูลเกี่ยวกับเซิร์ฟเวอร์ที่ทำหน้าที่รับรีจีสเตอร์ชื่อโดเมนและเซิร์ฟเวอร์ whois ที่เกี่ยวข้อง
- Organizational เพื่อแสดงรายชื่อโดเมนที่เกี่ยวข้องกับองค์กรเป้าหมาย
- Domain เพื่อแสดงข้อมูลที่เกี่ยวข้องกับชื่อโดเมนที่สนใจ
- Network เพื่อแสดงข้อมูลที่เกี่ยวข้องกับเน็ตเวิร์กที่สนใจหรือเกี่ยวข้องกับไอพีแอดเดรส ที่สนใจเพียงหมายเลขเดียว
- Point Of Contact (POC) เพื่อแสดงข้อมูลที่เกี่ยวข้องกับบุคคลที่สนใจ โดยทั่วไปมักจะเป็นผู้รับผิดชอบระบบหรือผู้ติดต่อได้ในกรณีที่ระบบมีปัญหา (administrative contact)

3.2.1.3 การล้วงความลับจาก DNS Server

หลังจากทราบชื่อโดเมนแล้ว ผู้บุกรุกสามารถเริ่มสอบถามจาก DNS Server ได้ ฐานข้อมูลที่เก็บอยู่บน DNS Server เป็นฐานข้อมูลที่จับคู่ระหว่างชื่อโฮสต์และไอพีแอดเดรสของโฮสต์นั้น ๆ หากทำการคอนฟิก DNS Server ไม่ดีพอจะทำให้ผู้บุกรุกสามารถทำโซนทรานสฟอร์จาก DNS Server ได้

คำว่าโซนทรานสเฟอร์(Zone Transfer) หมายถึง การที่ DNS Server ตัวที่เป็น Secondary Server เข้าไปโอนย้ายข้อมูล DNS จาก DNS Server ตัวที่เป็น Primary Server มาเก็บสำรองเอาไว้ในเครื่องของตนเองทั้งนี้เพื่อให้บริการค้นหาข้อมูลจาก DNS Server ดำเนินไปได้อย่างต่อเนื่อง ถึงแม้ว่า Primary Server จะ

ดาวน์หรือแครชไป โดยทั่วไป Primary Server ควรจะยอมให้เฉพาะ Secondary Server ที่เหมาะสมเท่านั้นที่สามารถทำโซนทรานเฟอร์จากตนเองไปได้ แต่ส่วนใหญ่ Primary DNS Server มักจะคอนฟิกพลาดทำให้สามารถสำเนาฐานข้อมูล DNS ผ่านทางการทำโซนทรานเฟอร์แก่เซิร์ฟเวอร์ตัวไหนก็ได้ที่ต้องการ องค์กรส่วนใหญ่มักไม่ค่อยใช้กลไกของ public/private DNS ซึ่งเป็นการแบ่งแยก DNS Server ตัวที่ดูแลรายชื่อที่โฮสต์ที่เปิดสู่อินเตอร์เน็ต(เรียกว่า public DNS Server) กับ DNS Server ตัวที่ดูแลรายชื่อ โฮสต์ที่อยู่ในเน็ตเวิร์กภายใน (private DNS server) ออกจากกัน การแบ่งแยกดังกล่าวมีจุดประสงค์เพื่อไม่ให้บุคคลภายนอก ทราบข้อมูลเกี่ยวกับชื่อโฮสต์และหมายเลขไอพีแอดเดรสภายใน ซึ่งถ้าผู้บุกรุกทราบข้อมูลส่วนนี้ผ่านอินเตอร์เน็ต ย่อมไม่เป็นผลดีแน่ เพราะเขาจะได้พิมพ์เขียว หรือแผนที่เส้นทางที่ละเอียดของเน็ตเวิร์กภายในบริษัท

3.2.1.4 การสำรวจเน็ตเวิร์ก

เป็นการค้นหาเน็ตเวิร์กและโทโปโลยีพร้อมทั้งเส้นทางในการเข้าถึงเน็ตเวิร์กนั้น ๆ

3.2.2 การสแกนเพื่อการตรวจสอบ (Scanning)

การสแกนเปรียบเสมือนการแกะรอย ผู้บุกรุกจะได้รับลิสต์หมายเลขไอพีแอดเดรสและข้อมูลเน็ทเวิร์กขององค์กรเป้าหมายผ่านทางการสอบถามจากฐานข้อมูล Whois และการดาวน์โหลด Zone Database จาก DNS Server เทคนิคเหล่านี้ล้วนให้ข้อมูลที่เป็นประโยชน์ต่อผู้บุกรุกมากที่เดียว ไม่ว่าจะเป็นชื่อพนักงานในองค์กร หรือเบอร์โทรศัพท์, ช่วงของไอพีแอดเดรส, DNS Server
ผู้บุกรุกจะต้องแน่ใจด้วยว่าจะต้องแน่ใจด้วยว่า เครื่องคอมพิวเตอร์ที่สแกนเจอนั้นเปิดใช้งานอยู่และเข้าถึงจากอินเตอร์เน็ตได้ และถ้าเป็นไปได้ควรทราบด้วยว่ามีหมายเลขพอร์ตใดเปิดอยู่บ้าง เทคนิคพื้นฐานที่นิยมทำกันคือ การ ping ไปยังเครื่องเป้าหมายจำนวนมากพร้อม ๆ กัน สามารถแบ่งประเภทของการสแกนได้ดังนี้

3.2.2.1 TCP connection scan

เป็นการคอนเน็กไปยังที่พอร์ตที่ต้องการบนเครื่องปลายทางแล้วขอเปิดคอนเน็คชันของโปรโตคอล TCP ด้วยกลไกมาตรฐานเรียกว่า TCP Three-way handshake(SYN, SYB/ACK และACK) แต่วิธีนี้มักถูกตรวจจับได้ง่าย

3.2.2.2 TCP SYN scan

เทคนิคนี้บางครั้งถูกเรียกว่า half-open scanning สาเหตุเพราะว่าคอนเน็คชันที่สมบูรณ์ของโปรโตคอล TCP ยังไม่ถูกเปิดขึ้น เพราะเมื่อเพ็คเก็ต TCP ที่เซตแฟลก SYN ไว้เป็น 1 (TCP SYN) ได้ถูกส่งไปเครื่องปลายทาง ถ้าเครื่องปลายทางตอบกลับมาด้วยแพ็คเก็ต TCP ที่เซตค่าแฟลก SYN และ ACK ไว้เป็น 1 (TCP SYN/ACK) นั่นก็เพียงพอแล้วที่จะสรุปว่า พอร์ตดังกล่าวอยู่ในสถานะ LISTENNING แต่ถ้าพอร์ตดังกล่าวไม่ได้เปิดอยู่ แพ็คเก็ต TCP ที่เซตค่าแฟลก RST และ ACK ไว้เป็น 1 (TCP RST/ACK)จะถูกส่งกลับมาแทน

3.2.2.3 TCP FIN scan

เทคนิคนี้จะส่งแพ็คเก็ต TCP ที่เซตค่าแฟลก FIN เป็น 1 (TCP FIN) ไปยังพอร์ตที่ต้องการ ถ้าไดรเวอร์ของโปรโตคอล TCP/IP ที่เครื่องปลายทางได้ถูกพัฒนาขึ้นมาโดยมี ฟีเจอร์ตามในมาตรฐาน RFC

หมายเลข 739 ครบถ้วน เครื่องปลายทางจะส่งแพ็คเก็ต TCP RST ของทุก ๆ พอร์ตที่ปิดอยู่กลับมา เราจะทราบหมายเลขพอร์ตที่ไม่เปิดให้บริการโดยปกติแล้ว เทคนิคนี้มักใช้ได้กับเครื่องปลายทางที่รันระบบยูนิกซ์

3.2.2.4 TCP Xmas Tree scan

เทคนิคนี้จะส่งแพ็คเก็ต TCP ที่เซตค่าแฟลก FIN, URG และ PUSH ไปยังพอร์ตเป้าหมายที่เครื่องปลายทาง และอาศัยมาตรฐาน RFC 739 เครื่องปลายทางจะส่งแพ็คเก็ต TCP RST ของทุกพอร์ตที่ปิดอยู่กลับมาให้

3.2.2.5 TCP Null scan

เทคนิคนี้จะส่งแพ็คเก็ต TCP ออกไปโดยเซตค่าของทุก ๆ แฟลกให้เป็น 0 หมด และอาศัยมาตรฐาน RFC 739 เครื่องปลายทางจะส่งแพ็คเก็ต TCP RST ของทุกพอร์ตที่ปิดอยู่กลับมาให้

3.2.2.6 TCP ACK scan

เทคนิคนี้จะใช้เพื่อค้นหา "rule" และ "policy" ต่าง ๆ ที่เซตไว้ที่ไฟร์วอลล์ เพื่อตรวจสอบดูว่าไฟร์วอลล์นั้น ๆ ทำหน้าที่แต่เพียงกรองแพ็คเก็ตง่าย ๆ หรือเป็นไฟร์วอลล์ที่มีความฉลาดพอสมควรและใช้เทคนิคการกรองแพ็คเก็ตชั้นสูง

3.2.2.7 TCP Windows scan

เทคนิคนี้ตรวจสอบพอร์ตที่เปิดอยู่ รวมทั้งตรวจดูว่า พอร์ตใดบ้างที่ถูกฟิลเตอร์เอาไว้ไม่ให้ผ่านเข้าไปถึง และพอร์ตหมายเลขใดได้รับการอนุญาตไว้บ้าง โดยอาศัยช่องโหว่จากความผิดปกติบางอย่างจากการแจ้งค่าของ TCP Windows Size ของโปโตคอล TCP/IP

3.2.2.8 TCP RPC scan

เทคนิคนี้ใช้งานได้เฉพาะกับเครื่องปลายทางที่รันยูนิกซ์เท่านั้น มันถูกใช้เพื่อตรวจสอบดูว่ามีเซอร์วิสใดทำงานอยู่บนเซอร์วิส RPC บ้าง รวมทั้งตรวจดูเวอร์ชันของเซอร์วิสนั้นและโปรแกรมที่เกี่ยวข้อง

3.2.2.9 UDP scan

เทคนิคนี้จะส่งเพ็คเก็ตของโปรโตคอล UDP ไปยังพอร์ตเป้าหมาย ถ้าเครื่องปลายทางตอบกลับมาด้วยแพ็คเก็ต ICMP type UNREACHABLE นั่นหมายความว่าพอร์ตที่ปิดอยู่ในทางตรงกันข้ามเราไม่ได้รับ ถ้าเราไม่ได้รับแพ็คเก็ต ICMP type ดังกล่าว เราสามาถสรุปได้ว่าพอร์ตนั้นเปิดอยู่เนื่องจากโปรโตคอล UDP เป็นโปรโตคอลลักษณะ connectionless คือไม่รับรองแพ็คเก็ตที่ส่งไปจะไปถึงปลายทางครบถ้วนหรือไม่ ดังนั้นความถูกต้องก็อาจจะขึ้นอยู่กับปัจจัยอื่นด้วย เช่น ปริมาณ
ทราฟฟิกในเน็ตเวิร์กและทรัพยากรบนเครื่องปลายทาง นอกจากนั้นมันยังเป็นเทคนิคที่ค่อนข้างช้า

3.2.3 การค้นหาและรวบรวมรายละเอียด

การค้นหาและรวบรวมรายละเอียดจะใช้เทคนิคการเปิดคอนเน็คชั้นไปยังเครื่องปลายทางโดยตรงและมีการส่งคำถามไปถามด้วย ข้อมูลต่าง ๆ ที่ได้จากขั้นตอนนี้อาจจะดูว่ามีไม่ส่งผลกระทบอะไรมากนัก แต่ข้อมูลเพียงเล็กน้อยเหล่านี้อาจจะนำไปสู่ช่องโหว่ของระบบได้ เช่น ทันทีที่ผู้บุกรุกทราบแอคเคาน์ของ

ผู้ใช้ที่ถูกต้อง หรือทราบชื่อแชร์โฟลเดอร์ ผู้บุกรุกจะทำการคาดเดารหัสผ่านหรือค้นหาจุดบกพร่องของสิทธิที่เซตไว้ที่แชร์โฟลเดอร์นั้น ๆ ประเภทของข้อมูลที่ผู้บุกรุกต้องการรวบรวมสามารถแบ่งได้เป็นสามกลุ่มใหญ่ ๆ ดังต่อไปนี้

- รายชื่อทรัพยากรในเน็ตเวิร์ก เช่น ชื่อแชร์โฟลเดอร์ต่าง ๆ
- รายชื่อแอคเคาน์ของผู้ใช้และรายชื่อกลุ่มต่าง ๆ
- รายชื่อแอปพลิเคชันและแบนเนอร์ของมัน

การรวบรวมข้อมูลเหล่านี้จะมีเทคนิคแตกต่างกันไปตามระบบปฏิบัติการ ดังนั้นจะต้องทราบประเภทของระบบปฏิบัติการของเครื่องปลายทางเสียก่อน

3.3 การโจมตีระบบปฏิบัติการยูนิกซ์

การที่จะโจมตีระบบยูนิกซ์นั้นผู้บุกรุกจะต้องทราบข้อบกพร่องที่มีอยู่หรือข้อบกพร่องที่มีแนวโน้มว่าจะเกิดขึ้นในอนาคต เช่น มีเซอร์วิสอะไรบ้าง เวอร์ชันของเซอร์วิสที่รัน โครงสร้างของระบบปฏิบัติการ รายชื่อแอคเคาน์ของผู้ใช้ ผู้บุรุกจะต้องค้นหาก่อนว่า สิ่งต่าง ๆ เหล่านี้มีช่องโหว่ด้านความปลอดภัยหรือไม่ อย่างไร ซึ่งหนทางในการค้นหาจะมีอยู่ด้วยกันหลายทาง ดังนี้

- ค้นหาจากเว็บไซต์ที่เป็นแหล่งรวบรวมช่องโหว่และวิธีการป้องกัน เช่น Bugtraq, Computer Emergency Response Team Advisories(www.cert.org) และเว็บไซต์ของผู้ผลิตยูนิกซ์เอง
- ใช้โปรแกรมสาธารณะ ,ฟรีแวร์หรือแชร์แวร์ต่างๆ ที่มีผู้โพสต์ไว้ที่เมล์ลิ่งลิสต์และเว็บไซต์ต่าง ๆ หรือเขียนโปรแกรมขึ้นมาเอง
- ใช้เครื่องมือสแกนหาช่องโหว่แบบอัตโนมัติเพื่อค้นหาว่าเครื่องเป้าหมายของการโจมตีมีช่องโหว่นั้นจริงหรือไม่

3.3.1 การโจมตีจากระยะไกล (remote attack)

เป็นการเจาะระบบผ่านทางเน็ตเวิร์กหรือผ่านช่องทางการสื่อสารอื่น ๆ เช่นการหมุนโทรศัพท์เข้ามายังโมเด็มที่ต่ออยู่กับระบบยูนิกซ์ โดยสรุปแล้วสามารถแบ่งวิธีการสำคัญสำหรับการเจาะระบบยูนิกซ์ได้ 3 วิธีใหญ่ ๆ ได้แก่

- การฉวยโอกาสจากเซอร์วิสที่รันอยู่ (Exploiting listening service)
- เจาะผ่านยูนิกซ์ที่ทำหน้าที่เป็นเราท์เตอร์(router) ที่ทำหน้าที่รักษาความปลอดภัยไปในตัวด้วยระหว่างเน็ตเวิร์กภายนอกกับภายใน
- การอาศัยแอปพลิเคชันหรือโปรแกรมที่ผู้ใช้เอ็กซีคิวต์เอง

Brute Force Attack คือการเดาแอ็คเคาน์/รหัสผ่านที่ใช้ล็อกออนเข้าไปยังเซอร์วิสต่าง ๆ โดยทดลองผสมผสานตัวอักษร ตัวเลขต่าง ๆ เข้าด้วยกันแล้วเดาว่าใช่รหัสผ่านหรือไม่

บัฟเฟอร์โอเวอร์โฟลว์แอทแท็ค (buffer overflow Attack) เงื่อนไขที่ทำการเกิดบัฟเฟอร์โอเวอร์โฟลว์เป็นจริงจะเกิดขึ้นเมื่อ ผู้ใช้หรือโปรเซสพยายามบรรจุอินพุตเข้าไปในบัฟเฟอร์(หรืออาเรย์ขนาดคงตัว) มากเกินกว่าขนาดของบัฟเฟอร์ได้ถูกจัดสรรไว้ ทำให้เกิดอินพุตข้อมูลเกิดการ "ล้น" หรือโอเวอร์โฟลว์ขึ้น ผู้บุกรุกจะสามารถเอ็กซีคิวโค๊ดคำสั่งต่างๆ ที่ root มี

3.3.2 การโจมตีระบบโดยตรง (local attack)

เป็นการโจมตีระบบโดยการเจาะระบบผ่างทางคอมมานด์เชลล์ (command shell) หรือผ่านการล็อกออนเข้าไปที่ระบบปฏิบัติการโดยตรงโดยอาศัยช่องโหว่ต่าง ๆ เช่น

ช่องโหว่เกี่ยวกับรหัสผ่านที่ตั้งไว้ไม่ดีพอ การแคร็กรหัสผ่านเรียกกันว่า "การโจมตีด้วยการคาดเดารหัสรหัสผ่านจากคำในดิกชันนารีโดยอัตโนมัติ(automated dictionary attack) "ซึ่งเรียกสั้น ๆ ว่า "ดิกชันนารีแอทแท็ค" การคาดเดารหัสผ่านสามารถทำแบบออฟไลน์ได้คือ เดาที่เครื่องของตนเองโดยยังไม่ต้องทดลองล็อกออนไปยังเซิร์ฟเวอร์ เพื่อทำให้ผู้ดูแลระบบสังเกตเห็นได้ยาก วิธีนี้ถือว่าเป็น local attack ก็เพราะผู้บุกรุกต้องได้ไฟล์ /etc/passwd หรือไฟล์รหัสผ่านที่ถูกชาโดว์(shadow) ไว้ สำหรับในระบบยูนิกซ์ ผู้บุกรุกไม่จำเป็นต้องพยายามคอนเน็คเข้าหาเซอร์วิสที่รันอยู่บนระบบยูนิกซ์เลย ผู้บุกรุกจะพยายามเดารหัสผ่านของแต่ละแอคเคาน์โดยการใช้โปรแกรมแคร็กเข้ารหัสคำต่าง ๆ ที่อยู่ในดิกชันนารีหรือเข้ารหัสคำต่าง ๆ ที่ผสมผสานกันแบบสุ่ม ผลที่ได้คือ ค่าแฮชของรหัสผ่านที่ถูกคำนวณโดยโปรแกรมแคร็กรหัสผ่าน แล้วเปรียบเทียบดูผลลัพธ์ที่ได้กับค่าแฮชของรหัสผ่านที่เก็บอยู่ในไฟล์ /etc/passwd ว่าเท่ากันหรือไม่ ถ้าค่าแฮชของรหัสผ่านที่โปรแกรมแคร็กรหัสผ่านคำนวณได้ตรงกันกับค่าแฮชรหัสผ่านที่เก็บอยู่ในไฟล์ /etc/passwd นั่นแสดงว่า คำในดิกชันนารีนั้นเป็นรหัสผ่านของผู้ใช้

โลคอลบัฟเฟอร์โอเวอร์โฟลว์ (local buffer overflow) ช่องโหว่ที่เกิดจากบัฟเฟอร์โอเวอร์โฟลว์เป็นโอกาสที่ผู้บุกรุกจะทำการคอมไพล์โปรแกรมที่ต้องการเพื่อจะโจมตีระบบหรือสร้างช่องทางที่จะโจมตี

การคอนฟิกระบบที่ไม่เหมาะสม ระบบสามารถถูกเจาะเข้าไปได้เพียงเพราะการคอนฟิกค่าต่างๆไว้ไม่ดีพอหรือผู้ดูแลระบบไม่ค่อยเอาใจใส่ เช่น หลังจากติดตั้งระบบเสร็จใหม่มันจะเซตมาตรฐานการรักษาความปลอดภัยที่เข้มงวดอยู่แล้ว แต่ถ้ามีการเปลี่ยนสิทธิในการเข้าถึงไฟล์ /etc/passwd ให้ทุกคนสามารถเขียนได้ อย่างนี้ความปลอดภัยทั้งหมดก็จะเป็นจุดอ่อนทันที

