บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 บทนำ

เนื้อหาในบทนี้กล่าวถึงรายละเอียดของทฤษฎีและงานวิจัยที่เกี่ยวข้องซึ่งได้แก่ ภัยคุกคามระบบคอมพิวเตอร์โดยนิยามความหมายของภัยคุกคามและการบุกรุก ช่องทางที่ผู้บุกรุกเลือกใช้ ผลกระทบที่เกิดขึ้นจากการบุกรุก และการจัดอนุกรมวิธาน (taxonomy) ของการบุกรุก ต่อมากล่าวถึงระบบตรวจจับการบุกรุกโดยเน้นถึงวิธีการตรวจจับการบุกรุกที่นำเสนอมาในอดีตและคุณลักษณะที่ดีของระบบตรวจจับการบุกรุกเพื่อเป็นแนวทางของการพัฒนาระบบ หลังจากนั้นกล่าวถึงระบบปฏิบัติการเน็ทบีเอสดีซึ่งเป็นเครื่องมือหลักของวิทยานิพนธ์โดยอธิบายถึงสถาปัตยกรรมของระบบปฏิบัติการ ซิซเท็มคอล (system call) ซึ่งเป็นหัวใจหลักของการพัฒนาระบบ ท้ายที่สุดจะกล่าวถึงวิธีการตรวจจับการบุกรุกโดยการวิเคราะห์การเปลี่ยนแปลงสถานะของโปรเซสซึ่งเป็นแนวทางของวิทยานิพนธ์ชุดนี้ จากหัวข้อต่างๆ ที่กล่าวมาข้างต้นจะอธิบายรายละเอียดของแต่ละหัวข้อตามลำดับดังนี้

2.2 ภัยคุกคามและการบุกรุกระบบคอมพิวเตอร์

ภัยคุกคาม (Threat)

ภัยคุกคาม ในที่นี้ตามคำนิยามของ [NCSC-TG-004, 1988] คือ บุคคล สิ่งของหรือเหตุการณ์ต่างๆ ที่มุ่งร้ายหรือเป็นสาเหตุของภัยอันตราย ที่จะเกิดขึ้นกับระบบคอมพิวเตอร์ในรูปแบบของการทำลาย การเปิดเผย แก้ไขข้อมูลและรวมไปถึงการทำให้ระบบไม่สามารถให้บริการแก่ผู้ใช้ได้ โดยภัยคุกคามอาจจะเกิดขึ้นจากอุบัติเหตุ เช่น ไฟไหม้ น้ำท่วม ฯลฯ หรือเกิดขึ้นจากเจตนาของบุคคลที่ประสงค์ร้ายต่อระบบ เช่น การพยายามที่จะข้ามผ่านกระบวนการรักษาความปลอดภัยของระบบคอมพิวเตอร์

การบุกรุก (Intrusion)

การบุกรุก ตามการนิยามของ Bace [Bace, 2001] คือพฤติกรรมที่พยายามกระทำการใดๆ ที่ส่งผลต่อการรักษาความลับ ความสมบูรณ์ และความพร้อมใช้ ของทรัพยากรในระบบหรือพยายามข้ามผ่านมาตรการรักษาความปลอดภัยของระบบคอมพิวเตอร์ เช่น ผู้ใช้ที่มีสิทธิ์ในระบบพยายามใช้สิทธิ์นอกเหนือจากที่ได้รับหรือพยายามใช้สิทธิ์นั้นในทางที่ผิด

จากนิยามความหมายของการบุกรุกข้างต้นได้กล่าวถึงคุณสมบัติ 3 ประการของการรักษาความปลอดภัยซึ่งได้แก่ การรักษาความลับ การรักษาความสมบูรณ์ และการรักษาความพร้อมใช้ของข้อมูล โดยคุณสมบัติแต่ละข้อมีรายละเอียดดังนี้ [Escamilla, 1998]

- **การรักษาความลับ (confidentiality)** เป็นการรับรองว่าจะมีการเก็บข้อมูลไว้เป็นความลับ และมีเพียงผู้มีสิทธิ์เท่านั้นจึงจะเข้าถึงข้อมูลนั้นได้
- **การรักษาความสมบูรณ์ (integrity)** เป็นการรับรองว่าข้อมูลไม่ถูกเปลี่ยนแปลงหรือถูกทำลายโดยผู้ที่ไม่มีสิทธิ์ ไม่ว่าจะเป็นโดยอุบัติเหตุหรือโดยเจตนา
- **การรักษาความพร้อมใช้ (availability)** เป็นการรับรองว่าข้อมูลและบริการต่างๆ มีความพร้อมที่จะเรียกใช้ได้ตามสิทธิ์ทุกครั้งที่ร้องขอ

ผู้บุกรุกจะกระทำการใดๆ เพื่อให้ระบบสูญเสียคุณสมบัติ 3 ประการข้างต้นโดยเลือกวิธีการใดวิธีการหนึ่งขึ้นอยู่กับวัตถุประสงค์ของการบุกรุก Stallings [Stallings, 1995] ได้แบ่งการบุกรุกตามลักษณะของการกระทำได้ 4 ประเภทได้แก่ interruption, interception, modification และ fabrication โดยแต่ละประเภทมีรายละเอียดดังนี้

- **interruption** คือ การทำให้ทรัพยากรของระบบถูกทำลาย ทำให้ระบบไม่สามารถให้บริการหรือไม่สามารถใช้งานได้อีกต่อไป เช่นการทำลายอุปกรณ์ การทำให้เครือข่ายท่วม (network flooding) หรือการลบแฟ้มข้อมูล เป็นต้น
- **interception** คือ การที่บุคคลหรือโปรแกรมที่ไม่ได้รับอนุญาตสามารถเข้าถึงทรัพยากรหรือข้อมูลได้ เช่น การดักฟังสัญญาณการสื่อสารในเครือข่าย การคัดลอกไฟล์โดยไม่ได้รับอนุญาต เป็นต้น
- **modification** คือ การที่บุคคลหรือโปรแกรมที่ไม่ได้รับอนุญาตสามารถเข้าถึงทรัพยากรและแก้ไขข้อมูล เช่น การแก้ไขข้อมูลในไฟล์ การปรับเปลี่ยนโปรแกรมให้มีการทำงานที่ต่างไปจากการทำงานปกติหรือการแก้ไขข้อมูลในเครือข่าย เป็นต้น
- **fabrication** คือการที่บุคคลหรือโปรแกรมที่ไม่ได้รับอนุญาตปลอมแปลงข้อมูลขึ้นมาในระบบ เช่น การปลอมข้อมูลข่าวสารที่รับส่งในเครือข่าย เป็นต้น

2.2.1 ช่องทางของการบุกรุก

Stallings [Stallings, 1995] ได้จัดกลุ่มของวิธีการบุกรุกระบบคอมพิวเตอร์ออกเป็น 2 แบบ คือ การบุกรุกแบบ active และการบุกรุกแบบ passive สำหรับการบุกรุกแต่ละแบบมีรายละเอียดดังนี้

การบุกรุกแบบ active

การบุกรุกแบบ active เป็นการบุกรุกที่ก่อให้เกิดการแก้ไขข้อมูลหรือสร้างข้อมูลขึ้นมาใหม่โดยการปลอมแปลง เช่น การเปลี่ยนแปลงข้อมูลในไฟล์หรือการเพิ่มไฟล์ที่ไม่ได้รับอนุญาตเข้าไปในระบบ รวมถึงการทำให้ระบบไม่สามารถให้บริการผู้ใช้ได้ การโจมตีแบบนี้ผู้ดูแลระบบสามารถตรวจจับได้ง่ายเนื่องจากมีร่องรอยจากการกระทำ เช่น การบุกรุกโดยไวรัส (virus) โปรแกรมโทรจัน (trojan) แบคทีเรีย (bacterium) หนอนอินเทอร์เน็ต (Internet worm) ประตูลับ (backdoor) Denial of Service (DoS) เป็นต้น

การบุกรุกแบบ passive

การบุกรุกแบบ passive เป็นการบุกรุกที่ไม่ทำให้เกิดการเปลี่ยนแปลงของข้อมูล แต่ผู้บุกรุกสามารถเข้าถึงข้อมูลได้โดยไม่โดยไม่ได้รับอนุญาต เช่น การแอบดักจับข้อมูลในสายสัญญาณหรือเครือข่าย การบุกรุกประเภทนี้ตรวจจับได้ยาก แต่สามารถป้องกันได้ง่าย เช่นเข้ารหัสข้อมูลก่อนรับส่ง เป็นต้น ตัวอย่างของการบุกรุกในกลุ่มนี้คือ eavesdropping, sniffer, wiretapping, social engineering และ port scanning เป็นต้น

รายละเอียดของการบุกรุกแต่ละชนิดนั้นสามารถศึกษาเพิ่มเติมจากรายงานของศุภโชคและคณะ [ศุภโชค, 2548] ซึ่งศึกษาถึงวิธีการบุกรุกที่เกิดขึ้นในอดีต แต่วิธีการบุกรุกตามรายงานข้างต้นนั้นเป็นเพียงส่วนหนึ่งเท่านั้น ยังมีช่องทางให้ผู้บุกรุกเข้ามาก่อกวนระบบได้อีกหลายวิธี ถ้าจัดกลุ่มวิธีการบุกรุกตามสาเหตุหรือวิธีการเกิด (genesis) จะพบว่าวิธีการบุกรุกที่กล่าวมาข้างต้นนั้นเกิดจากความตั้งใจของผู้บุกรุกที่จะหาเครื่องมือหรือวิธีการมาช่วยสนับสนุนการบุกรุกให้ประสบความสำเร็จเช่น การพัฒนาโปรแกรมไวรัส หนอนอินเทอร์เน็ต หรือโปรแกรมโทรจัน เป็นต้น วิธีการบุกรุกอีกประเภทหนึ่งคือผู้บุกรุกโจมตีระบบผ่านช่องทางซึ่งเกิดจากความผิดพลาดของขั้นตอนของการออกแบบ การพัฒนา การติดตั้ง และขั้นตอนของการบำรุงรักษาระบบ เป็นต้น ช่องทางที่ก่อให้เกิดการบุกรุกระบบดังที่กล่าวมานี้ เรียกว่า “จุดอ่อน (vulnerability)” ตัวอย่างจุดอ่อนที่ก่อให้เกิดการบุกรุกระบบคือ race condition, buffer overflow และ rootkits เป็นต้น รายละเอียดของจุดอ่อนเหล่านี้จะกล่าวถึงในลำดับต่อไป