3.3.3 โปรแกรมโทรจัน

ทันทีที่ได้สิทธิพิเศษของ root ผู้บุกรุกสามารถ "โทรจันไนซ์ (Trojanize)" โปรแกรมหรือคำสั่งใด ๆ ก็ได้บนยูนิกซ์ โทรจันอาจจะสร้างประตูทางลับ(backdoor) ไว้ที่ระบบของคุณโดยการรันโปรเซสที่คอยดักจับอยู่บนพอร์ต TCP (TCP listener) และขโมยเชลล์ (shell) ผ่านทางแชนเนลของการสื่อสารกับเซอร์วิสที่มาติดต่อกับพอร์ต TCP นั้น กลับไปยังเครื่องของผู้บุกรุกแทน เช่น ใช้คำสั่ง ls เพื่อเช็คดูว่ามีโปรแกรมโทรจันนั้น ๆ รันอยู่หรือไม่ ถ้ายังไม่ได้รัน มันจะสั่งรันแฮ็คอัปเวอร์ชันของ netcat แทนเพื่อให้ส่งผลของการเอ็กซ์ซีคิว /bin/sh กลับมายังเครื่องของผู้บุกรุกเมื่อผู้บุกรุกคอนเน็กไปหาเครื่องเป้าหมายนั้น

บทที่ 4
หลักการเขียนโปรแกรมเบื้องต้นที่ใช้ในการสร้างระบบ

4.1 ลีนุกซ์เคอร์เนลโมดูล

4.1.1 ทำความรู้จักกับลีนุกซ์เคอร์เนลโมดูล

ลีนุกซ์เคอร์เนลโมดูลใช้ในการขยายฟังก์ชันให้กับเคอร์เนลของลีนุกซ์โดยมีข้อดี คือ เป็นการโหลดข้อมูลแบบไดนามิกซึ่งทำให้ไม่ต้องทำการโหลดข้อมูลทั้งหมดโดยโหลดเพียงส่วนที่มีการใช้งานเท่านั้นและสามารถใช้งานได้เลยโดยไม่ต้องคอมไพล์เคอร์เนลใหม่

ลีนุกซ์โมดูลมีฟังก์ชันพื้นฐาน 2 ฟังก์ชัน คือ

1. int init_module(void) /* เป็นฟังก์ชันที่ใช้ในการกำหนดค่าเริ่มต้น */
2. void cleanup_module(void) /* เป็นฟังก์ชันที่ใช้ในการ shutdown */

โดยทั่วไปแล้วการโหลดโมดูลต้องมีสิทธิ์เป็นรูตเท่านั้น โดยใช้คำสั่งดังนี้ :

#insmod module.o

คำสั่งนี้ส่งผลต่อระบบ คือ

- โหลดออบเจ็กต์ไฟล์ (object file) ขึ้นมา ในที่นี้ คือ module.o
- เรียกซีสเต็มคอล (systemcall) ชื่อ create_module เพื่อทำการจองเนื้อที่ในหน่วยความจำ
- ซีสเต็มคอลชื่อ get_kernel_syms จะอ้างอิงในส่วนของ Kernel-symbols
- หลังจากกำหนดค่าเริ่มต้นโดยใช้ซีสเต็มคอล init_module แล้วก็จะทำการเอ็กซิคิวต์ (execute) ฟังก์ชัน int init_module(void) ต่อไป

ซึ่งรายละเอียดของ Kernel-Symbols จะขออธิบายไว้ในหัวข้อต่อๆ ไป

ตัวอย่าง ข้างล่างนี้จะเป็นตัวอย่างของการสร้างโมดูลอย่างง่ายๆ

```
#define __KERNEL__
#define MODULE
#include <Linux/kernel.h>
#include <Linux/module.h>
int init_module(void) {
        printk("<1>Hello World\n");
        return 0;
}
void cleanup_module(void) {
        printk("<1Bye, Bye");
}
```

สังเกตว่าในตัวอย่างข้างต้นจะใช้คำสั่ง printk ไม่ใช้ printf เนื่องจากการเขียนโปรแกรมในเคอร์เนล จะแตกต่างกับการเขียนโปรแกรมในยูสเซอร์สเปซมาก ดังนั้นเวลาการเขียนโปรแกรมในเคอร์เนลจำเป็นที่จะต้องระมัดระวังคำสั่งการใช้งานมากเนื่องจากคำสั่งเหล่านี้อาจมีข้อจำกัดในการใช้งาน ดังนั้นจึงควรเรียนรู้ฟังก์ชันต่าง ๆ ในแอพพลิเคชันของ userspace ไว้เพื่อช่วยในการเจาะระบบเคอร์เนลได้

จากตัวอย่างข้างต้นสามารถคอมไพล์โดยใช้คำสั่ง

```
# gcc    -c  -03  helloworld.c
```

โหลดโมดูลโดยใช้คำสั่ง

```
# insmod  helloworld.o
```

เมื่อโมดูลถูกโหลดขึ้นมาและแสดงตัวอักษรขึ้นมา เราใช้คำสั่งตรวจสอบเพื่อให้เห็นว่าลีนุกซ์โมดูลยังอยู่ในเคอร์เนลสเปซ ดังนี้

```
# lsmod
```

Module	Pages	Used by
Helloworld	1	0

คำสั่งข้างต้นจะอ่านข้อมูลใน /proc/modules เพื่อโหลดโมดูลที่ชื่อ helloworld โดยใช้เวลาชั่วขณะหนึ่ง ส่วน Page เป็นข้อมูลของหน่วยความจำ (จำนวนของ page ที่อยู่ในโมดูล) และ Used by จะบอกความถี่ที่ระบบเรียกใช้โมดูล helloworld (อ้างอิงการนับ) เราสามารถถอดโมดูลออกจากระบบได้ หากค่าของตัวแปร counter มีค่าเป็น 0 โดยคำสั่งที่ใช้ย้าย คือ

```
# rmmod  helloworld
```

ในลักษณะของลีนุกซ์โมดูลมีลักษณะคล้ายกับโปรแกรม DOS TSR รุ่นเก่า คือ สามารถจับสัญญาณอินเทอร์รัปต์ทุกสัญญาณที่เราต้องการได้ ดังเช่น WIN9X ของบริษัท Microsoft ที่มีการเรียก (call) VxD ซึ่งเหมือนกับ ลีนุกซ์โมดูลที่มีการเรียกฟังก์ชันของระบบ เรียกว่า ซีสเต็มคอล (systemcalls)

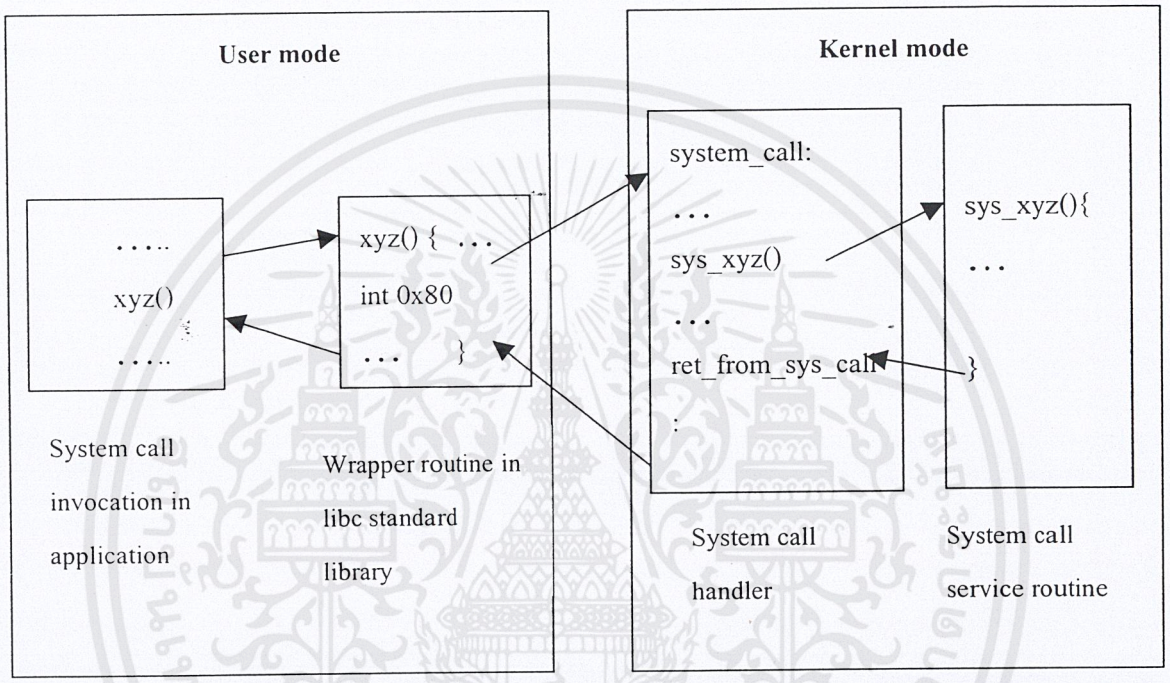
4.2 ซีสเต็มคอล

ระบบปฏิบัติการส่วนมากในทุกวันนี้อนุญาตให้โพรเซสต่างๆ สามารถติดต่อกับอุปกรณ์ เช่น ดิสก์ เครื่องพิมพ์ ได้จาก user mode เลยโดยมีการแบ่งเลเยอร์ระหว่างในส่วนฮาร์ดแวร์เลเยอร์ กับ แอพพลิเคชันเลเยอร์ ซึ่งในการจัดการเช่นนี้มีข้อดีอยู่หลายข้อ คือ

- ทำให้การเขียนโปรแกรมทำได้ง่ายขึ้นเพราะคนเขียนโปรแกรมไม่จำเป็นต้องเขียนโปรแกรมในระดับต่ำ
- ในแง่ของทางด้านการรักษาความปลอดภัยเพราะเคอร์เนลสามารถตรวจสอบการร้องขอได้เพราะจำเป็นต้องร้องขอผ่านเคอร์เนล

การติดต่อกันระหว่าง user mode process กับ hardware device นั้นทำโดยการเรียกใช้ซีสเต็มคอลผ่านทางเคอร์เนล สำหรับในลีนุกซ์นั้นการเรียก ซีสเต็มคอล นั้นทำได้โดยการอินเทอร์รัปต์หมายเลข 0x80 และส่งค่าหมายเลข ซีสเต็มคอล ไว้ที่รีจิสเตอร์ eax

จากรูปเป็นการแสดงความสัมพันธ์ของการเรียก ซีสเต็มคอล จากแอพพลิเคชัน เมื่อมีการร้องขอซีสเต็มคอลผ่านจากแอพพลิเคชัน ก็จะส่งต่อไปยัง wapper routine ที่ไลบารี libc เพื่ออินเทอร์รัปต์และเรียกให้ system call handler ทำงานส่วนนี้ใน kernel mode ซึ่งหลังจากนั้นก็รันโค้ดซีสเต็มคอล การเรียก system call routine นั้นเคอร์เนลมีตาราง system call dispatch table เก็บอยู่ในอาร์เรย์ชื่อ sys_call_table โดยสมาชิกแต่ละตัวมีตำแหน่งของ system call service routine อยู่

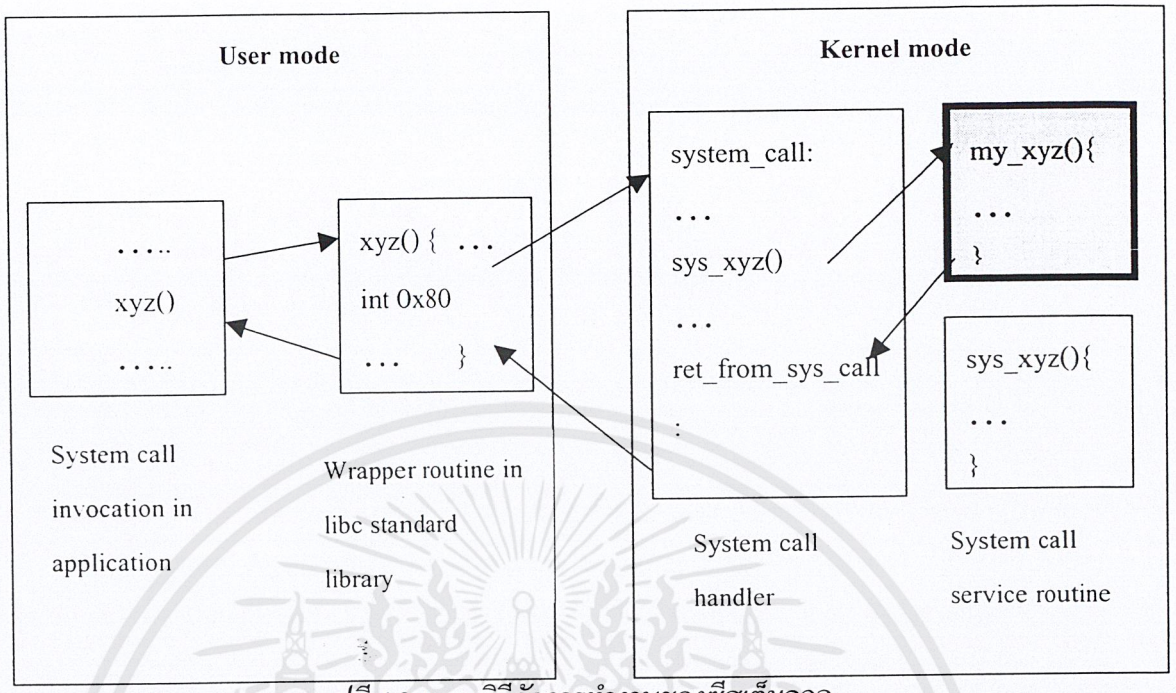


รูปที่ 4.1 แสดงความสัมพันธ์ของการเรียก ซีสเต็มคอล จากแอพพลิเคชัน

หมายเลขซีสเต็มคอลเป็นตัวชี้ไปยังอาร์เรย์ชื่อ sys_call_table[] โดยหมายเลขซีสเต็มคอลนี้จะตรงกับ (map) ฟังก์ชันการให้บริการ

4.3 การดักการทำงานของ ซีสเต็มคอล

จากรูปที่ 4.1 สังเกตว่าในส่วนของ kernel mode ก่อนมีการรัน system call routine นั้นต้องผ่านส่วน system call handler โดยดูตำแหน่งโปรแกรมของ system call routine ที่ตาราง sys_call_table ซึ่ง ณ จุดนี้ หากเราเปลี่ยนค่าตำแหน่งของโปรแกรมที่เก็บไว้ให้เป็นตำแหน่งของโปรแกรมที่เป็นโค้ดของเราเอง เมื่อ system call handler ทำการเรียกดูตำแหน่ง system call routine ค่าที่ได้กลายเป็นตำแหน่งของโปรแกรมของเราแทนการรัน system call routine ตามปรกติก็จะมารันในโปรแกรมของเรา จุดนี้เราสามารถเขียนโปรแกรมควบคุมการทำงานได้ตามความต้องการ โดยวิธีการคือสร้างโมดูลขึ้นมาแล้วไปเปลี่ยนค่าตำแหน่งโค้ดในตาราง sys_call_table และโหลดโค้ดเข้าไปในหน่วยความจำ



รูปที่ 4.2 แสดงวิธีดักการทำงานของซีสเต็มคอล

จากรูปกล่องสีทึบมุมบนซ้ายมือเป็นโปรแกรมที่เราเขียนขึ้นมาให้ทำงานแทน ซีสเต็มคอล เดิม โดยการเปลี่ยนค่าตำแหน่งโปรแกรมในตาราง sys_call_table ไปชี้ยังที่โปรแกรมเราแทน

อย่างไรก็ตามการดักการทำงานของ ซีสเต็มคอล นั้นอาจดูเหมือนเป็นการประสงค์ร้ายต่อระบบ ซึ่งก็เป็นความจริงเพราะถ้าหากผู้บุกรุกสามารถดักการทำงานของซีสเต็มคอลได้ก็จะสามารถควบคุมการทำงานของเครื่องได้อย่างมากเช่น ป้องกันการลบข้อมูลของตนโดยอาจใช้การดักการทำงานชอง ซีสเต็มคอล sys_unlink ขยายสิทธิ์การใช้งานของตนโดยดักการทำงานของ ซีสเต็มคอล sys_setuid เป็นต้น แต่ในความจริงแล้วเราสามารถนำวิธีการดักการทำงานมาช่วยในการดูแลระบบได้เช่นกันโดยเราต้องเลือกดักซีสเต็มคอลให้ตรงกับความต้องการของสิ่งที่เราต้องการทำ

ตัวอย่างข้างล่างนี้เป็นการใช้ประโยชน์จากการดักการทำงานของซีสเต็มคอลโดยหากผู้ดูแลระบบต้องการป้องกันไม่ให้มีการโหลดโมดูลเข้าสู่เคอร์เนลอีก ก็สามารถทำได้โดยการดักการทำงานของซีสเต็มคอล create_module ซึ่งทำหน้าที่ในการโหลดโมดูลเข้าสู่เคอร์เนล (ในคำสั่ง insmod)

```
#define __KERNEL__
#define MODULE
#include <linux/module.h>
#include <linux/kernel.h>

// ตาราง sys_call_table
extern void* sys_call_table[];
```

```
// ไว้สำหรับเก็บตำแหน่งปรกติของ ซีสเต็มคอล
asmlinkage int (*orig_create_module) (char *name, unsigned long size);
int my_ create_module (char *name, unsigned long size){
        return 0;                       // ที่ส่วนนี้โปรแกรมจะส่งค่า 0 กลับ เหมือนกับว่าไม่มี error เกิด
                                        // ขึ้นเลย แต่ความจริงแล้วเรายังไม่ได้ทำการโหลดโมดูลเลย
}

int init_module(void) {
// เมื่อมีการโหลดโมดูลนี้เข้าสู่เคอร์เนลจะทำการเรียกฟังก์ชันนี้ก่อน
        //  ทำการเก็บตำแหน่งของ ซีสเต็มคอล ต้นฉบับ
        orig_create_module=sys_call_table[__NR_create_module];
        // แทนที่ตำแหน่งโปรแกรมด้วยตำแหน่งของโปรแกรมของเรา
        sys_call_table[__NR_create_module]=(void *)my_create_module;
        return 0;
}
void cleanup_module(void){
// คืนตำแหน่งปรกติของ ซีสเต็มคอล เมื่อทำการลบโมดูลออกจากหน่วยความจำ
        sys_call_table[__NR_create_module]=orig_rmdir;
}
```

4.4 โพรเซส

4.4.1 ทำความรู้จักกับโพรเซส

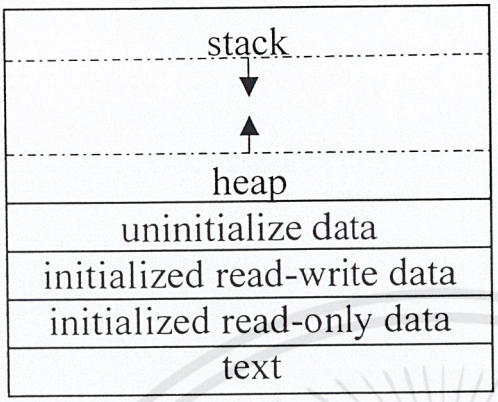
ตามข้อกำหนดของ X/Open หมายถึง ชุดคำสั่งในหน่วยความจำ กำลังรันอยู่ในขณะนั้น โดยมีชุดควบคุมการทำงาน 1 ชุดเป็นของตัวเอง โดยการรันนั้นจะต้องใช้งานรีซอร์สของระบบไปส่วนหนึ่ง พูดให้ง่ายๆ เข้าก็คือ งานที่กำลังรันอยู่ในหน่วยความจำนั่นเอง, ระบบที่เป็น multitasking เช่น ยูนิกซ์ สามารถจะรันโปรแกรมเพียงโปรแกรมเดียว ได้หลายโพรเซสในเวลาเดียวกัน เช่น เมื่อมีผู้ใช้งาน 10 คน กำลังใช้งานเท็กซ์อีดิตเตอร์อยู่ เป็นต้น นอกจากนี้ในฐานะที่เป็นระบบ muti-user หนทางเดียวที่จะสร้างโปรเซสใหม่ในระบบยูนิกซ์ คือการใช้ฟังก์ชั่นระดับต่ำ fork

ดังนั้น โพรเซสจะประกอบด้วยโค้ดของโปรแกรม ข้อมูลของโพรเซส ตัวแปร หมายเลขไฟล์ (file descriptor) และสภาพแวดล้อมการทำงานของโพรเซส

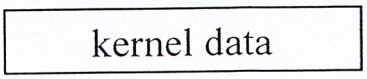
ข้อมูลบางอย่าง โพรเซสสามารถจะใช้งานร่วมกันได้ เช่น โค้ดของโปรแกรมที่เหมือนกัน หรือไลบารีที่ใช้งานร่วมกัน