2.2.1.1 Race condition

race condition หมายถึงเหตุการณ์ที่เกิดจากโปรเซส (หรือโปรแกรมที่กำลังทำงานในหน่วยความจำ) มากกว่าหนึ่งโปรเซสพยายามเข้าใช้ทรัพยากรเดียวกันในช่วงเวลาเดียวกัน ในขณะที่มีโปรเซสกำลังดำเนินงานกับทรัพยากรชุดใดชุดหนึ่ง อีกโปรเซสเข้ามาขัดจังหวะและดำเนินการกับทรัพยากรชุดนั้น ผลของการดำเนินงานข้างต้นไม่สามารถทำนายล่วงหน้าได้ว่าจะเกิดผลอ

ย่างไรขึ้นอยู่กับว่าโปรเซสไหนได้เข้าใช้ทรัพยากรนั้นๆ ซึ่งทรัพยากรที่มีโอกาสประสบปัญหานี้ได้แก่ หน่วยความจำและระบบไฟล์เป็นต้น

จุดอ่อนประเภทนี้มักจะเกิดขึ้นในระบบไฟล์ซึ่งเรียกว่าปัญหาของลำดับการทำงาน (sequence problem) โดยเกิดจากการประมวลผลชุดคำสั่งของระบบปฏิบัติการบางชุดคำสั่งไม่เป็นแบบอะตอม (non-atomic instruction) ในขณะที่โปรเซสกำลังประมวลข้อมูล โปรเซสดังกล่าวอาจจะถูกขัดจังหวะโดยระบบปฏิบัติการ ในช่วงเวลานั้นหากโปรเซสใหม่ต้องการเข้าใช้ทรัพยากรเดียวกันกับโปรเซสแรกก็สามารถทำได้ ผลกระทบที่อาจจะเกิดขึ้น ได้แก่ปัญหาของการดำเนินงานเกี่ยวกับไฟล์ ปัญหาของโปรแกรมสคริปต์ และปัญหาของ Time-of-Check-Time-of-Use (TOCTOU) ปัญหาในแต่ละข้อมีรายละเอียดดังนี้

ปัญหาของการดำเนินงานเกี่ยวกับไฟล์

ส่วนหนึ่งของโปรแกรมที่แสดงข้างล่างนั้นเป็นโปรแกรมแบบ setuid มีการตรวจสอบสิทธิ์การอ่านหรือเขียนแฟ้มข้อมูล (ในที่นี้คือแฟ้ม /tmp/x) หากโปรแกรมมีสิทธิ์ในการเข้าถึงแฟ้มดังกล่าวแล้วจะสามารถเปิดแฟ้มเพื่ออ่านหรือเขียนข้อมูลในแฟ้มได้ จากตัวอย่างนี้พบว่าถ้าหากกระบวนการทำงานข้างต้นไม่เป็นแบบอะตอมแล้ว เมื่อการประมวลฟังก์ชัน access() เสร็จโปรแกรมถูกขัดจังหวะโดยระบบปฏิบัติการได้ ผู้บุกรุกสามารถดำเนินการบางอย่างเพื่อเปลี่ยนชื่อแฟ้มในพารามิเตอร์ของฟังก์ชัน open() เป็นแฟ้มอื่น เช่น แฟ้มฐานข้อมูลผู้ใช้ (หรือ /etc/passwd) ผู้บุกรุกสามารถเข้าถึงฐานข้อมูลผู้ใช้ได้เมื่อโปรแกรมกลับมาทำงานอีกครั้ง

```
int fd;
if (access("/tmp/x", R_OK|W_OK))
{
        fd = open("/tmp/x",O_RDWR,0644)
}
```

ปัญหาของโปรแกรมสคริปต์

ปัญหาของโปรแกรมสคริปต์แบบ setuid ในระบบปฏิบัติการยูนิกซ์เป็นที่รู้จักมานาน การประมวลคำสั่งแบบอินเทอร์พรีเตอร์ (interpreter) ระบบปฏิบัติการจะอ่านไฟล์บรรทัดแรกเพื่อประมวลผลคำสั่ง #! /bin/sh และสั่งงานโปรเซสใหม่ แต่การทำงานของสริปต์นั้นไม่ได้เป็นแบบอะตอม ดังนั้นเมื่อระบบประมวลคำสั่ง #! เกิดการขัดจังหวะจากระบบปฏิบัติการ และผู้บุกรุก

เปลี่ยนชื่อโปรแกรมจาก /bin/sh เป็นโปรแกรมอื่น เมื่อระบบปฏิบัติการประมวลคำสั่งต่อเป็นผลให้โปรแกรมของผู้บุกรุกถูกเรียกขึ้นมาทำงาน

ปัญหาของ Time-of-Check-Time-of-Use

ปัญหาของ Time-of-Check-Time-of-Use หรือ TOCTOU เป็นจุดอ่อนของระบบที่เกิดจากการตรวจสอบสิทธิ์ของโปรแกรมที่จะเข้าถึงแฟ้มหรือทรัพยากรก่อน หลังจากนั้นโปรแกรมดังกล่าวจะดำเนินงานกับทรัพยากรที่ร้องขอ แต่ขั้นตอนการทำงานข้างต้นไม่เป็นแบบอะตอมจึงก่อให้เกิดปัญหา race condition ตัวอย่างจุดอ่อนที่เกิดจากปัญหาของ TOCTOU บนระบบปฏิบัติการ SunOS และระบบปฏิบัติการ HP/UX เมื่อผู้ใช้สั่งงานโปรแกรม passwd ซึ่งเป็นโปรแกรมแบบ setuid เพื่อกำหนดรหัสผ่านใหม่ การทำงานของโปรแกรม passwd มีกระบวนการทำงานดังนี้ [Bishop, 1996]

1. โปรแกรมอ่านข้อมูลของผู้ใช้จากแฟ้ม /etc/passwd
2. โปรแกรมสร้างแฟ้มข้อมูลชั่วคราวชื่อ /etc/ptmp
3. โปรแกรมเปิดแฟ้ม /etc/passwd อีกครั้ง เพื่อคัดลอกข้อมูลไปยัง /etc/ptmp และแก้ไขข้อมูลของผู้ใช้ซึ่งเก็บอยู่ในแฟ้ม /etc/ptmp
4. หลังจากนั้นโปรแกรม passwd เปลี่ยนชื่อแฟ้ม /etc/ptmp เป็นแฟ้ม /etc/passwd

แต่ขั้นตอนการทำงานข้างต้นไม่เป็นแบบอะตอมจึงเกิดช่องทางให้ผู้บุกรุกโจมตีระบบได้ โดยผู้บุกรุกอาศัยจุดอ่อนดังกล่าวแอบคัดลอกข้อมูลจาก /etc/passwd ไปยัง home directory ของ root เพื่อสร้างแฟ้มชื่อ .rhost ถ้าหากการดำเนินการดังกล่าวประสบความสำเร็จ ผู้บุกรุกสามารถเข้าสู่ระบบและได้รับสิทธิ์เป็น root โดยไม่ผ่านการตรวจสอบตัวตน [Bishop, 1996]

2.2.1.2 Buffer overflow

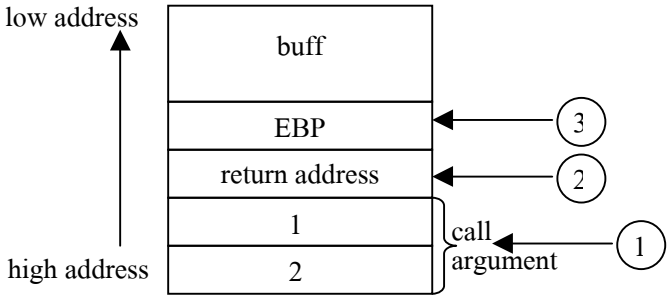
บัฟเฟอร์ (buffer) ในที่นี้คือพื้นที่ของหน่วยความจำสำหรับเก็บข้อมูลในขณะที่โปรเซสกำลังทำงาน พื้นที่ดังกล่าวถูกจองไว้สำหรับข้อมูลที่มีขนาดจำกัด ถ้าหากโปรเซสพยายามที่จะเก็บข้อมูลลงในบัฟเฟอร์โดยที่ขนาดของข้อมูลมีขนาดใหญ่กว่าบัฟเฟอร์แล้วข้อมูลบางส่วนจะล้นออกจากหน่วยความจำที่จองไว้ ข้อมูลส่วนเกินจะล้นเข้าสู่หน่วยความจำตำแหน่งข้างเคียง ซึ่งจะเป็นผลให้โปรเซสทำงานผิดพลาดได้ สำหรับผลกระทบที่เกิดขึ้นนั้นขึ้นอยู่กับข้อมูลส่วนที่ล้น เช่น ตำแหน่งของชุดคำสั่งประสงค์ร้าย เป็นต้น รายละเอียดในลำดับถัดไปเป็นการกล่าวถึงการเรียกใช้ฟังก์ชันของโปรเซส และการบุกรุกด้วย buffer overflow