4.4.2 โครงสร้างของโพรเซส

user context



kernel context



รูปที่ 4.3 แสดงโครงสร้างอย่างคร่าวๆ ของโพรเซส

user context จะประกอบไปด้วยส่วนต่างๆดังนี้

- text จะเก็บคำสั่งระดับต่ำ (machine instruction) ของเครื่องที่จะรันโดยฮาร์ดแวร์ ปรกติ จะกำหนดให้มี เพอร์มิสชั่นเป็น read only ดังนั้นโปรเซสไม่สามารถจะแก้ไขข้อมูลในส่วนนี้ได้ นอกจากนี้บางคำสั่งยังสามารถใช้งานร่วมกันได้อีกด้วย แต่ออบเจ็คบางอย่างก็ ไม่สามารถจะใช้งานร่วมกันได้ เช่น ตัวแปลสำหรับแต่ละโปรเซส, file descriptor หรือตัวแปรกำหนดสภาวะการทำงานของแต่ละโปรเซส

- data เก็บข้อมูลที่จะใช้ในโปรแกรม จะแบ่งเป็นสามส่วนด้วยกัน คือ uninitialized data, initialized read-write data และ initialized read only data
- heap จะใช้งานในขณะรันโพรเซส เพื่อกำหนดพื้นที่หน่วยความจำให้เพียงพอต่อความต้องการใช้งาน
- stack สำหรับควบคุมการใช้งานตัวแปร การเรียกใช้ฟังก์ชั่นแกละการส่งค่ากลับ, counter บันทึกจุดที่กำลังรัน (execution thread)

พื้นที่ว่างระหว่าง heap และ stack จะทำให้ระบบปฏิบัติการ สามารถใช้งานออบเจ็คทั้งสองได้อย่างมีประสิทธิภาพ (dynamically)

Kernel context จะควบคุมและใช้งานโดยเคอร์เนลเท่านั้น เพื่อเก็บข้อมูลที่จำเป็นสำหรับติดตามการทำงานของโพรเซส หรือเพื่อเริ่มรัน หยุดรันโปรเซส เช่นตำแหน่งของโพรเซส ขนาดของ

โพรเซส โพรเซสโดยทั่วไปไม่สามารถจะแอ็คเซสข้อมูลในส่วนนี้ได้ ข้อมูลจะเก็บไว้ในโครงสร้างข้อมูลของโพรเซส (Process Table) เช่น

- process identifiler หรือ pid ปรกติจะมีค่าตั้งแต่ 2-32000 (หมายเลขโปรเซสของระบบ จะมีค่าตั้งแต่ 1-100 ,reserved pid) ดังนั้นหมายเลขโพรเซสที่ใช้งานจึงมีค่า ตั้งแต่ 101
- parent process identifier (PPID), Real User ID(USER ID),Real Group ID(group ID), ลำดับความสำคัญของโพเซส, เทอร์มินอล, สถานะของโปรเซสในขณะนั้น

4.4.2.1 การสร้างโพรเซสขึ้นมาใหม่

ถ้าต้องการให้โปรแกรมทำงานมากกว่าหนึ่งโปรเซส ในเวลาเดียวกัน ก็ต้องสร้างโปรเซสอื่นขึ้นมาใหม่ เช่น init ที่ทำงานเมื่อตอนบูต, ในขณะที่ฟังชั่นตระกูล exec จะเป็นเพียงการเปลี่ยนชุดของคำสั่งที่จะรันในโปรเซส

โดยการใช้ฟังก์ชั่นระดับต่ำ fork เพื่อสร้างโพรเซสลูกใหม่ โปรเซสที่สร้างขึ้นใหม่จะมีคุณสมบัติเหมือนกับโปรเซสต้นฉบับทุกประการ แต่จะมีค่า PID เป็นของตนเอง และมีเนื้อที่ในการทำงานเป็นของตนเอง เช่น หน่วยความจำ เวลาในการใช้งานซีพียู สภาวะในการทำงาน มีรูปแบบการใช้งานดังต่อไปนี้

```
#include  <sys/types.h>
#include  <unistd.h>

pid_t  fork(void);
```

ฟังชั่นจะส่งค่ากลับเป็น PID ของโพรเซสลูกที่สร้างขึ้นมาใหม่ และรันงานต่อไปได้ เพียงแต่ถ้าโพรเซสลูกเรียก fork อีกครั้งจะส่งค่ากลับเป็น 0

ถ้า fork ทำงานไม่สำเร็จจะส่งค่ากลับเป็น −1 ซึ่งอาจจะเนื่องจากจำนวนโปรเซสลูกที่จำกัดไว้ ที่ตัวแปล CHILD_MAX ในกรณีนี้ error จะกำหนดให้เป็น EAGAIN แต่ถ้าโครงสร้างข้อมูลที่เก็บรายละเอียดของโพรเซสในระบบเต็ม (process table) จะกำหนดให้เป็น ENOMEM

โปรแกรม seefork.c ต่อไปนี้จะแสดงการใช้ฟังชั่น fork()

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main(){
        pid_t     pid;
        char *message;
        int n;
```

```
printf ("fork program starting\n");
pid = fork();
switch(pid){
        case -1:        exit(1);
        case  0:        message = "This is child";
                        n = 5;
                        break;
        default:        message = "This is parent";
                        n =3;
                        break;
}
for(; n>0;n--){
        puts(message);
        sleep(1);
}
exit(0);
}
```

โปรแกรม จะสร้างโพรเซสใหม่ โดยแบ่งการทำงานออกเป็น 2 โพรเซส โพรเซสลูกจะพิมพ์ข้อความ 5 ครั้ง แต่โพรเซสต้อนฉบับจะพิมพ์ข้อความเพียง 3 ครั้ง ดังต่อไปนี้

```
S       ./seefork
fork program starting
This is parent
This is child
This is parent
This is child
This is parent
This is child
S This is child
This is child
```

โปรเซสต้นฉบับจะทำงานเสร็จก่อน ดังนั้นจะเห็นว่ามีเครื่องหมาย $ อยู่หน้าข้อความ เนื่องจากจบการรันโปรเซสต้นฉบับ ในขณะที่โปรเซสลูกกำลังรันอยู่

หลังจากใช้ฟังก์ชั่นนี้ ในการสร้างโปรโซสใหม่ ออบเจ็คต่างๆ ต่อไปนี้ในโพรเซสลูก จะยังไม่เปลี่ยนค่า

- real user ID(uid)
- real group ID(gid)
- effective user ID
- effective group ID
- process group ID
- root directory, current working directory
- creation mask
- การตอบสนองต่อสัญญาณต่างๆ

ส่วนค่าต่อไปนี้ จะเปลี่ยนไปจากโพรเซสต้นฉบับ

- (unique) process ID
- parent process ID
- มี file descriptor เป็นของตัวเอง (จะคัดลอกมาจากโปรเซสต้นฉบับ)

โดยทั่วไปฟังก์ชั่น fork จะใช้งานในกรณีที่ โพรเซสต้องการที่จะเอ็กซ์คิวท์ โปรแกรมใหม่เนื่องจากวิธีการสร้างโพรเซส จะมีอยู่เพียงวิธีเดียวเท่านั้น คือการใช้ฟังก์ชั่น fork ดังนั้นโปรเซสจะต้อง fork เพื่อสร้างโพรเซสใหม่ จากนั้นค่อยใช้ฟังชั่น exec เพื่อรันโปรแกรม

4.5 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 (ซึ่งในระบบที่ใช้เป็นเป็นเคอร์เนล 2.4.3 ซึ่งสามารถตรวจดูลำดับของข้อมูลได้ใน source code ของเคอร์เนลที่ /usr/src/linux/fs/proc/array.c) ซึ่งความหมายของแต่ละตัวแปรดูได้ตามตารางที่ 4.1

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

ตารางที่ 4.1 แสดงความหมายของข้อมูลในไฟล์ /proc/[number]/stat

ตัวแปร	ความหมาย
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 4.1 แสดงความหมายของข้อมูลในไฟล์ /proc/[number]/stat (ต่อ)

10. ไฟล์ status

เก็บข้อมูลของโพรเซสเช่นเดียวกันโดยแสดงให้อยู่ในรูปที่เข้าใจได้ง่าย แต่ข้อมูลน้อยกว่าในไฟล์ stat

4.6 บัฟเฟอร์ในการทำงานของอุปกรณ์(Character Device)

ในระบบที่ทำการสร้างขึ้นมานี้จะใช้วิธีการนี้ในส่งผ่านข้อมูลระหว่างโพรเซส โดยจะทำงานในลักษณะของโมดุล คือ สามารถจะโหลดได้เมื่อเคอร์เนลต้องการใช้งาน (modularized) ทำงานบนหน่วยความจำ เสมือนอุปกรณ์จริง (Memory-based)

อุปกรณ์ที่เป็นแคแรกเตอร์(Charracter) จะแทนด้วยชื่อไฟล์ ซึ่งปกติจะเก็บไว้ในไดเร็กทอรี /dev โดยเมื่อใช้คำสั่ง ls –l ส่วนเพอร์มิสชั่นของไฟล์จะขึ้นต้นด้วยอักษร “C” ระบุว่าเป็นอุปกรณ์ชนิดแคแรกเตอร์

เคอร์เนลจะใช้หมายเลข Major เพื่อกำหนดไดรเวอร์ที่จะใช้กับอุปกรณ์ เช่น /dev/null และ /dev/zero จะใช้ไดร์เวอร์หมายเลข 1 เป็นต้น

อันที่จริงแล้ว การเขียนดีไวซ์ไดร์เวอร์บนลีนุกซ์เป็นเพียงการเขียนโมดุลบางอย่าง และลงทะเบียนกับระบบไฟล์ VFS เพื่อแจ้งให้เคอร์เนลทราบ เมื่อมีการแอ็คแซสอุปกรณ์ดังกล่าว VFS จะได้เรียกไปยังรูทีนของอุปกรณ์ได้ถูกต้อง

การเพิ่มไดร์เวอร์เข้าไปในระบบทำได้โดยใช้ฟังก์ ซึ่งกำหนดไว้ในไฟล์ <linux/fs.h> ดังต่อไปนี้

```
int  register_chrdev(unsigned int major, const char *name,struct file_operation *fops);
```

- major เป็นหมายเลข major สำหรับอุปกรณ์ ถ้ากำหนดค่าเป็นศูนย์ระบบจะหาค่า major ที่จะไม่ได้ใช้งานให้
- name ชื่อของอุปกรณ์
- fops เป็นพอยน์เตอร์ชี้ไปยังโครงสร้างข้อมูล file_operation ซึ่งเป็นรูทีนของอุปกรณ์

ฟังก์ชั่นจะส่งค่ากลับเป็นจำนวนเต็มลบถ้ามี Error เกิดขึ้น ศูนย์หรือจำนวนเต็มบวกถ้าทำงานเสร็จ สำหรับไฟล์ของอุปกรณ์ name จะสร้างขึ้นมาโดยใช้ฟังชั่น mknod โดยปรกติจะเก็บไว้ที่ไดเร็กทอรี /dev โดยคำสั่งจะใช้อาร์กิวเมนต์ 3 อย่างดังตัวอย่างต่อไปนี้

```
mknod   /dev/schd0 c  127  0
```

เป็นการสร้างไฟล์ดีไวซ์ไดรเวอร์ของอุปกรณ์แคแรกเตอร์ (C) ซึ่งมีหมายเลข minor ควรจะมีค่าระหว่าง 0- 255 เนื่องจากเคอร์เนลเวอร์ชั่นปัจจุบัน จะใช้เพียง 1 ไบต์เก็บข้อมูล minor

หลังจากฟังชั่นทำงานสำเร็จแล้ว หมายเลข major ของอุปกรณ์จะเก็บไว้ในโครงสร้างข้อมูล device_struct ซึ่งมีดัชนีเป็นหมายเลข major เรียกว่าเวคเตอร์ chrdevs ภายในโครงสร้างข้อมูล device_struct จะมีพอยน์เตอร์ชี้ไปยังรูทีนการทำงานของอุปกรณ์ที่ระบุในโครงสร้างข้อมูล file_operation ดังต่อไปนี้

```
static struct file_operations auditsuccess_fops =
{       THIS_MODULE,                /* struct module *owner;*/
        NULL,                   /* skeleton_llseek */
        auditsuccess_read,       /* skeleton_read */
        NULL,                     /* skeleton_write */
        NULL,                   /* skeleton_readdir */
        NULL,                   /* skeleton_poll */
        NULL,             /* skeleton_ioctl */
        NULL,                   /* skeleton_mmap */
```

```
        auditsuccess_open,      /* skeleton_open */
        NULL,                   /* skeleton_flush */
        auditsuccess_release,           /* skeleton_release */
        NULL,                   /* skeleton_fsync */
        NULL,                   /* skeleton_fasync */
        NULL,                   /* skeleton_lock */
        NULL,                   /* skeleton_readv */
        NULL                    /* skeleton_writev */
#endif
};
```

ข้างบนเป็นตัวอย่างของโครงสร้างขอมูล file_operation ที่ใช้งานจริงในระบบนี้ ซึ่งกำหนดให้มีรูทีนในการทำงานอย่างคือ Open, release,Read โดยเมื่อแอพพลิเคชั่นต้องการใช้ไฟล์ก็จะเปิดไฟล์โดยใช้ major number เป็นดัชนีที่ชี้ไปยังโครงสร้างข้อมูล

การลบโมดูลออกจากระบบ ฟังชั่นต่อไปนี้จะยกเลิกหมายเลข major ที่ใช้งาน เมื่อมีการยกเลิกหมายเลข major จะว่างไดรเวอร์อื่นสามารถนำไปใช้งานอีก โดยปรกติจะเรียกจาก cleanup_module

```
Int   unregister_chrdev (unsigned  int  major, const char *name);
```

เคอร์เนลจะเปรียบเทียบหมายเลข major และชื่อ name ในกรณีที่ไม่ตรงกันหรือหมายเลข major มีค่ามากกว่าค่าที่กำหนดให้ จะส่งค่ากลับเป็น –EINVAL ในกรณีที่ไม่สามารถ จะใช้ unreigster โมดูลได้ โปรเซส /proc/devices จะแสดงข้อผิดพลาดเมื่ออ่านครั้งต่อไปเรียกว่า “Oops” เป็นข้อความที่แสดงเมื่อเคอร์เนลพยายามจะแอ็คเซสแอดเดรสที่ไม่ถูกต้อง

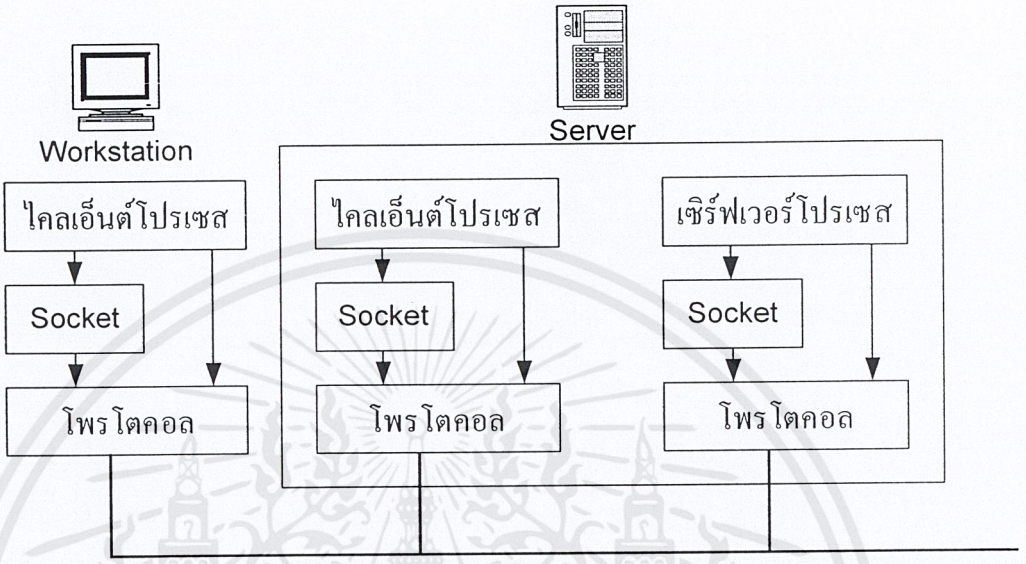
ในกรณีที่ลบโมดูลออกจากระบบ โดยไม่มีการใช้ฟังชั่น unregister ก่อนจะเกิดเหตุการณ์ที่คาดเดาได้ยาก เนื่องจากหมายเลข major จะไม่มีชื่อ node ของอุปกรณ์มาเปรียบเทียบใน strcmp

นอกจากนี้ต้องไม่ลืมที่จะลบไฟล์ node ของอุปกรณ์ออกจากระบบด้วย เนื่องจากอาจมีผลต่อการกำหนดหมายเลข major แบบไม่คงที่ครั้งต่อไป

4.7 ซ็อกเก็ต (Socket)

เป็นกลไกลที่ทำให้โปรแกรมเมอร์สามารถแอ็คเซสเน็ตเวิร์กโปรโตคอล เพื่อให้โปรเซสสามารถติดต่อกันแบบ โลคอลผ่านเน็ตเวิร์กได้ (Network Application Programming Interface) ยูทิลิตี้ของยูนิกซ์ที่ทำงานผ่านเน็ตเวิร์ก ปกติจะใช้ socket เช่น rlogin หรือ ftp

แนวทางการสร้างและใช้งานซ็อกเก็ตจะแตกต่างกับไปปีเนื่องจากมีการระบุโพรเซสไคลเอ็นต์และเซิร์ฟเวอร์อย่างชัดเจน นอกจากนี้สำหรับโพรเซสเซิร์ฟเวอร์เพียงโพรเซสเดียว ปกติแล้วสามารถทำงานกับโพรเซสไคลเอ็นต์หลายโพรเซสได้



รูปที่ 4.4 การติดต่อระหว่างไคลเอ็นต์เซิร์ฟเวอร์ผ่านซ็อกเก็ต

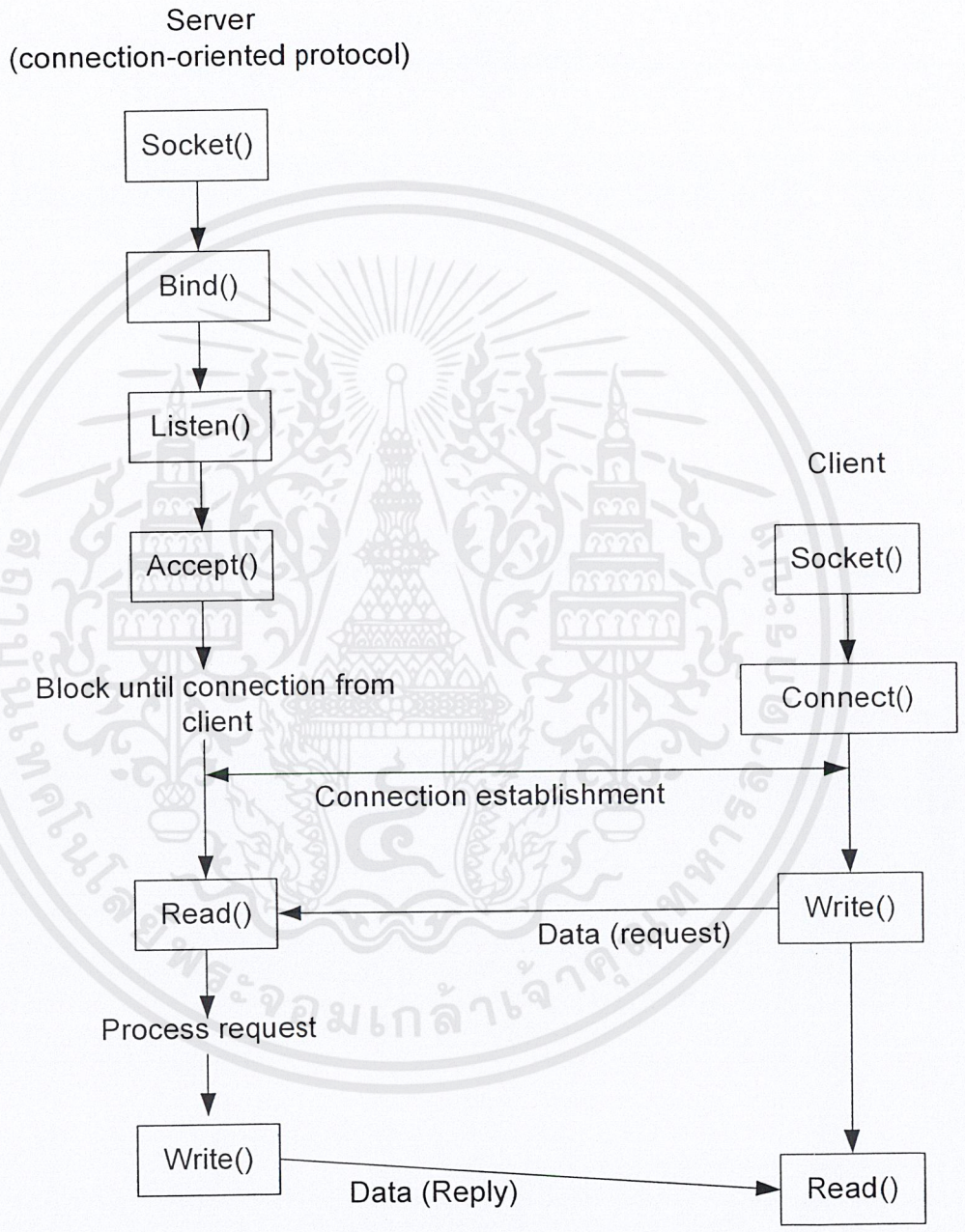
4.7.1 หลักการเบื้องต้นการติดต่อโดยใช้ซ็อกเก็ต (Connection Oriented)

สิ่งแรกที่จะต้องทำคือ เซิร์ฟเวอร์จะต้องสร้างซ็อกเก็ตซึ่งถือว่าเป็นรีซอร์ส ของระบบอย่างหนึ่ง ที่ควบคุมโดยโพรเซสเซิร์ฟเวอร์

ลำดับต่อไปก้คือกำหนดชื่อให้กับซ็อกเก็ตโดยปกติ local socket จะเป็นชื่อไฟล์ที่กำหนดไว้ใน /tmp หรือ /usr/tmp แต่ถ้าเป็น network socket จะเป็นออบเจ็คกำหนดจุดเชื่อมต่อของโพรเซสอื่นประกอบไปด้วย หมายเลขพอร์ตและจุดเชื่อมต่อของโพรเซส (port number/access point) ซึ่งจะเป็นค่าเฉพาะสำหรับเน็ทเวิร์คนั้น การกำหนดชื่อให้กับโปรเซสทำได้โดยใช้ฟังก์ชัน bind

จากนั้นโพรเซสเซิร์ฟเวอร์จะรอรับการเชื่อมต่อจากโพรเซสไคลเอ็นต์โดยใช้ซ็อกเก็ตที่กำหนดชื่อไปข้างต้น โดยใช้ฟังก์ชั่น listen เพื่อสร้าง queue รอรับข้อมูลการเชื่อมต่อหลังจากนั้นถ้ามีไคลเอ็นต์โพรเซสติดต่อเข้ามาโพรเซสเซิร์ฟเวอร์จะตอบรับการติดต่อโดยใช้ฟังก์ชัน accept ซึ่งจะเป็นการสร้างซ็อกเก็ตขึ้นมาใหม่ ซ็อกเก็ตใหม่นี้จะเป็นซ็อกเก็ตสำหรับติดต่อกับไคลเอ็นต์โพรเซสโดยเฉพาะ สำหรับซ็อกเก็ตที่กำหนดชื่อโดยใช้ฟังก์ชัน bind ก็ยังสามารถจะนำไปใช้งานกับไคลเอ็นต์โพรเซสอื่นได้เช่นกันและโดยปกติแล้วโพรเซสเซิร์ฟเวอร์หนึ่งโพรเซส สามารถให้บริการโพรเซสไคลเอ็นต์ได้หลายโพรเซส

สำหรับซ็อกเก็ตบนฝั่งไคลเอ็นต์จะสร้างง่ายกว่า คือใช้ฟังก์ชัน socket แล้วเรียกฟังก์ชัน connect เพื่อสร้างแชนเนลการเชื่อมต่อกับเซิร์ฟเวอร์โพรเซส หลังจากสร้างแชนเนลการติดต่อได้แล้ว สามารถใช้หมายเลขเพื่อแทนแชนเนลในการเชื่อมต่อนั้น ๆ โดยสามารถจะใช้แชนเนลสื่อสารข้อมูลระหว่างโพรเซสได้ทั้ง 2 ทาง



รูปที่ 4.5 ลำดับการติดต่อระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์

โปรแกรมตัวอย่าง client_proc.c จะแสดงการใช้ socket ที่ยังไม่ได้กำหนดชื่อ และทำการเชื่อมต่อกับ socket ของเซิร์ฟเวอร์ , server_socket

```
                    /* ส่วนเริ่มต้นของไฟล์ */
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<sys/un.h>
#include<unistd.h>
int main()
{  int  sock_fildes;
    int  addr_len;
    struct  sockadd_un  address;
    int connect_res;
    char ch= 'm';
                    /* ส่วนสร้าง socket สำหรับไคลเอ็นต์โพรเซส */
    sock_fildes = socket(AF_UNIX, SOCK_STREAM, 0);
                    /* ตั้งชื่อให้กับ socket */
    address.sum_family = AF_UNIX;
    strcpy(address.sun_path, "server_socket")
    addr_len  = sizeof(address);
                    /* เชื่อมต่อกับ socket ของเซิร์ฟเวอร์โพรเซส */
    connect_res = connect(sock_fildes, (struct  sockaddr, *)&address, addr_len);
    if(connect_res == -1)
      {  perror("oops: client1");
         exit(1);
          }
                    /* แอ็คเซสข้อมูลผ่าน sock_fildes */
    write(sock_fildes, &ch, 1);
     read(sock_fildes, &ch, 1);
    printf("char from server = %c\n",ch);
    close(sock_fildes);
    exit(0);
}
```

โปรแกรมเซิร์ฟเวอร์ server_proc.c จะรอรับการเชื่อมต่อจากไคลเอนต์ โดยการสร้าง socket กำหนดชื่อให้กับ socket หลังจากนั้นสร้าง queue เพื่อรอรับข้อความร้องขอการเชื่อมต่อและยอมรับการเชื่อต่อนั้น

```
/* ส่วนเริ่มต้นของไฟล์ */
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<sys/un.h>
#include<unistd.h>          int main()
{  int server_sock_fildes, client_sock_filde;
   int server_addrlen, client_addrlen;
   struct  sockaddr_un server_address;
   struct  sockaddr_un client_address;
                  /* ลบ socket เก่าออกแล้วสร้าง socket ใหม่ */
   unlink("server_socket");
   server_sock_fildes = socket(AF_UNIX,  SOCK_STREAM, 0);
                  /* กำหนดให้ socket เป็นของโพรเซสโดยใช้ฟังก์ชัน bind */
   server_address.sun_family = AF_UNIX;
   strcpy(server_address.sun_path, "server_socket");
   server_addrlen = sizeof(server_address);
   bind(server_sock_fildes, (struct  sockaddr *)&server_address, server_addrlen);
                  /* สร้าง queue เพื่อรับคำร้องขอการเชื่อมต่อและ รอการเชื่อมต่อ */
   listen(server_sock_fildes, 5);
   while(1)
   { char ch;
     printf("server waiting\n");
                  /* ยอมรับการเชื่อมต่อ */
     client_sock_fildes = accept(server_sock_fildes, (struct sockaddr *)&client_address,
                                    &client_addrlen);
          /* ใช้ฟังก์ชัน read/write เพื่อเขียนข้อมูลไปยังไคลเอนต์โพรเซส client_sock_fildes */
     read(client_sock_fildes, &ch, 1);
     ch++;
     write(client_sock_fildes, &ch, 1);
    close(client_sock_fildes);
  }}
```

โปรแกรมเซิร์ฟเวอร์ในตัวอย่างจะทำงานกับโปรแกรมไคลเอนต์ ทีละหนึ่งโปรแกรมเท่านั้น โดยจะอ่านตัวอักษรจากไคลเอนต์ เพิ่มค่าขึ้นไปทีละ 1 และเขียนค่ากลับไปยังไคลเอนต์ ระบบจะรอจนกว่าจะ

ทำงานให้ไคลเอนต์ปัจจุบันจนเสร็จ จึงจะทำงานกับไคลเอนต์ถัดไปซึ่งถือว่า เป็นเรื่องที่ยอมรับกันไม่ได้ ในระบบที่ใช้งานจริงเซิร์ฟเวอร์จะสามารถจัดการกับงานได้ทีละหลาย ๆ ไคลเอนต์ ในเวลาเดียวกัน

4.7.2 ชนิดของซ็อกเก็ต (Socket Type)

แต่ละโดเมนของซ็อกเก็ต อาจจะมีหลาย ๆ socket type ซึ่งหมายถึงวิธีการส่งข้อมูลของแต่ละโดเมนไม่เพียงแต่ AF_UNIX ซึ่งเราสามารถจะใช้การติดต่อได้ทั้ง 2 ทาง โดเมนของเน็ตเวิร์ก AF_INET จะมีวิธีการส่งข้อมูล 2 วิธีคือ stream(TCP) และ datagrams(UDP)

วิธีการแรก จะมีความน่าเชื่อถือของการส่งข้อมูลมากกว่า จะสร้างแชนเนลของการเชื่อมต่อจนกระทั่งการติดต่อนั้นเสร็จสิ้น(connectio-orient) รวมทั้งสามารถจะติดต่อกันได้ทั้ง 2 ทาง ข้อมูลที่ถูกส่งไปจะได้รับการรับรอง ติดตามผลการส่ง ว่าส่งไปถึงหรือไม่ ถ้ามีปัญหา ก็ต้องส่งใหม่, ถ้าข้อมูลมีขนาดใหญ่เกินไปจะแบ่งออกเป็นส่วนเล็ก ๆ และไปประกอบกันอีกทีที่ปลายทาง, stream socket กำหนดโดย SOCK_STREAM ในโดเมน AF_INET

ส่วนอีกวิธีหนึ่งคือ datagrams กำหนดโดยใช้ SOCK_DGRAM จะไม่สร้างแชนเนลในการเชื่อมต่อ (connectionless) หรือแม้แต่การจัดการลำดับของข้อมูลที่จะรับ ส่งเสร็จแล้วก็ไม่มีการติดตามผลการส่งข้อมูล ไม่มีการตรวจสอบใด ๆ ทั้งสิ้น นอกจากนี้ยังมีการจำกัดขนาดของข้อมูลที่จะส่ง ถูกกำหนดไว้ในโดเมน AF_INET โดยใช้โพรโตรคอล UDP/IP ในการติดต่อ

อย่างไรก็ตามในแง่ของการใช้รีซอร์ส จะไม่สิ้นเปลืองเวลาของระบบมากนักเนื่องจากไม่ต้องจองแชนเนลใช้งาน โดยทั่วไปจะใช้สำหรับการส่งข้อมูลบางอย่าง ที่ต้องทำเป็นประจำ (single-short nquiries)

	AF_UNIX	AF_INET
SOCK_STREAM	YES	TCP
SOCK_DGRAM	YES	UDP

ซ็อกเก็ตโพรโตคอล (Socket Protocols)

รายละเอียดของซ็อกเก็ต ที่เกี่ยวข้องกับตระกูลของโพรโตคอลบนระบบเน็ตเวิร์กเปิดของระบบยูนิกซ์(AF_INET) แบบ connection oriented(TCP/IP) และสำหรับติดต่อกันบนคอมพิวเตอร์เครื่องเดียวกัน(AF_UNIX) เท่านั้น ดังตารางที่ 4.2

ตระกูลของโพโตคอล	วิธีการส่งข้อมูล	ชื่อที่ใช้งานใน socket	โพรโตคอลที่ใช้งานจริง
AF_INET	SOCK_DGRAM	IPPROTO_UDP	UDP
AF_INET	SOCK_STREAM	IPPROTO_TCP	TCP
AF_INET	SOCK_RAW	IPPROTO_ICMP	ICMP
AF_INET	SOCK_RAW	IPPROTO_RAW	(raw)

ตารางที่ 4.2 รายละเอียดของซ็อกเก็ต

บทที่ 5
การออกแบบและการสร้างโปรแกรม

5.1 ทฤษฎีที่เกี่ยวข้อง

5.1.1 Hamming distance

Hamming distance เป็นจำนวนของตำแหน่งที่แตกต่างกันในสองข้อความ ที่มีความยาวเท่ากัน ตัวอย่างเช่น

Ex1

ระหว่าง 1011101 กับ 1001001 ค่า Hamming distance เท่ากับ 2

Ex2

ระหว่าง 2143896 and 2233796 ค่า Hamming distance เท่ากับ 3

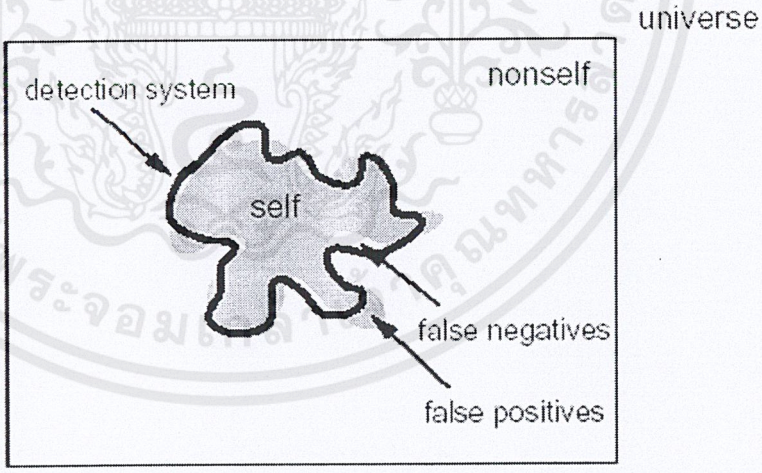
Ex3

AATGCGT

TAGGCTT

*A**G**T -- Hamming distance เท่ากับ 5

5.1.2 ประเภทของความผิดพลาดในระบบตรวจจับผู้บุกรุก (Classification Error)



รูปที่ 5.1 การแสดงลักษณะของความผิดพลาดในระบบตรวจจับผู้บุกรุก

ถ้าเราสมมุติให้เซตของสตริงทั้งหมด ความยาว L เป็น เซตของ U ซึ่งสามารถแบ่งออกเป็นสองสับเซตคือ คือ S (Self) และ N (Nonself) จะได้ว่า $U=S \cup N$ และ $S \cap N = \varnothing$

ส่วนที่อยู่ในสับเซต S(Self) คือเป็นส่วนที่ถือว่ามีความปลอดภัยหรือเป็นปกติ (Normal) แส่วนที่อยู่ในสับเซตของ N(Nonself) ถือว่าไม่มีความในปลอดภัยหรือไม่ปกติ (Anomalous) และส่วนที่อยู่ในเส้นทึบสีดำคือส่วนที่ระบบป้องกันผู้บุกรุกสามารถตรวจจับได้

ชนิดของความผิดพลาดจะมีอยู่ 2 ชนิดคือ false negatives และ false positive เราสามารถแบ่งความความแตกต่างของความผิดพลาดทั้งสองชนิดได้ดังนี้

- false positives เป็นสตริงในส่วนที่มีความปลอดภัย(Self) แต่ละถูกระบบตรวจจับบอกว่าเป็นสิงที่ผิดปกติ(Anomalous)
- false negatives เป็นสตริงในส่วนที่มีความไม่ปลอดภัย(Nonself) แต่ระบบตรวจจับบอกว่าเป็นสิ่งที่ปกติ(Normal)

5.2 แนวความคิดในการออกแบบระบบ

5.2.1วิธีการที่เลือกใช้สร้างระบบตรวจจับผู้บุกรุก

วิธีการที่จะกล่าวถึงนี้คือ การตรวจจับผู้บุกรุกแบบตรวจจับพฤติกรรมที่ผิดปกติ โดยสร้างโพไฟล์ของพฤติกรรมที่ปกติสำหรับโปรแกรมที่เราสนใจ มี 2 ลำดับในการตรวจจับพฤติกรรมที่ไม่ปกติ อันดับแรกเราสร้างโพไฟล์หรือฐานข้อมูลของพฤติกรรมที่ปกติ จากนั้นจะใช้ฐานข้อมูลในการแสดงพฤติกรรมของโพรเซสสำหรับตรวจจับสิ่งผิดปกติ ซึ่งระบบนี้เป็นการเก็บซีเควนต์ของซีสเต็มคอลหลังจากรันโปรแกรม แล้วพิจารณาที่ความปกติ กับความผิดปกติที่เกิดขึ้น

ระบบคอมพิวเตอร์ถูกรบกวนเนื่องจากมีระบบความปลอดภัยที่ไม่ดี เช่น การเกิดบัฟเฟอร์โอเวอร์โฟลว์ในระบบยูนิกซ์ หรือ บักในอินเตอร์เน็ทเอ็กซ์พลอเรอร์ ของไมโครซอฟต์ แอพพลิเคชัน และ ระบบปฏิบัติการหลายอย่างมีระบบความปลอดภัยที่บกพร่อง ระบบความปลอดภัยที่ดีควรจะสามารถที่จะแบ่งแยกความแตกต่าง เช่น ปัญหาต่างๆ รวมไปถึงวิธีการใช้งานโปรแกรมที่หลากหลาย และ การสร้างซอฟต์แวร์ที่ผิดกฎหมายมาใช้งาน

เมื่อระบบความปลอดภัยทำงานได้ตามแผนที่กำหนดอย่างสมบูรณ์และถูกต้องตามคอนฟิกที่ได้กำหนดเอาไว้ ถึงแม้จะเป็นการตั้งสมมติฐานขึ้นมา แต่มันก็เป็นแนวความคิดที่จะเป็นไปได้ ระบบคอมพิวเตอร์ไม่ได้อยู่คงที่ จะมีการเปลี่ยนแปลงไปเรื่อย ๆ โดยผู้ขาย ผู้บริหารระบบ และผู้ใช้งาน การเพิ่มหรือลบ โปรแกรม และการคอนฟิก ดังนั้นเราจะต้องสร้างระบบที่มั่นใจได้ว่ามีความมั่นคงและถูกต้องอยู่เสมอในช่วงเวลาที่มีการเปลี่ยนแปลง มีรูปแบบที่ไม่ขึ้นอยู่กับการเปลี่ยนแปลงระบบ นอกจากความมั่นใจแล้ว การใช้งานโปรแกรม เช่น การเข้ารหัส (encryption) การควบคุมการติดต่อ (access control) ไฟวอลล์ และ ออดิต(Audit)การติดตาม ทั้งหมดนี้สามารถทำการหลอกลวงได้ การสร้างระบบความปลอดภัยที่สมบูรณ์นั้นอาจเป็นไปได้ยาก แต่ถ้ามีการศึกษาค้นคว้าก่อนถ้าเรายอมรับในระบบรักษาความปลอดภัยของเรา วิธีการหรือส่วนของโครงสร้างที่เกิดความผิดพลาด เราก็ต้องยอมรับมันด้วย เราสามารถพัฒนาเครื่องมือรักษาความปลอดภัยที่เราใช้อยู่เช่น ระบบตรวจจับการบุกรุก ซึ่งระบบตรวจจับการบุกรุกนี้ใกล้เคียงระบบความปลอดภัยพื้นฐานบน สมติฐานที่คิดเอาไว้

มีหลายวิธีการที่ระบบตรวจจับการบุกรุกจะใช้ในการตรวจหาความผิดปกติ เป็นส่วนสำคัญสำหรับโพรไฟล์(profile) ของพฤติกรรมที่ปกติ ในระบบยูนิกซ์ การรันโพรเซส(หรือโปรแกรม)จะมีการกำหนดระดับของสิทธิเมื่อมีการเรียกใช้จากผู้ใช้ อย่างไรก็ตามโพรเซส สามารถเปลี่ยนระดับของสิทธิเป็นซูเปอร์ยูสเซอร์ได้ (ผู้ใช้ที่มีสิทธิสูงสุด) หนึ่งในปัญหาของความปลอดภัยในระบบยูนิกซ์เกี่ยวกับระดับสิทธิของโพรเซสคือ โพรเซสต้องการสถานะซูเปอร์ยูสเซอร์ ในการใช้งานทรัพยากร (resource) ของระบบ แต่เมื่อเกิดกรณี program error ในส่วนของโค้ดหรือมีการคอนฟิกผิดพลาด ขณะที่โพรเซสรันอยู่ในสิทธิของซูปเปอร์ยูสเซอร์ ผู้ใช้ปกติก็สามารถเปลี่ยนสิทธิไปเป็นซูเปอร์ยูสเซอร์โดยอาศัยช่องโหว่ของปัญหานั้นได้

การตรวจจับสิ่งผิดปกติของโพรเซสที่มีสิทธิพิเศษนี้จะพิจารณาโปรแกรมที่ทำการติดตามและกำลังทำงานอยู่และทำการเตือน หากมีการเปลี่ยนแปลงโปรแกรม โดยพิจารณาโปรแกรมที่เป็นแบล็กบ็อกซ์ เราไม่ต้องการที่จะรู้ข้อมูลหรือฟังก์ชันภายในของโปรแกรม เราสามารถสรุปได้จากสิ่งที่เห็น โดยปกติสิ่งที่เราเห็นของโพรเซสในระบบยูนิกซ์จะเรียกว่าซีสเต็มคอลล์ เพราะว่าโพรเซสของระบบยูนิกซ์จะติดต่อทรัพยากรของระบบโดยใช้ซีสเต็มคอลล์

ระบบตรวจจับผู้บุกรุกที่ทำการสร้างขึ้นนี้ จะเป็นการเก็บซีเควนต์ของซีสเต็มคอลหลังจากรันโปรแกรม แล้วพิจารณาที่ความปกติ กับความผิดปกติที่เกิดขึ้น ปัญหาที่พบคือจะเก็บซีเควนซ์ของซีสเต็มคอลอย่างไร ความยาวของซีเควนต์เป็นเท่าไร จากนั้นเราจะตรวจสอบพฤติกรรมที่เป็นปกติขึ้นซึ่งอยู่กับขนาดของซีเควนซ์ของซีสเต็มคอลล์ที่เราติดตาม

5.2.2 โพรไฟล์ที่เก็บพฤติกรรมปกติ (Profile Normal Behavior)

5.2.2.1 อัลกอริทึมที่ใช้สร้างฐานข้อมูล

อัลกอริทึมที่ใช้สร้างฐานข้อมูลปกติที่ง่ายที่สุด จะใช้วิธีสแกนตามลำดับของซีสเต็มคอลล์โดยโปรแกรมที่สร้างขึ้นมา แล้วสร้างฐานข้อมูลที่เก็บลำดับที่ไม่ซ้ำกันของซีเควนซ์ทั้งหมดมีความยาวเป็น k ดังนั้นแต่ละโปรแกรมที่สนใจจะมีฐานข้อมูลที่แตกต่างกัน

เป็นวิธีที่ยุ่งยากมากบนระบบยูนิกซ์เพราะโปรแกรมเราสามารถเรียกใช้ได้มากกว่าหนึ่งโปรแกรม โพรเซสจะถูกสร้างขึ้นมาด้วยซีสเต็มคอล์ฟอร์ค หรือ วีฟอร์ค ความแตกต่างของซีสเต็มคอล์ฟอร์คทั้งสองอย่างคือ ซีสเต็มคอล์ฟอร์คจะสร้างโพรเซสขึ้นมาโดยทำการก็อปปี้โพรเซสเดิม แต่วีฟอร์คจะทำการสร้างโพรเซสขึ้นมาแทนที่โพรเซสเดิมที่มีอยู่โดยไม่มีการเปลี่ยนโพรเซสไอดี ในที่นี่จะทำการติดตามเฉพาะซีสเต็มคอล์ฟอร์คเท่านั้น และติดตามส่วนที่เกี่ยวข้องที่เป็นปกติเราไม่ติดตามวีฟอร์คเพราะว่าวีฟอร์คจะเอ็กซีคิวซ์โปรแกรมใหม่

นี่คือตัวอย่างการสร้างฐานข้อมูลปกติ จะตรวจได้จากการติดตามซีสเต็มคอลล์

open, read, mmap, mmap, open, read, mmap

ถ้าเรากำหนดขนาดของสไลด์ดิงวินโดว์ให้มีความยาวเป็น k แต่ละซีเควนซ์มีความยาวเป็น k จะไม่ซ้ำกันจากตัวอย่างนี้เรากำหนดให้มีความยาวเป็น 3 ดังนั้นเราจะได้ซีเควนซ์ที่ไม่ซ้ำกันคือ

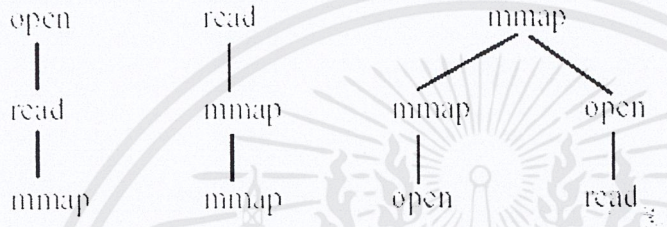
open, read, mmap

read, mmap,mmap

mmap,mmap,open

mmap,open,read

นี่คือการเก็บซีเควนซ์ในรูปของทรี โดยรูตของทรีจะเป็นซีเควนซ์แต่ละอันและเซ็ตของทรีจะสอดคล้องกับซีเควนซ์ดังรูปที่.....



รูปที่ 5.2 ตัวอย่างของซีสเต็มคอลที่เก็บในรูปของทรี

จากฐานข้อมูลดังกล่าวเป็นการเก็บข้อมูลแบบไบนารีทรี จากทรีที่เห็นข้างบนเราจะพบว่าเมื่อมีลำดับของซีสเต็มคอลล์ใหม่เข้ามาเป็น

mmap, ioctl,open

mmap, read,fstat64

จะเป็นไปไม่ได้ที่เราจะเก็บลำดับของซิสเต็มคอลล์ดังกล่าวลงในทรีข้างต้น ดังนั้นเราจึงเพิ่มข้อกำหนดใหม่บางประการ เพื่อให้เก็บลำดับของซิสเต็มคอลที่จะเข้ามาใหม่ได้ทั้งหมด ดังนี้

1. โหนดลูกที่อยู่ทางซ้ายของโหนดพ่อ ให้ถือว่าเป็นซีเควนซ์ที่มีลำดับของซิสเต็มคอลก่อนหน้าคือโหนดพ่อและไล่ลำดับกลับไปจนถึงรูทโหนด
2. โหนดลูกที่อยู่ทางขวาของโหนดพ่อ ให้ถือว่าเป็นคนละซีเควนซ์กับโหนดพ่อ แต่เป็นซีเคนซ์ที่มีลำดับของซิสเต็มคอลก่อนหน้าเป็นพ่อของโหนดพ่อและไล่ลำดับกลับไปจนถึงรูทโหนด

เพื่อความเข้าใจเราจะแสดงการเก็บข้อมูลลงทรี โดยหากเรามีลำดับของซิสเต็มคอลดังนี้

mmap,mmap,open

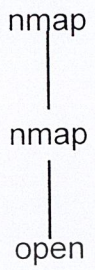
mmap,open,read

mmap, ioctl,open

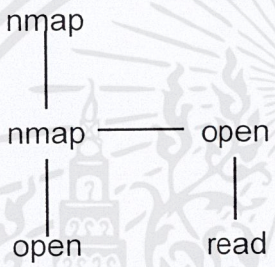
mmap, open,fstat64

เมื่อเรานำลำดับทั้งหมดมาสร้างเป็นทรี เราจะได้ทรีในแต่ละครั้งที่ใส่ข้อมูลดังนี้

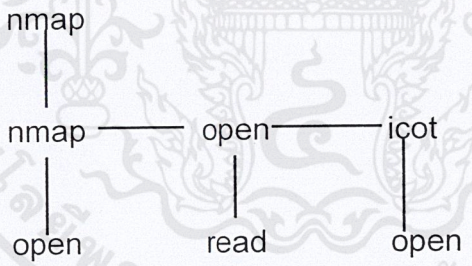
ลำดับที่1 เมื่อใส่ mmap,mmap,open



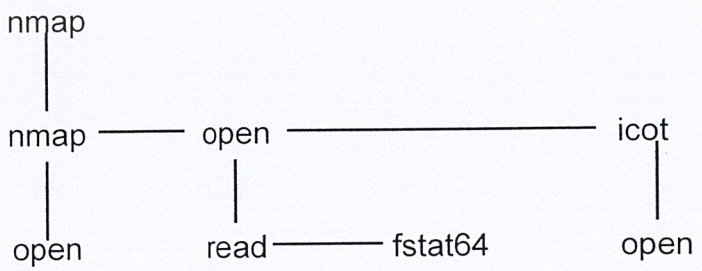
ลำดับที่ 2 เมื่อใส่ mmap,open,read



ลำดับที่3 เมื่อใส่ mmap, ioctl,open



ลำดับที่ 4 เมื่อใส่ mmap, open,fstat64



จะเห็นว่าทรีดังกล่าวเป็นทรีที่เกิดจากรูทโหนดที่มีซิสเต็มคอลเดียวกัน ดังนั้นในการสร้างเป็นฐานข้อมูลเราต้องสร้างเป็นอาเรย์ของทรีโดยขนาดของอาเรย์จะขึ้นอยกับจำนวนของซิสเต็มคอลทั้งหมด ในความเป็นจริงเราจะแทนลำดับของซิสเต็มคอลด้วยตัวเลขก่อนนำข้อมูลเข้าไปเก็บในทรีเพื่อประโยชน์ของการเปรียบเทียบและขนาดของฐานข้อมูล

5.2.2.2 การจัดเก็บไฟล์ฐานข้อมูล

ในส่วนของการบันทึกข้อมูลในทรีดังกล่าวลงไฟล์เพื่อเก็บไว้เป็นฐานข้อมูล ได้มีการออกแบบรูปแบบของไฟล์ดังนี้

1. บรรทัดแรกของไฟล์จะเป็นตัวบอกความยาวของลำดับซิสเต็มคอลอยู่ในไฟล์
2. บรรทัดถัดมาจะเป็นรูทโหนดของทรีและตามด้วยข้อมูลที่อยู่ในทรี สลับกันจนบันทึกข้อมูลในฐานข้อมูลจนหมด(ฐานข้อมูลคืออาเรย์ของทรี)
3. ตัวเลขที่เป็นค่าลบจะเป็นค่าของจำนวนโหนดที่ต้องนับย้อนขึ้นมาก่อนนำซิสเต็มคอลถัดไปมาเรียงลำดับต่อกัน
4. บรรทัดสุดท้ายจะเก็บ “-1” เพื่อบอกจุดสิ้นสุดของฐานข้อมูล

ตัวอย่างของการเก็บฐานข้อมูลตั้งแต่ตรวจจับซิสเต็มคอลและการเก็บข้อมูลลงไฟล์มีดังนี้

สมมุติว่าลำดับของซิสเต็มคอลที่ทำการตรวจจับได้มีดังนี้

open, read, mmap, mmap, open, read, mmap,ioctl,open,read

แทนลำดับของซิสเต็มคอลดังกล่าวด้วยตัวเลข

1, 2, 3, 3, 1, 2, 3, 4, 1, 2

หาลำดับของซิสเต็มคอลความยาว k ที่ไม่ซ้ำกันจะได้

1, 2, 3

2, 3, 3

2, 3, 4

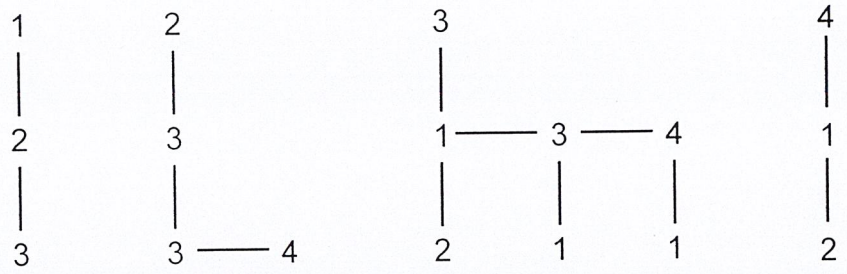
3, 1, 2

3, 3, 1

3, 4, 1

4, 1, 2

นำลำดับของซิสเต็มคอลดังกล่าวไปสร้างเป็นทรีตามหลักการที่กล่าวแล้วข้างต้นจะได้อาเรย์ของทรีทั้งหมดดังนี้



นำทรีดังกล่าวมาเก็บของเป็นไฟล์ฐานข้อมูลจะได้รูปแบบของไฟล์ดังนี้