การเรียกใช้ฟังก์ชันของโปรเซส

โปรเซสใช้หน่วยความจำแสตก (stack) สำหรับดำเนินงานเกี่ยวกับการเรียกใช้ฟังก์ชันเมื่อฟังก์ชันถูกเรียกใช้ โปรเซสจะจองหน่วยความจำแสตกเพื่อเก็บค่าพารามิเตอร์ (parameter) ซึ่งถูกส่งผ่านไปยังฟังก์ชันและใช้หน่วยความจำส่วนนี้เก็บตำแหน่งของชุดคำสั่งที่จะประมวลผลต่อไป (instruction pointer register หรือ IP)

ตัวอย่างต่อไปนี้เป็นโปรแกรมภาษาซีที่มีการเรียกใช้ฟังก์ชันซึ่งมีการส่งผ่านค่าพารามิเตอร์ เมื่อฟังก์ชัน main() เรียกใช้ฟังก์ชัน func() ระบบปฏิบัติการจะจัดเก็บค่าพารามิเตอร์ (1) return address (2) และ รีจิสเตอร์ที่เกี่ยวข้อง (3) ไว้ในแสตก ดังภาพประกอบที่ 2.1

```
void f(int a, int b){
        char buff[10];
}
void main(){
        f(1,2);
}
```



ภาพประกอบ 2.1 โครงสร้างของหน่วยความจำแสตก

จากภาพประกอบที่ 2.1 แสดงโครงสร้างของหน่วยความจำแสตกที่จัดเก็บข้อมูลต่างๆ ที่กล่าวมาแล้วข้างต้น เมื่อโปรเซสเรียกใช้ฟังก์ชัน f(1, 2) ระบบปฏิบัติการจะพุช (push) ค่าต่างๆ ต่อไปนี้ลงสู่แสตกตามลำดับ

1. ค่าพารามิเตอร์ซึ่งได้แก่ค่า 2 และ 1
2. return address หมายถึงตำแหน่งของโปรแกรมที่จะประมวลผลต่อหลังจากที่ทำงานในฟังก์ชันเรียบร้อยแล้ว
3. รีจิสเตอร์ EBP หรือ based pointer register ใช้สำหรับชี้ตำแหน่งฐานของแสตกซึ่งใช้คู่กับ ESP หรือ stack pointer register
4. ตัวแปรโลคอลที่ประกาศใช้ในฟังก์ชัน ในที่นี้คือ buff

เมื่อโปรเซสได้ดำเนินการต่างๆ ในฟังก์ชัน func() เรียบร้อยแล้ว ระบบปฏิบัติการจะคืนค่า (pop) ในหน่วยความจำแสตก หลังจากนั้นหน่วยประมวลผลกลางจะกลับไปประมวลผลคำสั่ง ณ ตำแหน่งที่ระบุไว้ใน return address

การบุกรุกระบบด้วย buffer overflow

จากกระบวนการเรียกใช้ฟังก์ชันพบว่า เมื่อการทำงานในฟังก์ชันสิ้นสุดลงหน่วยประมวลผลกลางจะประมวลผลคำสั่งถัดไปณ ตำแหน่งที่ระบุไว้ใน return address ผู้บุกรุกอาศัยช่องทางข้างต้นโดยแก้ไขตำแหน่งของหน่วยความจำซึ่งเป็นอยู่ใน return address เพื่อเปลี่ยนข้อมูลตำแหน่งที่ถูกต้องเป็นตำแหน่งของชุดคำสั่งประสงค์ร้าย ดังตัวอย่างส่วนหนึ่งของโปรแกรมต่อไปนี้

```
char code[] = "AAAABBBBCCCCDDDD";
void func(){
        char buff[8];
        strcpy(buff, code);
}
```

ส่วนหนึ่งของโปรแกรมดังกล่าวได้จองหน่วยความจำชื่อ buff ขนาด 8 ไบต์ หลังจากนั้นคัดลอกค่าจากตัวแปร code ซึ่งยาว 16 ไบต์ไปเก็บไว้ที่ตัวแปร buff จากการทำงานข้างต้นจะเกิด buffer overflow บนตัวแปร buff ซึ่งอธิบายได้จากโครงสร้างของแสตกในภาพประกอบที่ 2.2

buff								EBP				return address			
A	A	A	A	B	B	B	B	C	C	C	C	D	D	D	D

low address high address

ภาพประกอบ 2.2 โครงสร้างของหน่วยความจำแสตกเมื่อเกิด buffer overflow

จากภาพประกอบที่ 2.2 แสดงให้เห็นว่า ตัวแปร buff ไม่สามารถเก็บค่าที่ได้จากตัวแปร code ทั้งหมด มีข้อมูลบางส่วนล้นเข้าไปในตำแหน่งของ EBP และ return address ดังนั้นเมื่อ

การทำงานของฟังก์ชันสิ้นสุดลง ตำแหน่งถัดไปที่จะถูกประมวลผลคือตำแหน่งที่เก็บไว้ใน return address จากตัวอย่างนี้คือ DDDD ในกรณีของโปรแกรมบุกรุก ค่าของ DDDD คือตำแหน่งของหน่วยความจำที่จัดเก็บชุดคำสั่งประสงค์ร้ายที่พร้อมทำงาน

2.2.1.3 Rootkit

Siliman [Siliman, 2004] ได้ให้นิยามของคำว่า rootkit ไว้ว่า rootkits คือเครื่องมือที่ผู้บุกรุกนิยมใช้สำหรับบุกรุกระบบเนื่องจากเครื่องมือดังกล่าวมีประสิทธิภาพเทียบเคียงได้กับเครื่องมือของผู้ดูแลระบบ ผู้บุกรุกจะติดตั้งโปรแกรม rootkit หลังจากที่สามารถเข้าสู่ระบบเป้าหมายได้ การติดตั้งโปรแกรมเหล่านี้ผู้บุกรุกต้องการสิทธิ์ของผู้ดูแลระบบสำหรับการสร้างโปรแกรมประสงค์ร้ายโดยอาศัยจากสิทธิ์ของผู้ดูแลระบบจากการโจมตีระบบผ่านจุดอ่อนของโปรแกรมที่มีสิทธิพิเศษซึ่งจะถ่ายทอดไปยังโปรแกรม rootkit ที่กำลังติดตั้ง ตัวอย่างโปรแกรมในกลุ่มของ rootkit เช่น โปรแกรม backdoor และโปรแกรมดักจับแพ็คเก็ต เป็นต้น

โปรแกรม backdoor

backdoor คือช่องทางที่ผู้บุกรุกใช้เข้าสู่ระบบโดยไม่ผ่านกระบวนการตรวจสอบตัวตนจากระบบปฏิบัติการ โปรแกรมดังกล่าวจะถูกติดตั้งลงในระบบปฏิบัติการเป้าหมายหลังจากที่บุกรุกระบบได้สำเร็จ หลังจากนั้นผู้บุกรุกจะเข้าสู่ระบบผ่านช่องทางนี้โดยไม่จำเป็นต้องอาศัยจุดอ่อนของระบบอีก นอกจากนี้อาจจะสั่งงานโปรแกรมชุดนั้นเพื่อเก็บข้อมูลหรือกระทำการใดๆ ตามเป้าหมายของผู้บุกรุก เช่น

- **login backdoor** เป็นโปรแกรมที่ได้จากการแก้ไขโปรแกรม login ของระบบปฏิบัติการยูนิกซ์เพื่อที่จะเก็บและบันทึกรหัสผ่านของผู้ใช้คนอื่นเมื่อผู้ใช้คนนั้นติดต่อเข้ามาในระบบ
- **telnetd backdoor** เป็นโปรแกรมที่ได้จากการแก้ไขโปรแกรม in.telnetd ซึ่งจะอนุญาตให้ผู้บุกรุกสามารถเข้าสู่ระบบได้โดยใช้รหัสผ่านอื่นที่ไม่มีในฐานข้อมูลผู้ใช้
- **services backdoor** เป็นโปรแกรมที่ได้การแก้ไขโปรแกรมในกลุ่มของโปรแกรมเซิร์ฟเวอร์ของระบบ เช่น ftp, http หรือ mail เป็นต้น
- **library backdoor** เป็นการแก้ไขชุดไลบรารีของระบบเนื่องจากระบบปฏิบัติการยูนิกซ์มีการใช้ไลบรารีร่วมกัน (shared library) ถ้าหากชุดไลบรารีเหล่านี้ถูกแก้ไขแล้วโปรแกรมทุกโปรแกรมที่เรียกใช้ไลบรารีนี้จะได้รับผลกระทบนี้ด้วย

โปรแกรมตรวจจับแพ็คเก็ต (packet sniffers)

packet sniffer เป็นโปรแกรมที่คอยตรวจตราข้อมูลที่ส่งผ่านเครือข่าย ผู้บุกรุกใช้โปรแกรมกลุ่มนี้สำหรับคอยฟัง (listen) และขโมยข้อมูลจากเครือข่ายหรือบริการต่างๆ เช่น โปรแกรม ftp และ โปรแกรม telnet รับส่งรหัสผ่านในรูปแบบของข้อมูลที่สามารถอ่านได้ จึงง่ายแก่การตรวจจับโดยโปรแกรม sniffers