```
#Seq_length: 3
1
1  2  3  -1
2
2  3  3  -2  4  -1
3
3  1  2  -3  3  1  -3  4  1  -1
4
4  1  2  -1
-1
```

รูปที่ 5.3 รูปแบบของไฟล์ฐานข้อมูล

5.2.3 การวัดพฤติกรรมที่ผิดปกติ (Measuring Anomalous Behavior)

เมื่อมีฐานข้อมูลของพฤติกรรมปกติ ต่อมาจะใช้วิธีการที่ได้สร้างฐานข้อมูลในการตรวจสอบการติดตามพฤติกรรมใหม่ โดยการมองที่ซีเควนซ์ทั้งหมดที่ความยาว k และมีการกำหนดความอันตรายของความผิดปกติ เช่น จำนวนของความผิดพลาดที่เกิดขึ้นจากการติดตามพฤติกรรมที่ผิดปกติ โดยจะรายงานจำนวนที่ทำผิดกฎ และ เปอร์เซ็นต์ต่อจำนวนการใช้งานทั้งหมดที่ได้จากการติดตาม ในสภาวะที่สมบูรณ์เราต้องการฐานข้อมูลที่ปกติที่เก็บความแตกต่างทั้งหมดของพฤติกรรมที่ปกติ แต่ไม่ต้องการเก็บทุกเหตุการณ์ที่เป็นไปได้ของพฤติกรรมที่ถูกกฎ โดยถือว่าพฤติกรรมที่อยู่ในรูปแบบที่ใกล้เคียงกันจะเป็นสับเซตของพฤติกรรมที่ถูกกฎ เราไม่ได้ต้องการตรวจจับการบุกรุกเพียงอย่างเดียวแต่นอกเหนือจากนั้นเราต้องการแสดงปัญหาของระบบด้วย เช่น โพรเซสกำลังรันอยู่มันอาจจะเอ็กซีคิวผิดพลาดทำให้ได้ผลลัพธ์ออกมานอกเหนือจากซีเควนซ์ที่มีอยู่ เราจะบอกได้ว่าเป็นการกระทำที่ผิดกฎ ไม่ใช่บอกว่าเป็นการกระทำที่ผิดปกติ

ถ้าฐานข้อมูลไม่ได้เก็บความแตกต่างของพฤติกรรมทั้งหมดหรือเมื่อเราพบเหตุการณ์ของซีเควนซ์โดยบังเอิญที่ไม่มีในฐานข้อมูลปกติ เราก็สรุปได้ว่าเป็นพฤติกรรมที่ผิดปกติ เราสามารถพิจารณา

ความไม่ถูกต้องครั้งเดียวเป็นการจำความผิดพลาดได้ ในความจริงแล้ว เหมือนกับว่าจะเป็นไปไม่ได้ที่เราจะเก็บพฤติกรรมที่เป็นปกติที่แตกต่างกันทั้งหมด ดังนั้นเราต้องมองถึงว่าฐานข้อมูลปกติของเราจะไม่สมบูรณ์และครอบคลุมพฤติกรรมปกติทั้งหมด วิธีการหนึ่งที่นับจำนวนความผิดพลาดที่เกิดขึ้นจากการติดตาม อย่างไรก็ตามนี่คือปัญหา เพราะการนับขึ้นอยู่กับขนาดของส่วนที่เราสนใจ (ความยาวของซีเควนซ์)

เราสามารถเลือกใช้ แฮมมิ่งดิสแตนซ์(Hamming distance) ระหว่างซีเควนซ์ที่จะทำการวัด ถึงแม้ว่าตัวเลือกของเราจะมีให้เลือกตามใจชอบ แต่เราไม่สามารถพิสูจน์ได้ทันทีจากการวัด ดังนั้นเราจึงกำหนดค่าของมันโดยการสังเกต

เราใช้แฮมมิ่งดิสแตนซ์ ระหว่างสองซีเควนซ์เพื่อคำนวนความแตกต่างของซีเควนซ์ที่เข้ามาใหม่กับซีเควนซ์ปกติที่มีอยู่จริง ระหว่าง 2 ซีเควนซ์ ความคล้ายคลึงกันสามารถคำนวนโดยใช้กฎการสัมพันธ์กัน อย่างไร กฎความสัมพันธ์ในที่นี้เป็นพื้นฐานของแฮมมิ่งดิสแตนซ์ ความแตกต่างระหว่าง 2 ซีเควนซ์ i และ j แสดงโดยแฮมมิ่งดิสแตนซ์ d(i,j) สำหรับแต่ละซีเควนซ์ i ที่เข้ามาใหม่ เรากำหนดค่าต่ำสุดของแฮมมิ่งดิสแตนซ์เป็น $d_{min}(i)$ ระหว่าง i และซีเควนซ์ปกติ

$$d_{min}(i) = \min\{d(i,j) \text{for all normal sequences } j\}$$

d_{min}เป็นค่าที่แสดงความอันตราย นี่เป็นการวัดที่ไม่ขึ้นอยู่กับความยาวของซีสเต็มคอลล์ที่เราติดตาม ตัวอย่างของการคำนวณค่าความแตกต่างแสดงได้ดังนี้

open, read, mmap, mmap, open, read, mmap

สามารถสร้างเป็นฐานข้อมูลปกติได้

open, read, mmap

read, mmap,mmap

mmap,mmap,open

mmap,open,read

ถ้าเราติดตามซีสเต็มคอลล์แล้วมีการเปลี่ยนแปลงไปเป็น

open, read, mmap, mmap, open, mmap, mmap

จะได้ซีเควนซ์ใหม่ที่ไม่มีอยู่ในฐานข้อมูลปกติคือ

mmap, open, mmap

open, mmap, mmap

มีความไม่สัมพันธ์กันอยู่สองแห่งคิดเป็น 40% ของการติดตาม และค่า d_{min} เท่ากับ 1 อย่างไรก็ตาม การจับสิ่งผิดปกตินั้นทำอย่างนี้เราต้องใช้เวลามากด้วย อัลกอริทึมทีจะใช้ยืนยันซีเควนซ์ที่ปกติ มีประสิทธิ

ภาพน้อยมาก ถ้ากำหนดระดับของสิ่งผิดปกติต่อสิ่งที่ปกติของซีเควนซ์ เป็น R_A ดังนั้นค่าเฉลี่ย คอมเพล็กซิตี้ ของการคำนวณ $d_{min}(i)$ ต่อซีเควนซ์เป็น

$N(k-1)R_A + (k-1)(1-R)$ จะได้เป็น $O(k(R_AN + 1))$

5.2.3.1 การแบ่งแยกความผิดพลาด (Classification Error)

การตรวจจับการบุกรุก อย่างน้อย 1 ซีเควนซ์ที่เกิดขึ้นจากการบุกรุก ต้องถูกแยกออกว่าผิดปกติในส่วนของการวัดของเรา อะไรที่เราต้องการอย่างน้อย 1 ซีเควนซ์ ที่เกิดจาการบุกรุกที่มี $d_{min} > 0$ เราวัดความอันตรายของความผิดปกติโดย d_{min} เราจะรายงานค่าสูงสุดของ d_{min} ที่เจอระหว่างการติดตาม เพราะว่าการแสดงความอันตรายของสัญญาณที่พบในการติดตาม เราจะคำนวณความผิดปกติให้เป็น S_A โดย

$$S_A = \max\{d_{min}(i) \text{ for all new sequence } i\}$$

ในตัวอย่างนอกจาก $S_A = 1$ โดยทั่วไปเราจะไม่รายงานค่าของ S_A จริง แต่เราจะรายงานในรูปแบบของ S'_A โดยในการเปรียบเทียบค่า S_A สำหรับค่าความแตกต่างของ k จะได้ S'_A เป็น

$$S'_A = S_A / k$$

นอกจากนั้นเราจะให้มีความผิดพลาดน้อยที่สุด แต่เราจะต้องยอมรับความผิดพลาดของ การแบ่งแยกที่ผิดพลาดมากกว่า การเตือนที่ผิดพลาดสามารถลดลงโดยเพิ่มลำดับชั้นของการป้องกัน ถึงแม้เราจะเพิ่มลำดับขั้นจะไม่ได้ลดค่าของการเตือนที่ผิดพลาดทั้งหมด ตัวอย่างที่จะแสดงให้เห็นคือ พิจารณาระบบที่มีการป้องกัน L ชั้นที่ผู้บุกรุกจะต้องทะลุผ่าน ซึ่งแต่ละชั้นจะมีความน่าจะเป็นที่ผู้บุกรุกจะรอดพ้นการจรวจจับเป็น P_n (P_n เป็นอัตราของการแบ่งแยกที่ผิดพลาด) ถ้าความน่าจะเป็นของการตรวจจับที่เกิดขึ้นแต่ชั้นที่ผู้บุกรุกจะรอดพ้นการตรวจจับเป็น P_n^L ดังนั้นอัตราของ neg รวมกันทั้งหมดจะเป็นเลขยกกำลัง จะลดลงโดยการเพิ่มจำนวนชั้นของการป้องกัน ถ้าเราสมมติว่า ความน่าจะเป็นของแต่ละชั้นเป็น P_f ของการเกิดการเตือนที่ผิดพลาด ดังนั้นจำนวนของการเกิด การเตือนที่ผิดพลาด เป็น P_f^L

5.3 โครงสร้างของระบบและลำดับการทำงาน

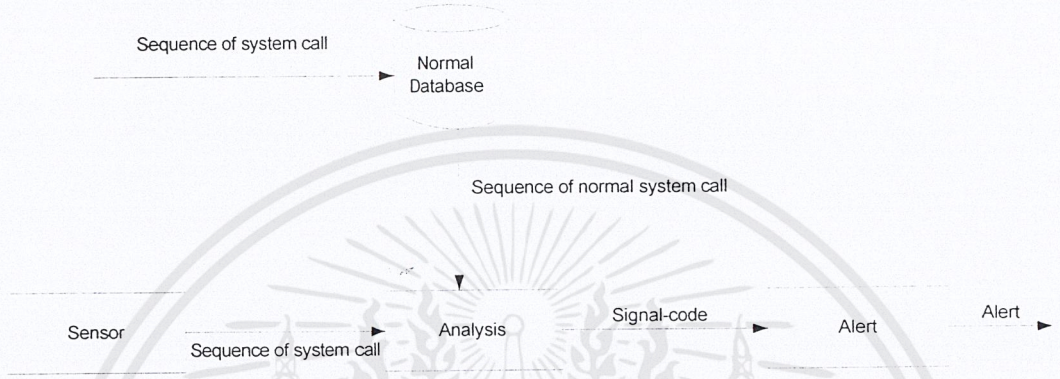
Input

ลำดับการทำงานของซิสเต็มคอลล์ของโปรแกรมที่เราต้องการตรวจสอบการทำงาน

Output

ข้อมูลการแจ้งเตือนผู้บุกรุก เช่น เมล์, ล็อกไฟล์

5.3.1 โครงสร้างของระบบ



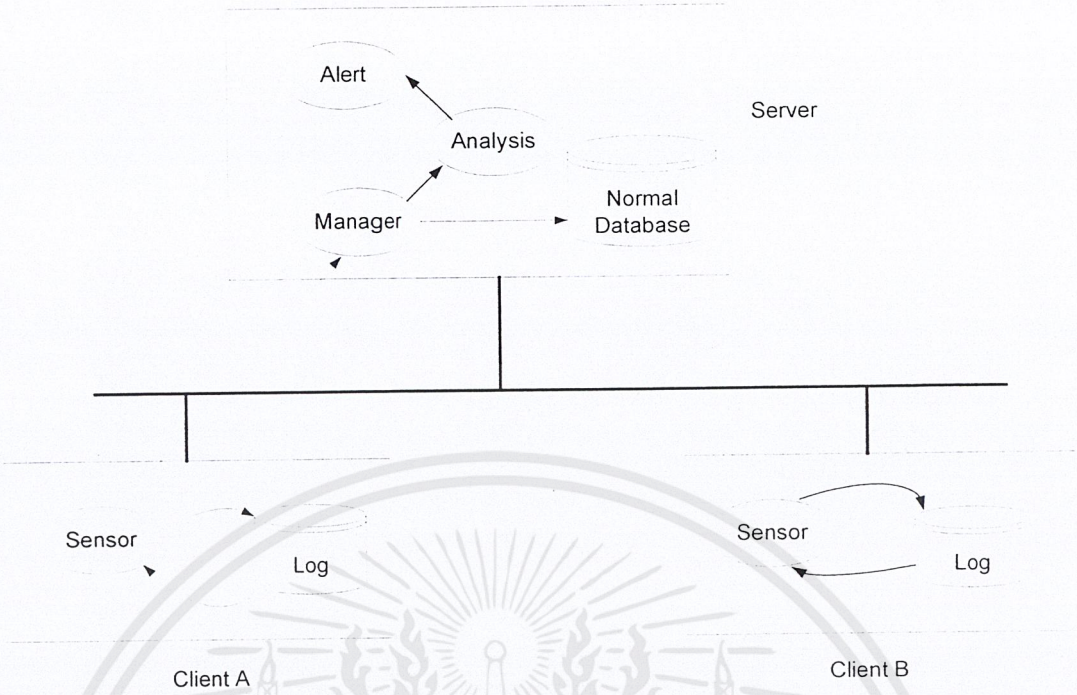
รูปที่ 5.4 รูป โครงสร้างของระบบ

- Sensor เป็นตัวตรวจดักจับซิสเต็มคอลล์ของโปรแกรมที่ต้องการ
- Analysis เป็นตัววิเคราะห์ข้อมูลว่าเป็นการบุกรุกหรือไม่
- Alert เป็นตัวแจ้งเตือนผู้บุกรุก
- Normal Database เป็นตัวเก็บฐานข้อมูลผู้ใช้งานที่ใช้งานในลักษณะที่ปกติ

5.3.2 การทำงานของระบบ

จากรูปโครงสร้างของโปรแกรมที่ผ่านมา เราจะนำมาทำงานในรูปแบบของไคลเอ็น –เซิร์ฟเวอร์ (Client – Server) สาเหตุที่เราทำงานในรูปแบบนี้เพราะว่า

- ปกติแล้วเมื่อเกิดการบุกรุกแล้วโฮสที่ถูกบุกรุกอาจไม่สามารถแจ้งเตือนผู้บุกรุกเองได้ จึงต้องอาศัยโฮสต์อื่นในการแจ้งเตือนให้
- เพิ่มความเร็วในการทำงานเพราะเราสามารถแยกตัววิเคราะห์ไปอยู่ในเครื่องที่มีประสิทธิภาพสูงได้
- ไม่เกิดความซ้ำซ้อนของการเก็บฐานข้อมูลผู้ใช้งานปกติ เพราะบางทีฐานข้อมูลอาจมีจำนวนมาก



รูปที่ 5.5 การใช้งานและการทำงานของโปรแกรม

จากรูปที่ 5.5 จะมีขั้นตอนการทำงานดังนี้คือ

1. เซิร์ฟเวอร์จะต้องทำงานก่อนเพื่อรองรับการติดต่อจากไคลเอ็น
2. เมื่อไคลเอ็นเริ่มทำงาน จะทำการการขอเชื่อมต่อกับเซิร์ฟเวอร์
3. ถ้าสำเร็จจะทำงานและส่งข้อมูลการติดตามการทำงานของโพรเซสไปให้เซิร์ฟเวอร์ทำการประมวลผล
4. ถ้าไม่สำเร็จ และไคลเอ็นมีฐานข้อมูลเป็นของตัวเอง ไคลเอ็นจะทำการประมวลผลและทำการเก็บข้อมูลจากการประมวลผลลงล็อกไฟล์
5. เมื่อเซิร์ฟเวอร์รับข้อมูลการติดตามการทำงานของโพรเซสมาจะตรวจสอบค่าคอนฟิกว่าให้เก็บข้อมูลหรือวิเคราะห์ข้อมูล
6. หากเป็นการวิเคราะห์ข้อมูล แล้วตรวจพบว่ามีการบุกรุกก็จะมีการแจ้งเตือนผู้ดูแลระบบและบันทึกข้อมูลลงล็อกไฟล์

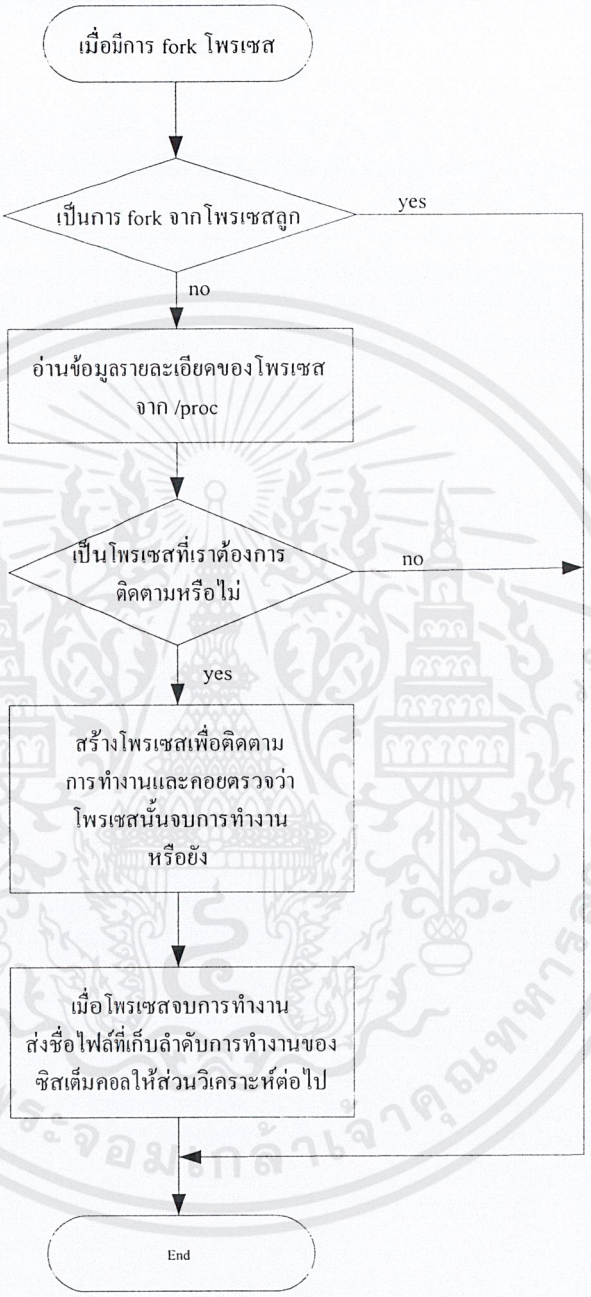
5.3.3 โครงสร้างของโปรแกรม

5.3.3.1 ส่วนของตัวตรวจจับการทำงานของซิสเต็มคอล(Sensor)

ในส่วนนี้จะแบ่งย่อยออกได้เป็นสองส่วนคือ ส่วนที่ติดการทำงานของโพรเซสที่เกิดจากการฟอก(Fork) เป็นส่วนที่จะต้องทำงานบนเคอร์เนล โดยจะทำเป็นโมดูลและทำการเพิ่มเข้าไปในเคอร์เนล อีกส่วนเป็นส่วนที่ฝังอยู่ในเชลล์เพื่อดักจับโพรเพซที่ถูกสั่งให้ทำงานโดยเชลล์

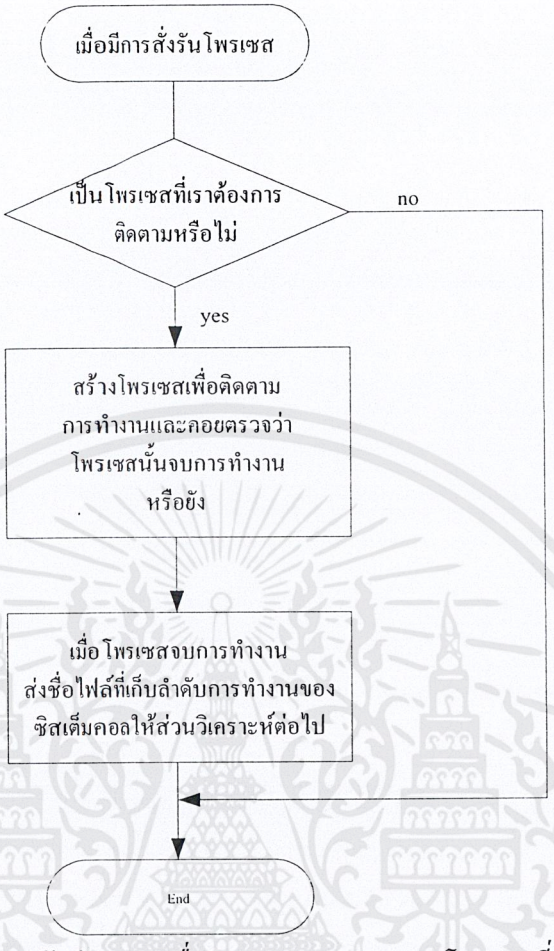
1. ส่วนของการติดตามการทำงานซิสเต็มคอลฟอก เราต้องทำการขัดจังหวะการทำงานซิสเต็มคอลฟอก(Intercept system call fork) โดยนำโมดูลของเราเข้าไปแทนที่ และต้องมีการรีจิสเตอร์ไฟล์

โอเปอเรเตอร์(File operator) เพื่อใช้ในการส่งผ่านข้อมูล จากนั้นเมื่อเกิดการฟอกโฟรเซสโมดูลของเราก็จะทำงาน โดยมีลำดับการทำงานดังนี้



รูปที่ 5.6 วิธีการสร้างโพรเซสเพื่อติดตามการทำงานของซิสเต็มคอลล์ฟอก

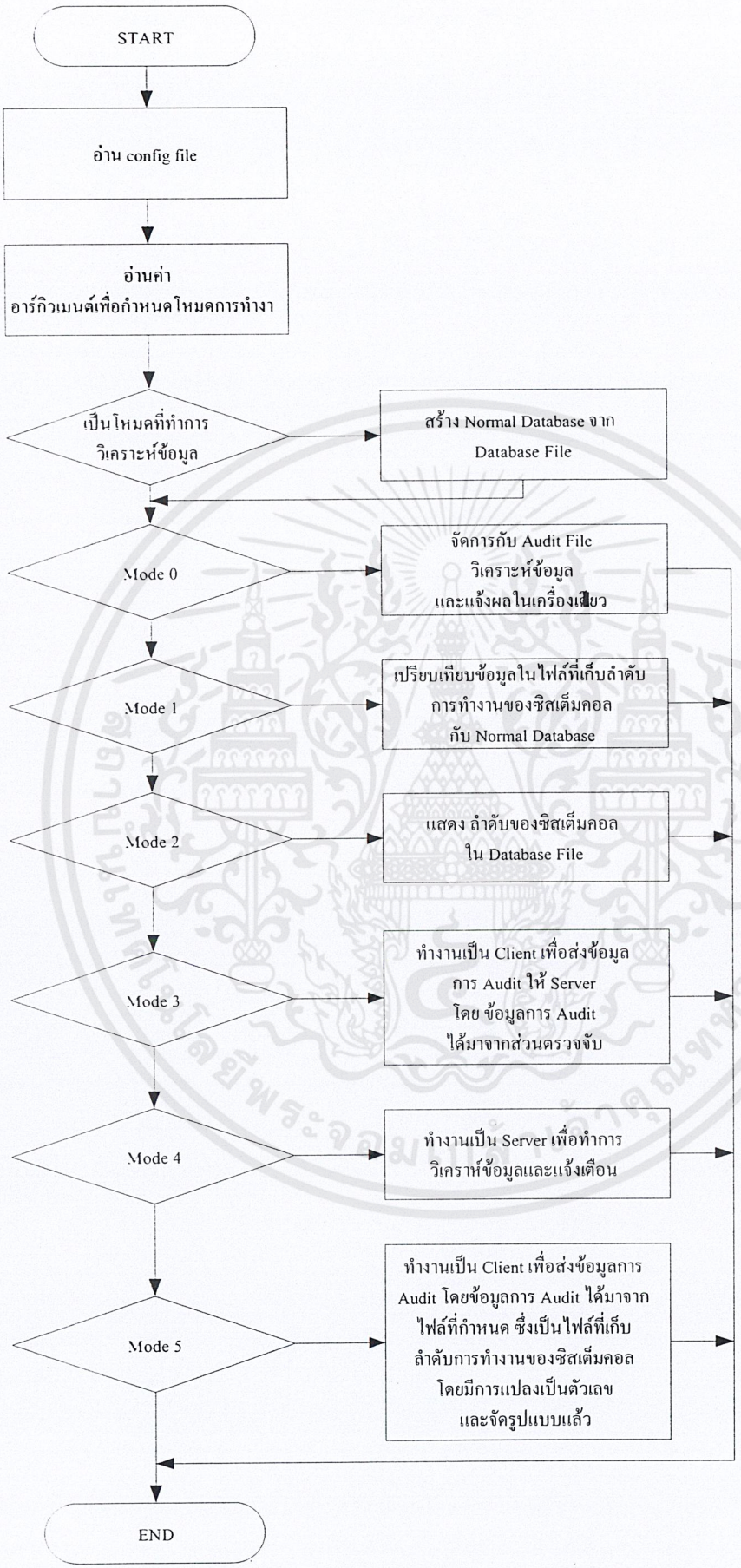
2. ส่วนติดตามการทำงานของโพรเซสที่สร้างโดยเชลล์ ในที่นี้จะเลือกใช้ Bash เพราะว่าเป็นเชลล์ที่ได้รับความนิยมสูงและมีการใช้งานกันมาก โดยดัดแปลงเพื่อให้มีการสั่งสร้างออติดโพรเซส(โพรเซสที่ใช้ติดตามการทำงาน) พร้อมๆกับสั่งสร้างโพรเซสที่ผู้ใช้งานต้องการ โดยมีลำดับการทำงานดังนี้คือ



รูปที่ 5.7 วิธีการสร้างโพรเซสเพื่อติดตามการทำงานของโพรเซสที่สร้างโดยเชลล์

5.3.3.2 ส่วนวิเคราะห์ข้อมูล(Analysis)

ในส่วนวิเคราะห์ การทำงานจะขึ้นอยู่กับโหมดที่กำหนด ค่าของโหมดจะขึ้นอยู่กับการกำหนดตัวแปลในคอนฟิกไฟล์และอาร์กิวเมนต์ที่ใส่ตอนสั่งรันโปรแกรม การทำงานโดยทั่วไปก็คือ อ่านค่าคอนฟิกไฟล์ สร้างดาต้าเบสถ้ามีความจำเป็นต้องใช้งาน โดยดาต้าเบสจะเป็นลิงลิสต์ของ ดาต้าเบสทุกโปรแกรมที่ต้องการจะติดตาม ดาต้าเบสของแต่ละโปรแกรมเก็บโดยใช้อาเรย์ของทรีขนาดของอาเรย์จะขึ้นอยู่กับจำนวนของซิสเต็มคอล และรูปแบบของทรีก็เป็นเช่นเดียวกับที่ได้อธิบายไว้แล้วในตอนต้น เมื่อทำการสร้างดาต้าเบสเสร็จโปรแกรมก็จะทำงานตามโหมดที่กำหนด โดยโฟชาร์ตการการทำงานคร่าวๆ มีดังนี้



รูปที่ 5.8 ลำดับการทำงานของส่วนวิเคราะห์ข้อมูล

5.3.3.3 ส่วนของการแจ้งเตือนและเก็บล็อกไฟล์

การแจ้งเตือนจะมีสองอย่างคือแจ้งเตือนผ่านเมล์และเก็บข้อมูลลงล็อกไฟล์ โดยข้อมูลที่มีการแจ้งเตือนและเก็บลงล็อกไฟล์คือ

UID	หมายเลขประจำตัวของผู้ใช้งานที่ใช้งานผิดปกติ
UNAME	ชื่อของผู้ใช้งาน
PID	หมายเลขโพรเซสที่ตรวจจับความผิดปกติได้
PROCESS NAME	ชื่อของโพรเซส
TIME	เวลาที่ทำการตรวจจับได้

5.3.3.4 ส่วนของการติดต่อและส่งผ่านข้อมูล

1. การส่งผ่านข้อมูลของโพรเซสในแต่ล่ะส่วนในส่วนนี้จะใช้วิธีการส่งผ่านข้อมูลโดยผ่านไฟล์ (file operation) ซึ่งในการใช้งานเราจะต้องเขียนไดร์เวอร์สำหรับไฟล์นี้ขึ้นมาด้วย