จากรายละเอียดของวิธีการหรือช่องทางของการบุกรุกระบบดังที่กล่าวมาข้างต้น พบว่า การบุกรุกระบบแต่ละประเภทมีวิธีการทำงาน ความยากง่ายของการตรวจจับและการป้องกันที่แตกต่างกัน บางวิธีการสามารถป้องกันได้ตั้งแต่กระบวนการออกแบบและพัฒนาระบบ แต่บางวิธีการนั้นจำเป็นต้องคอยเฝ้าระวังและหาทางรับมือกับเหตุการณ์ที่จะเกิดขึ้นโดยที่ผู้ดูและระบบไม่สามารถทำนายเหตุการณ์ล่วงหน้าได้ ดังนั้นผู้ดูแลระบบเองจำเป็นต้องมีเครื่องมือต่างๆ เพื่อเป็นตัวช่วยในการตรวจจับและป้องกันการบุกรุกที่จะเกิดขึ้น

2.2.2 ผลกระทบของการบุกรุกระบบ

พัฒนาวดีและคณะ [พัฒนาวดี, 2548] ได้ศึกษาถึงผลกระทบของการบุกรุกระบบที่จะเกิดขึ้นถ้าหากการบุกรุกนั้นประสบผลสำเร็จ โดยศึกษาจากรายงานการบุกรุกระบบที่เกิดขึ้นในอดีตและสรุปไว้ว่า ผลกระทบของการบุกรุกระบบแบ่งออกเป็น 3 กลุ่มได้แก่

- **ผู้บุกรุกได้รับชื่อบัญชีผู้ใช้ในระบบ** ถ้าหากสิทธิ์ของชื่อบัญชีดังกล่าวที่มีสิทธิ์สูงสุดในระบบแล้ว ผู้บุกรุกได้รับสิทธิ์ในการควบคุมทรัพยากรทั้งหมดหรือในกรณีที่ผู้บุกรุกได้รับสิทธิ์เทียบเท่ากับเจ้าของโปรแกรมที่มีจุดอ่อน ผู้บุกรุกสามารถใช้ทรัพยากรในระบบเท่ากับเจ้าของโปรแกรมเหล่านั้นซึ่ง ในกรณีนี้ถ้าหากโปรแกรมที่มีจุดอ่อนนั้นเป็นโปรแกรมแบบ setuid แล้วผู้บุกรุกอาจจะได้รับสิทธิ์ที่เทียบเท่ากับผู้ดูแลระบบ
- **ผู้บุกรุกสามารถเข้าถึงแฟ้มของระบบ** ผลกระทบที่เกิดขึ้นได้แก่ ผู้บุกรุกจะเผยแพร่หรือเปิดเผยข้อมูลสำคัญ แก้ไข สร้างหรือทำลายแฟ้มข้อมูลที่มีอยู่ในระบบ ในบางกรณีผู้บุกรุกอาจจะมีสิทธิ์ที่จะแก้ไขฐานข้อมูลผู้ใช้หรือกำหนดสิทธิ์ให้ผู้ที่ไม่มีสิทธิ์สามารถเข้าใช้ระบบได้
- **ส่วนของผลกระทบอื่นๆ** เช่นผู้บุกรุกสามารถใช้จุดอ่อนของโปรแกรมในการติดตั้งโปรแกรมโทรจัน โปรแกรม rootkits หรือแม้แต่ทำให้ระบบไม่สามารถให้บริการระบบได้ตามปกติ

ภัยคุกคามที่เกิดขึ้นในระบบส่วนใหญ่มาจากผู้บุกรุกซึ่งอาจจะอยู่ในองค์กรหรือมาจากนอกองค์กร ผู้บุกรุกดังกล่าวอาศัยเครื่องมือต่างๆ ดังที่กล่าวมาแล้ว เพื่อบุกรุกระบบผ่านจุดอ่อนของระบบ ระดับความรุนแรงของผลกระทบที่เกิดขึ้นนั้นขึ้นอยู่กับสิทธิ์ที่ผู้บุกรุกได้รับ

2.3 การจัดอนุกรมวิธานของการบุกรุก

อนุกรมวิธาน (taxonomy) [Bishop, 1995] ความหมายโดยทั่วไปคือระบบของการจัดหมวดหมู่สิ่งใดๆ สำหรับการรู้จำ (recognize) หรือตั้งชื่อสิ่งนั้น การทำอนุกรมวิธานนี้เป็นที่รู้จักในสาขาชีววิทยาซึ่งเป็นการจำแนกพืชและสัตว์เป็นกลุ่มๆ โดยสมาชิกในกลุ่มมีความสัมพันธ์กัน เช่นสิ่งมีชีวิตชนิดเดียวกันย่อมถูกจัดให้อยู่กลุ่มเดียวกัน

การจัดอนุกรมวิธานของการบุกรุกคอมพิวเตอร์มีแนวคิดเดียวกับการจัดทำอนุกรมวิธานทางชีววิทยา โดยมีเป้าหมายในการจัดจำแนกจุดอ่อนและรูปแบบการบุกรุกที่ตรวจพบให้อยู่ในกลุ่มความสัมพันธ์และรูปแบบที่สามารถนำไปใช้ประโยชน์ได้ เช่น ออกแบบและพัฒนาระบบการป้องกันการโจมตีระบบ และช่วยให้ระบบเฝ้าระวังสามารถตรวจจับการโจมตีผ่านจุดอ่อนเป็นต้น ในช่วงหลายปีที่ผ่านมากลุ่มวิจัยหลายกลุ่มได้นำเสนออนุกรมวิธานที่มีเป้าหมายและจุดเด่นที่แตกต่างกัน เช่น