```
static struct file_operations auditsuccess_fops =
{
#if LINUX_VERSION_CODE < KERNEL_VERSION(2,3,0)
        NULL,                   /* skeleton_llseek */
        auditsuccess_read,      /* skeleton_read */
        NULL,                   /* skeleton_write */
        NULL,                   /* skeleton_readdir */
        NULL,                   /* skeleton_poll */
        NULL,                   /* skeleton_ioctl */
        NULL,                   /* skeleton_mmap */
        auditsuccess_open,      /* skeleton_open */
        NULL,                   /* skeleton_flush */
        auditsuccess_release,           /* skeleton_release */
        NULL,                   /* skeleton_fsync */
        NULL,                   /* skeleton_fasync */
        NULL,                   /* skeleton_check_media_change */
        NULL,                   /* skeleton_revalidate */
        NULL                    /* skeleton_lock */
#else /* for LINUX_VERSION_CODE 2.4.0 and later */
        THIS_MODULE,            /* struct module *owner;*/
```

```
        NULL,                   /* skeleton_llseek */
        auditsuccess_read,      /* skeleton_read */
        NULL,                   /* skeleton_write */
        NULL,
        NULL,                   /* skeleton_readdir */
        NULL,                   /* skeleton_poll */
        NULL,                   /* skeleton_ioctl */
        NULL,                   /* skeleton_mmap */
        auditsuccess_open,      /* skeleton_open */
        NULL,                   /* skeleton_flush */
        auditsuccess_release,   /* skeleton_release */
        NULL,                   /* skeleton_fsync */
        NULL,                   /* skeleton_fasync */
        NULL,                   /* skeleton_lock */
        NULL,                   /* skeleton_readv */
        NULL                    /* skeleton_writev */
#endif
};
```

รูปที่ 5.9 โครงสร้างของไฟล์โอเปอร์เรชั่น(ต่อ)

จากรูปเป็นการกำหนดโครงสร้างของไฟล์ (file operation) ซึ่งกำหนดให้อ่านได้อย่างเดียว เพื่อป้องกันการเขียนข้อมูลจากผู้บุกรุกซึ่งจะทำให้โปรแกรมทำงานผิดพลาดได้

การใช้งานไฟล์นี้จะเรียกใช้เหมือนกันการเขียนอ่านไฟล์ทั่วไปแต่ในความเป็นจริงแล้วเป็นการเขียนอ่านข้อมูลบนหน่วยความจำ โดยเราสามารถกำหนดโครงสร้างของข้อมูลที่เรา ทำการเขียนอ่านได้ ซึ่งโครงสร้างของข้อมูลที่ใช้ในโปรแกรมนี้คือ

```
struct fd_list
{
        long   fd;
        char * filename;
        struct fd_list * next;
};
```

รูปที่ 5.10 โครงสร้างของข้อมูล ไฟล์ที่ใช้ในการเก็บข้อมูลการติดตามโพรเซสในส่วนของเชลล์

จากรูปเป็นโครงสร้างของลิงค์ลิสที่ใช้เก็บข้อมูลของไฟล์ที่เกิดจากการติดตามโพรเซสในส่วนของเชลล์ และส่งผ่านข้อมูลนี้ไปยังส่วนวิเคราะห์ข้อมูลหรือส่งผ่านข้อมูลผ่านเน็ตเวิร์ก

```
struct fork_list
{
        int     id;
        long    pid;
        char    * pname;
        struct fork_list * next;
};
```

รูปที่ 5.10 โครงสร้างของข้อมูล ไฟล์ที่ใช้ในการเก็บข้อมูลการติดตามโพรเซสที่เกิดจากการฟอก(fork)

จากรูปเป็นโครงสร้างของลิงค์ลิสที่ใช้เก็บข้อมูลของไฟล์ที่เกิดจากการติดตามโพรเซสที่เกิดจากการฟอก(fork) และส่งผ่านข้อมูลนี้ไปยังส่วนวิเคราะห์ข้อมูลหรือส่งผ่านข้อมูลผ่านเน็ตเวิร์ก

2. การส่งผ่านข้อมูลของส่วนตรวจจับและส่วนวิเคราะห์ผ่านเน็ตเวิร์ก

ส่วนนี้เป็นการติดต่อและส่งผ่านข้อมูลโดยใช้ โพรโตคอล TCP/IP ในส่วนของเซิร์ฟเวอร์จะสามารถรองรับไคล์เอ็นได้หลายตัวการจัดการจับใครเอ็นหลายๆตัวจะทำโดยวิธีการ SELECT ข้อมูลที่ส่งผ่านเน็ตเวิร์กจะมีสองส่วนคือ ส่วนของรายเอียดของโพรเซสที่ทำการตรวจจับและส่วนของข้อมูลจริง(ข้อมูลลำดับการทำงานของซิสเต็มคอล) โดยข้อมูลทั้งสองจะมีโครงสร้างดังนี้

LEN	CMD	DATA

จากรูปเป็นโครงสร้างของเพ็กเก็ตที่ใช้ส่งผ่านข้อมูลลำดับการทำงานของซิสเต็มคอลระหว่างตัวตรวจจับและตัววิเคราะห์โดยความหมายของแต่ละส่วนมีดังนี้

LEN เป็นความยาวของแพ็กที่ส่งไปขนาดสูงสุดเท่ากับ 255

CMD เป็นคำสั่งที่กำหนดว่าแพ็กเก็ตนี้เป็นแพ็กเก็ตของอะไร

DATA เป็นข้อมูลที่เราจะส่งไป

UID	UNAME	HOSTNAME	PID	PROCESS NAME	TIME

จากรูปเป็นโครงสร้างเพ็กเก็ตที่ใช้ส่งข้อมูลรายละเอียดของโพรเซสที่ทำการติดตามระหว่างตัวตรวจจับและตัววิเคราะห์ โดยความหมายของแต่ละส่วนคือ

UID หมายเลขประจำตัวของผู้ใช้งานที่ใช้งานผิดปกติ

UNAME	ชื่อของผู้ใช้งาน
HOSTNAME	ชื่อของ Host
PID	หมายเลขโพรเซสที่ตรวจจับความผิดปกติได้
PROCESS NAME	ชื่อของโพรเซส
TIME	เวลาที่ทำการตรวจจับได้

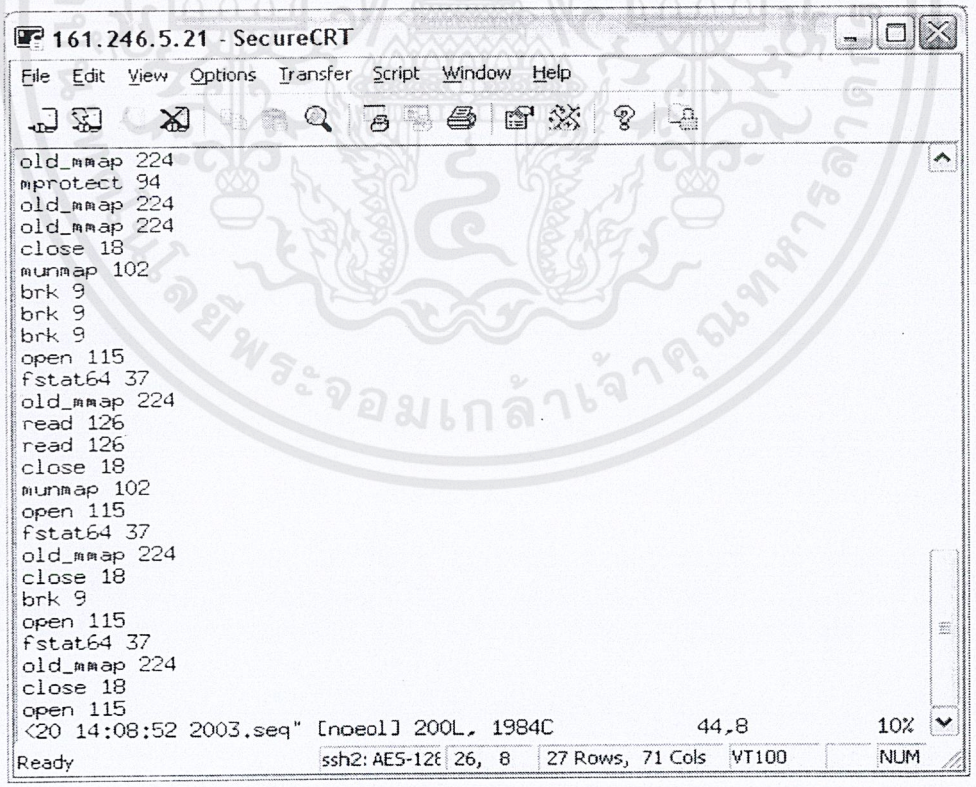
บทที่ 6
ตัวอย่างการทำงานและการทดสอบระบบ

ระบบตรวจจับผู้บุกรุกที่ถูกสร้างขึ้นมา เมื่อทำการติดตั้งระบบเสร็จเรียบร้อยแล้ว ขั้นแรกของการทำงานคือ หากเรายังไม่มีฐานข้อมูลลำดับการทำงานที่ปกติของโปรแกรมที่เราต้องการติดตามการทำงาน เราจะต้องสร้างฐานข้อมูลนี้ขึ้นมาก่อนโดยต้องแน่ใจว่าฐานข้อมูลที่เราทำการจัดเก็บ ไม่มีส่วนของลำดับการทำงานที่เกิดจากการบุกรุกหรือเกิดจากการทำงานที่ผิดพลาดของโปรแกรมที่เรากำลังติดตามรวมอยู่ด้วย

เมื่อเราแน่ใจว่าฐานข้อมูลที่เราทำการจัดเก็บพร้อมแล้วก็ ให้ทำการกำหนดให้โปรแกรมทำงานในโหมดตรวจสอบการบุกรุก และทดลองใช้งานโปรแกรมดู โปรแกรมจะต้องไม่มีการแจ้งเตือนหากไม่มีการบุกรุกจริงเกิดขึ้น

6.1 การตรวจจับลำดับการทำงานของซิสเต็มคอลและการเก็บข้อมูลลงฐานข้อมูล

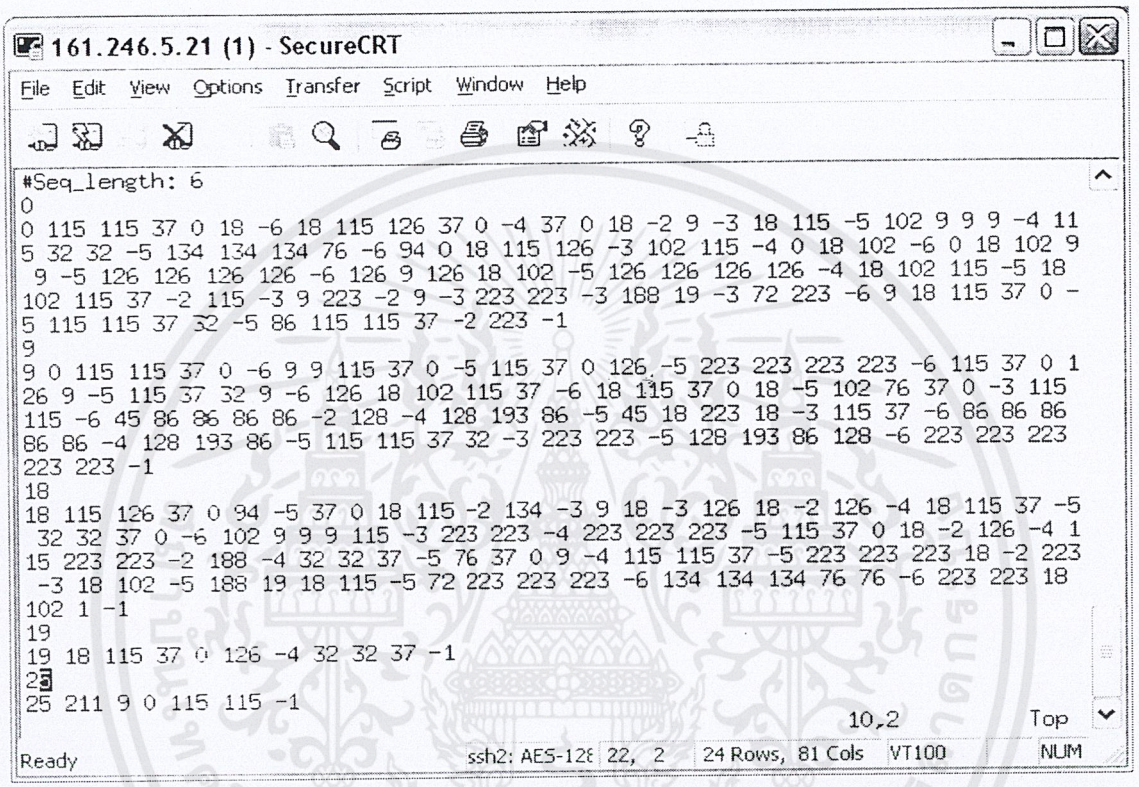
เมื่อเรากำหนดให้โปรแกรมในงานในโหมดจัดเก็บข้อมูลการทำงานของซิสเต็มคอลและมีการกำหนดให้เก็บข้อมูลการออดิต (audit) ไว้ด้วย เราจะได้ลักษณะของไฟล์ออดิตดังนี้



รูปที่ 6.1 ลักษณะของออดิตไฟล์ที่ได้จากการตัวตรวจจับการทำงานของซิสเต็มคอล

จากออดิตไฟล์ดังกล่าวตัวเลขด้านหน้าจะแทนชื่อของซิสเต็มคอล ซึ่งโปรแกรมที่เรากำลังติดตามเรียกใช้งาน ตัวเลขด้านหลังเป็นตัวเลขที่ใช้แทนชื่อซิสเต็มคอลดังกล่าวเมื่อเราเก็บข้อมูลลงฐานข้อมูลหรือทำการเปรียบเทียบ

เมื่อนำข้อมูลการออดิตดังกล่าวมาเก็บลงฐานข้อมูลจะได้ลักษณะของฐานข้อมูลดังนี้ (ฐานข้อมูลมูลนี้กำหนดค่าความยาวของลำดับการทำงานเท่ากับ 6)



รูปที่ 6.2 ลักษณะของฐานข้อมูลการทำงานที่ปกติ

จากรูปบรรทัดแรกจะบอกว่าไฟล์นี้เก็บฐานข้อมูลที่มีความยาวของลำดับการทำงานเป็นเท่าไร บรรทัดที่มีเลขอยู่ตัวเดียวหมายถึง เป็นรูทโหนดของทรีที่ขึ้นต้นด้วยซิสเต็มคอลนั้นๆ

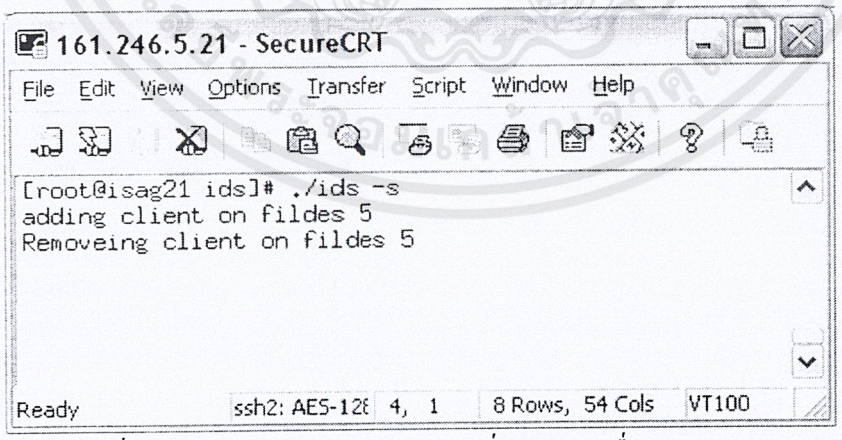
6.2 การทำงาน

โปรแกรมจะมีรูปแบบในการทำงานให้เลือกหลายรูปแบบโดยแต่ละรูปแบบจะมีการทำงานดังนี้

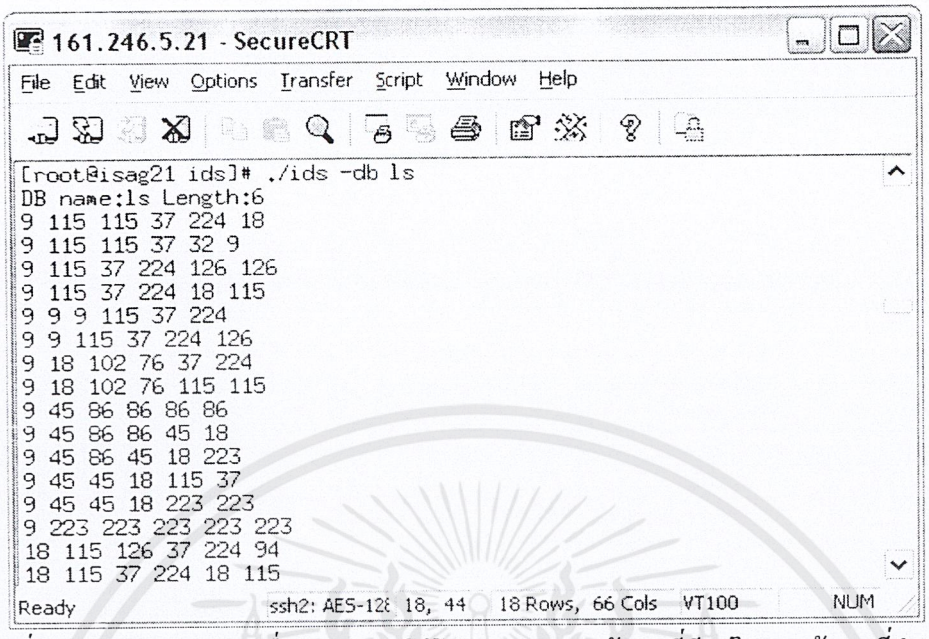
ids [-seq] [-db] [-c] [-s] [-sendseq] [-help]

ค่าออปชั่น	รูปแบบของการทำงาน
-seq *file_name*	หมายถึง ทำการเปรียบเทียบฐานข้อมูลลำดับการทำงานที่ปกติ (Normal Database) กับ ไฟล์ที่เก็บลำดับการทำงานของซิสเต็มคอล์ที่กำหนด
-db *process_name*	หมายถึง ให้แสดงฐานข้อมูลลำดับการทำงานที่ปกติ ของโปรแกรมที่กำหนด
-c	หมายถึง ทำงานเป็น IDS Client ทำหน้าที่ตรวจจับการทำงานของโปรแกรมที่กำหนดและส่งผลลัพธ์ของการตรวจจับไปวิเคราะห์หาความผิดปกติที่ IDS Server
-s	หมายถึง ทำงานเป็น IDS Server ทำหน้าที่วิเคราะห์หาความผิดปกติจากข้อมูลลำดับการทำงานของซิสเต็มคอลที่ได้มาจาก IDS Client
-sendseq *file_name*	หมายถึง ส่งไฟล์ที่เก็บลำดับการทำงานของซิสเต็มคอลตามที่กำหนดไปทำการเปรียบเทียบฐานข้อมูลลำดับการทำงานที่ปกติ (Normal Database) ที่ IDS Server
-help	หมายถึง แสดงรูปแบบการใช้งาน และความหมาย

ตารางที่ 6.1 แสดงค่าออปชั่นและความหมายของรูปแบบการทำงานของโปรแกรม



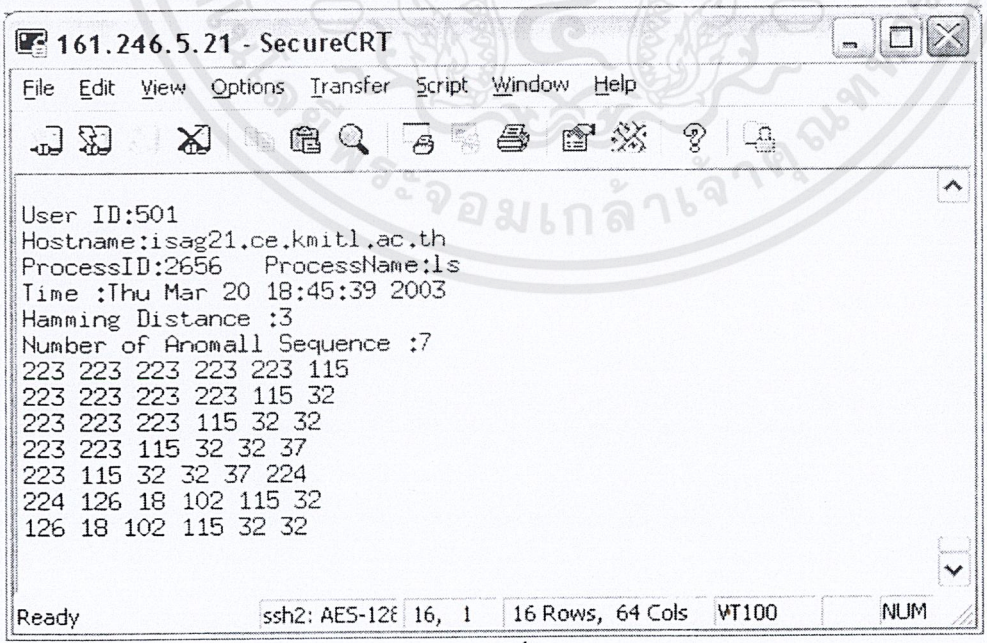
รูปที่ 6.3 การทำงานของ IDS Server เมื่อมีการขอเชื่อมต่อ จาก IDS Client



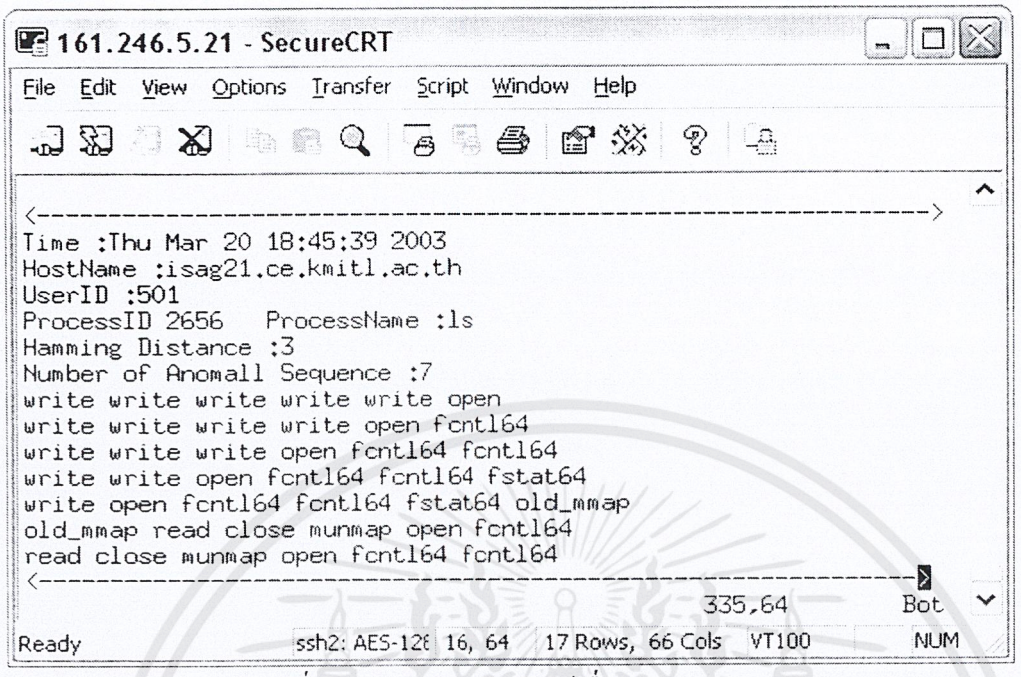
รูปที่ 6.4 แสดงการทำงานเมื่อเรากำหนดให้โปรแกรมแสดงข้อมูลที่มีอยู่ในฐานข้อมูลที่กำหนด

6.3 การวิเคราะห์และการแจ้งเตือน

รูปแบบของการแจ้งเตือนสามารถกำหนดให้แสดงผลบนหน้าจอด้วยโดยกำหนดให้ตัวแปล print_result_to_screen เป็น '1' แต่ปกติแล้วระบบเก็บข้อมูลการแจ้งเตือนลงล็อกไฟล์อย่างเดียว ล็อกไฟล์ที่จัดเก็บข้อมูลดังกล่าวคือ scids.log



รูปที่ 6.5 ลักษณะของการแจ้งเตือนเมื่อตรวจพบการบุกรุกบนหน้าจอคอมพิวเตอร์



รูปที่ 6.6 ลักษณะของล็อกไฟล์เมื่อระบบตรวจพบการบุกรุก

จากรูปดังกล่าวมีการรายงานลำดับการทำงานของซิสเต็มคอลที่ผิดปกติด้วย เพราะว่าเรากำหนดให้ตัวแปล report_anomall_seq เป็น '1' การกำหนดให้โปรแกรมคำนวณค่า Hamming Distance โดยการกำหนดให้ค่าตัวแปล hmd_on เป็น '1'

นอกจากนี้ยังมีการแจ้งเตือนผู้ดูแลระบบผ่านทางเมล์ด้วย โดยข้อมูลที่เมล์ไปหาผู้ดูแลระบบคือ วันที่ เวลา ชื่อของโปรแกรมที่เกิดการบุกรุก และ uid ของผู้ที่ทำการบุกรุก

6.4 การปรับแต่งการทำงานของระบบ

มีตัวแปลสองตัวในไฟล์ sc.config ซึ่งการกำหนดค่าของตัวแปลสองตัวนี้ต่างกันจะทำให้ระบบมีการแจ้งเตือนผลลัพธ์จากการวิเคราะห์ต่างกันด้วยคือ

seq_length เป็นตัวแปลที่ใช้กำหนดความยาวของลำดับการทำงานที่จะทำการตรวจสอบ

num_anomall_seq_alert ตรวจพบความผิดปกติเป็นจำนวนเท่าไรถึงจะทำการแจ้งเตือนผู้ดูแลระบบ

จากการทดลอง ค่าที่แนะนำคือ ให้ num_anomall_seq_alert เท่ากับ 5 และให้ seq_length เท่ากับ 6 จะทำให้ประสิทธิภาพของตัววิเคราะห์ทำงานได้ดีสุด

บทที่ 7
สรุปและวิจารณ์

วิเคราะห์ผลการทดลอง

จากการทดสอบเป็นที่น่าพอใจ เนื่องจากระบบสามารถตรวจทำการแจ้งเตือนได้หากโปรแกรมที่กำลังติดตามการทำงานมีการเปลี่ยนแปลงรูปแบบการทำงานของซิสเต็มคอลไปจากเดิม ดังนั้นระบบจะสามารถตรวจจับการบุกรุกได้ หากวิธีการในการบุกรุกนั้นทำให้ลำดับการทำงานของโปรแกรมเปลี่ยนไปจากเดิม เช่น บักของโปรแกรมซึ่งทำให้โปรแกรมเกิดลำดับการทำงานที่ไม่มีอยู่ในฐานข้อมูลลำดับการทำงานของซิสเต็มคอลปกติ และการตรวจจับก็สามารถแจ้งเตือนได้ทันทีเมื่อมีการบุกรุก ระบบสามารถป้องการตัวเองจากการถูกบุกรุกได้ในระดับหนึ่งเนื่องจากโฮสที่ทำการวิเคราะห์หาผู้บุกรุก เป็นคนละโฮสกับที่ผู้บุกรุกทำการบุกรุก ดังนั้นโอกาสที่ผู้บุกรุกจะทำการแก้ไขล็อกไฟล์หรือ ค่าการทำงานของระบบก็เป็นไปได้ยากขึ้นด้วย

ปัญหาและอุปสรรค

1. การเขียนโปรแกรมติดต่อกับเคอร์เนลโดยตรงค่อนข้างยุ่งยาก หากการเขียนโปรแกรมไม่รัดกุมพอจะทำให้คอมพิวเตอร์ทั้งระบบทำงานผิดพลาดได้

2. การติดต่อกันระหว่างส่วนที่ทำงานบนเคอร์เนลและส่วนที่ทำงานบนยูสเซอร์เสปช (user space) ค่อนข้างทำได้ลำบาก

3. การที่ต้องใช้หลายขั้นตอนกว่าจะนำข้อมูลมาวิเคราะห์หาผู้บุกรุกได้ ทำให้ระบบทำงานช้าลง และยากต่อการพัฒนา

4. ความไม่เข้าใจวิธีการของการบุกรุกดีพอ ทำให้การเขียนระบบตรวจจับผู้บุกรุกเป็นไปได้ไม่ดีเท่าที่ควร อีกทั้งการทดสอบเพื่อหาจุดอ่อนของระบบก็เป็นไปได้ลำบาก

ขอบเขตและข้อจำกัดของโครงการ

1. จากการที่ระบบจะต้องสร้างโพรเซสขึ้นมาติดตามการทำงานของโปรแกรมที่ต้องการตรวจสอบ และต้องนำข้อมูลนั้นมาทำการวิเคราะห์หาผู้บุกรุก ทำให้การทำงานโดยรวมของระบบคอมพิวเตอร์ช้าลง

2. ระบบจะสามารถตรวจจับได้เฉพาะการบุกรุกที่ทำให้เกิดการเปลี่ยนแปลงลำดับการทำงานของซิสเต็มคอลของโปรแกรมที่กำลังติดตามเท่านั้น หากมีการบุกรุกเกิดขึ้นในโปรแกรมที่ระบบไม่ได้ทำการตรวจจับการทำงาน ก็ทำให้ไม่สามารถตรวจจับการบุกรุกนั้นๆ ได้

3. ระบบจะทำงานได้ดีก็ต่อเมื่อการเก็บข้อมูลในฐานข้อมูลลำดับการทำงานของซิสเต็มคอลที่ปกติ (Normal Database) มีความสมบูรณ์ หากการเก็บข้อมูลไม่สมบูรณ์พอก็จะทำให้เกิดการเตือนที่ผิดพลาดได้สูง

4. ฐานข้อมูลของโปรแกรมเดียวกันแต่ต่างเวอร์ชั่น ก็ต้องใช้ฐานข้อมูลคนละตัว ดังนั้นการเก็บข้อมูลลำดับการทำงานของซิสเต็มคอลที่ปกติ จึงเป็นเรื่องที่ยุ่งยากสำหรับระบบนี้

5. การที่ระบบนำข้อมูลการออดิตมาจากแหล่งเดียวคือ ข้อมูลลำดับการทำงานของซิสเต็มคอลดังนั้น ระบบจึงไม่สามารถจะตรวจจับการบุกรุกได้ครบทุกรูปแบบ

แนวทางการประยุกต์และการพัฒนา

1. ปรับปรุงการเก็บข้อมูลลำดับการทำงานของซิสเต็มคอล ให้สามารถเก็บค่า address ของตำแหน่งซิสเต็มคอลที่เรียกไป เพราะถ้าเรามีค่า address เราสามารถสร้างฐานข้อมูลแบบ finite-state automaton (FSA) ความน่าเชื่อถือของฐานข้อมูลก็จะมีมากขึ้น และสามารถใช้อัลกอริทึมในรูปแบบอื่นเพื่อทำการวิเคราะห์หาผู้บุกรุกในรูปแบบที่ระบบปัจจุบันไม่สามารถทำได้

2. ปรับปรุงวิธีการในการวิเคราะห์หาผู้บุกรุก เช่นการบุกรุกบางรูปแบบอาจมีการทำงานในบางส่วนของโปรแกรมบ่อยมาก อาจเพิ่มตัววิเคราะห์การบุกรุกในรูปแบบนี้ด้วย

3. เพิ่มวิธีการในการเก็บข้อมูลการออดิต เพราะว่าเมื่อมีข้อมูลของระบบมาก และข้อมูลนั้นมีประโยชน์ต่อการวิเคราะห์ความน่าเชื่อถือของการวิเคราะห์ก็จะมากขึ้น

4. ปรับปรุงวิธีการในการส่งผ่านข้อมูลในแต่ละส่วนเพื่อให้ระบบป้องกันผู้บุกรุกสามารถทำงานได้เร็วขึ้น

5. ปรับปรุงวิธีการในการแจ้งเตือนเช่นให้ สามารถแจ้งเตือนผ่านมือถือได้

ภาคผนวก ก.
การติดตั้งและการตั้งค่าการทำงานของระบบ

การติดตั้ง

ไฟล์ที่ใช้ในการติดตั้งจะประกอบด้วยไดเรคทอรีที่สำคัญดังนี้คือ

/audit เก็บไฟล์ที่ใช้ในการตรวจจับการทำงานของโพรเซสที่เกิดจากการฟอก(fork) และไฟล์ที่ทำงานเป็นตัวติดต่อและส่งผ่านข้อมูลของระบบในแต่ละส่วน

/bash-2.05a เก็บโปรแกรม bash ซึ่งเป็นเชล์ที่ได้รับการแก้ไขให้สามารถตรวจจับการทำงานของโปรแกรมที่มีการสั่งการทำงานผ่านเชลล์ หรือโปรแกรมที่เริ่มต้นการทำงานโดยการเรียก ซิสเต็มคอลเอ็กคิว (execve)

/normal_db เก็บฐานข้อมูลลำกับการทำงานของซิสเต็มคอล์ที่ปกติของแต่ละโปรแกรม

/src เก็บไฟล์ในส่วนที่ทำการวิเคราะห์หาความผิดปกติ ส่วนที่ทำงานเป็น client-server ส่วนที่ทำงการแจ้งเตือนการบุกรุก

ลำดับในการติดตั้งมีดังนี้คือ

1. ทำการล็อกอินเป็นรูท (root) จากนั้นที่ไดเรกทอรีนอกสุดให้พิมพ์คำสั่ง

   # make

ระบบจะทำงานดังนี้คือ

1.1 สร้าง ไฟล์ที่ใช้เป็นบัฟเฟอร์ในการส่งผ่านข้อมูลสองไฟล์โดยคำสั่ง

mknod -m 666 /dev/skeleton c 32 0;

mknod -m 666 /dev/auditsuccess c 33 0;

1.2 ระบบจะทำการคอมไพล์ที่เกี่ยวข้องทั้งหมด และจะได้ไฟล์ใหม่ดังนี้คือ

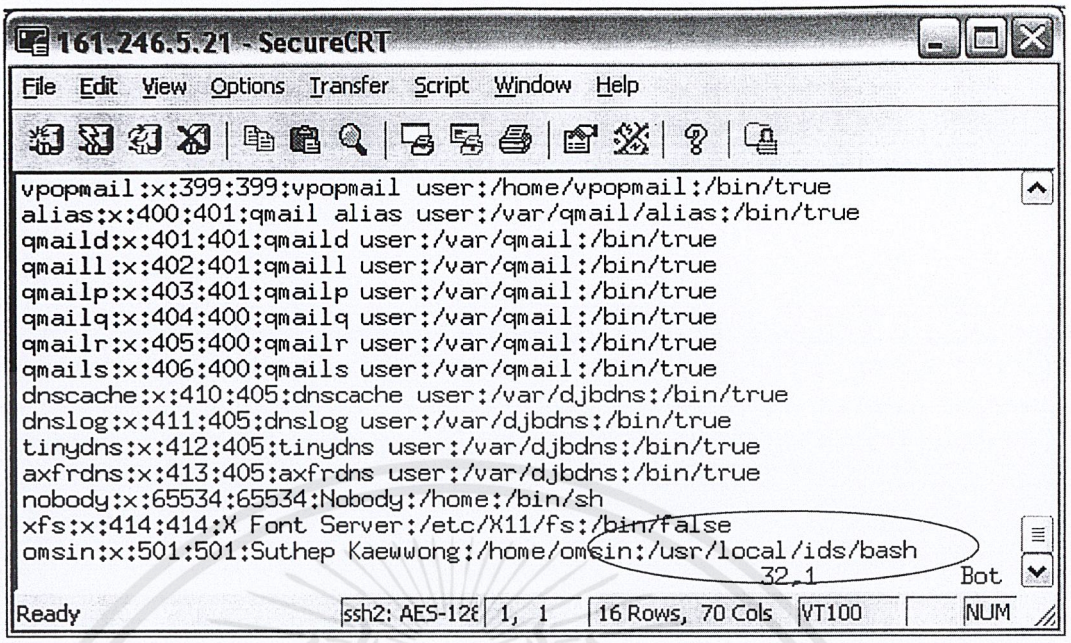
- tracesys.o เป็นโมดูลที่ทำการตรวจจับการทำงานของโพรเซสที่เกิดจากการฟอก (fork)

-transfer เป็นไฟล์ที่ทำงานเป็นตัวติดต่อและส่งผ่านข้อมูลของโปรแกรมในแต่ละส่วน

-bash เป็น bash shell ที่ได้รับการไขให้สามารถตรวจจับการทำงานของซิสเต็มคอลได้

-ids เป็นไฟล์หลัก ที่ทำงานเป็นตัววิเคราะห์และแจ้งเตือนเมื่อตรวจพบการบุกรุก

2. แก้ไขไฟล์ /etc/passwd เพื่อให้เมื่อผู้ใช้งานเข้าสู่ระบบจะทำงานผ่านเชลล์ที่เรา กำหนด โดยกำหนดให้เรียกใช้งานเชลล์ที่ได้จากการคอมไพล์ในข้างต้น



3. แก้ไขไฟล์ init ที่ใช้มีการเรียกให้งาน tracesys.o , transfer , ids ทุก ครั้งที่ระบบทำงาน

การตั้งค่าการทำงานของระบบ

การกำหนดค่าการทำงานของระบบจะกำหนดค่าในไฟล์ sc.config โดย ความหมายของค่าตัวแปรต่างๆในไฟล์ sc.config มีดังนี้

ชื่อตัวแปล	ความหมาย
user_name	ชื่อ Login ในการที่ IDS Client จะขอเชื่อมต่อกับ IDS Server เพื่อส่งข้อมูลการตรวจจับการทำงานของซิสเต็มคอล (Audit Data) เพื่อให้ IDS Server ทำการวิเคราะห์ว่ามีความผิดปกติหรือไม่
Passwd	รหัสผ่านในการที่ IDS Client จะขอทำการเชื่อมต่อกับ IDS Server เพื่อส่งข้อมูลการตรวจจับการทำงานของซิสเต็มคอล (Audit Data) เพื่อให้ IDS Server ทำการวิเคราะห์ว่ามีความผิดปกติหรือไม่
server_address	IP Address ของ IDS Server
server_port	หมายเลขพอร์ตของ IDS Server ที่จะใช้ในการส่งข้อมูลการตรวจจับการทำงานของซิสเต็มคอล (Audit Data) ระหว่าง IDS Server และ IDS Client
buffer_size	ขนาดของบัฟเฟอร์ที่ IDS Server และ IDS Client จะใช้ในการรับและส่งข้อมูล
max_diff_sc	จำนวนของรายชื่อซิสเต็มคอลล์ที่มีได้สูงสุด
add_ot_db	กำหนดว่าจะนำข้อมูลที่ได้จากการตรวจจับมาทำการวิเคราะห์หรือ เก็บลงฐานข้อมูล โดย '0' หมายถึง นำข้อมูลการออดิต มาวิเคราะห์หาผู้บุกรุก

	'1' หมายถึง เก็บข้อมูลการออดิตลง normal Database
hmd_on	กำหนดว่าจะให้คำนวณหาค่า Hamming distance ด้วยหรือไม่ '0' หมายถึง ไม่คำนวณ '1' หมายถึง คำนวณ
report_anomall_seq	กำหนดว่าในการรายงานความผิดปกติที่เกิดขึ้นให้รายงาน รายชื่อของซิสเต็มคอลที่มีความผิดปกติด้วยหรือไม่
report_all	เป็นการกำหนดว่าให้โปรแกรมรายงานค่าทุกอย่างที่สามารถรายงานได้
keep_seq_file	กำหนดว่าให้เก็บไฟล์ลำดับการทำงานของซิสเต็มคอลที่ได้จากการตรวจจับไว้หรือไม่
print_result_to_screen	กำหนดว่าให้แสดงผลลัพธ์ของการทำงานทางจอภาพด้วยหรือไม่
num_anomall_seq_alert	กำหนดจำนวนของลำดับการทำงานของซิสเต็มคอลที่ผิดปกติ ว่ามีจำนวนเท่าไรถึงจะทำการแจ้งเตือน
detect_process_name	รายชื่อของโปรแกรมที่ระบบจะคอยตรวจจับการทำงาน

การกำหนดรูปแบบในการทำงาน

โปรแกรมจะมีรูปแบบในการทำงานให้เลือกหลายรูปแบบโดยแต่ละรูปแบบจะมีการทำงานดังนี้

ids [-seq] [-db] [-c] [-s] [-sendseq] [-help]

ค่าออปชั่น	รูปแบบของการทำงาน
-seq file_name	หมายถึง ทำการเปรียบเทียบฐานข้อมูลลำดับการทำงานที่ปกติ (Normal Database) กับ ไฟล์ที่เก็บลำดับการทำงานของซิสเต็มคอล์ที่กำหนด
-db process_name	หมายถึง ให้แสดงฐานข้อมูลลำดับการทำงานที่ปกติ ของโปรแกรมที่กำหนด
-c	หมายถึง ทำงานเป็น IDS Client ทำหน้าที่ตรวจจับการทำงานของโปรแกรมที่กำหนดและส่งผลลัพธ์ของการตรวจจับไปวิเคราะห์หาความผิดปกติที่ IDS Server
-s	หมายถึง ทำงานเป็น IDS Server ทำหน้าที่วิเคราะห์หาความผิดปกติจากข้อมูลลำดับการทำงานของซิสเต็มคอลที่ได้มาจาก IDS Client
-sendseq file_name	หมายถึง ส่งไฟล์ที่เก็บลำดับการทำงานของซิสเต็มคอลตามที่กำหนดไปทำการเปรียบเทียบฐานข้อมูลลำดับการทำงานที่ปกติ (Normal Database) ที่ IDS Server
-help	หมายถึง แสดงรูปแบบการใช้งาน และความหมาย

บรรณานุกรม

1. S. Forrest, S. A. Hofmeyr, A. Somayaji, and T.A. Longstaff. Asense of self for unix processes. In Proceedings of the 1996 IEEE Symposium on Security and Privacy, pages 120-128, Los Alamitos, CA, 1996. IEEE Computer Society Press.

2. S. A. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls.

3. C. Warrender,S. Forrest,and B. Pearlmutter.Detecting Intrusions Using System Calls: Alternative Data Models

4. W. Lee and J. Stolfo. Learning Patterns from Unix Process Execution Traces for Intrusion Dectection.