การจัดกลุ่มอนุกรมวิธานในระยะแรกๆ ทางสาขานี้มีอนุกรมวิธานสองกลุ่มได้แก่ อนุกรมวิธานของ Protection Analysis (PA) [Bisbey, 1978] และของ Research in Secured System (RISOS) [Abbott, 1976] งานทั้งสองกลุ่มนี้เน้นการจัดกลุ่มของจุดอ่อนมากกว่าการบุกรุก และสร้างรากฐานสำหรับการจัดอนุกรมวิธานใหม่ๆ รูปแบบของการจัดกลุ่มของงานวิจัยทั้งสองกลุ่มนี้มีความคล้ายคลึงกัน

ต่อมา Bishop [Bishop, 1995] ได้ชี้ให้เห็นว่าการจัดแบ่งกลุ่มตามแนวคิดของ PA และ RISOS นั้นมีความกำกวมจึงเสนอรูปแบบการจัดอนุกรมวิธานใหม่ โดยจัดอนุกรมวิธานของจุดอ่อนของระบบโดยนำเสนอเงื่อนไขของการจัดกลุ่มออกเป็น 6 เงื่อนไขได้แก่

- **nature** แสดงถึงธรรมชาติการจุดอ่อนโดยอ้างอิงงานวิจัยของ PA
- **time of introduction** เป็นช่วงเวลาของการเกิดจุดอ่อน เช่น จุดอ่อนเกิดขึ้นในช่วงของการออกแบบระบบ การพัฒนาระบบ หรือการติดตั้งและปรับแต่งระบบ
- **exploitation domain** เป็นการนิยามถึงผลกระทบที่ได้รับเมื่อการบุกรุกผ่านจุดอ่อนนั้นประสบความสำเร็จ เช่น ผู้บุกรุกได้รับสิทธิ์ของ root

- **effect domain** เป็นกลุ่มของระบบเป้าหมายที่เกิดจุดอ่อน เช่นอุปกรณ์คอมพิวเตอร์ ระบบปฏิบัติการ การทำงานของเครือข่ายทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ เป็นต้น
- **minimum number** เป็นการระบุถึงองค์ประกอบขั้นต่ำสุดที่จำเป็นในการโจมตีระบบเป้าหมาย เช่นการบุกรุกระบบผ่านจุดอ่อนของ sendmail นั้นจำเป็นต้องมีโปรเซสของ sendmail ทำงานอยู่ในขณะที่เกิดการบุกรุก
- **source** เป็นการระบุแหล่งที่มาของการนิยามจุดอ่อนดังกล่าว เช่น Common Vulnerabilities and Exposures (CVE) เป็นต้น

การจัดอนุกรมวิธานตามแนวคิดของ Bishop เป็นเพียงการจำแนกการบุกรุกตามจุดอ่อนของระบบเพื่อใช้เป็นฐานข้อมูลอ้างอิงสำหรับหรับงานวิจัยอื่นๆ ในเวลาต่อมามีกลุ่มวิจัยบางกลุ่มอาศัยแนวคิดนี้เป็นพื้นฐานสำหรับการหาวิธีการใหม่ๆ ของการจัดอนุกรมวิธานของการบุกรุกระบบ

Howard [Howard, 1997] ได้นำเสนออนุกรมวิธานของการบุกรุกระบบคอมพิวเตอร์ โดยอิงตามการวิธีการบุกรุกเป็นหลักซึ่งแตกต่างกับแนวคิด Bishop โดยนิยามองค์ประกอบของการจำแนกการบุกรุกออกเป็น 5 ส่วนได้แก่

- **attack** หมายถึงระดับของบุคคลที่พยายามบุกรุกระบบโดยประกอบด้วยระดับของผู้บุกรุกโดยกล่าวถึงผู้ท้าทายหรือลองวิชาจนถึงระดับการโจมตีร้ายแรง
- **tools** หมายถึงเครื่องมือที่ผู้บุกรุกเลือกใช้เพื่อให้บรรลุวัตถุประสงค์ เช่นการพัฒนาโปรแกรมบุกรุก การขโมยข้อมูลในเครือข่าย เป็นต้น
- **access** หมายถึงช่องทางหรือจุดอ่อนที่ผู้ใช้เลือกใช้ในการเข้าสู่ระบบ จุดอ่อนดังกล่าวอาจจะเกิดในกระบวนการออกแบบ พัฒนาหรือติดตั้งระบบ
- **result** หมายถึงผลลัพธ์ที่จะเกิดขึ้นเมื่อการบุกรุกประสบความสำเร็จ เช่นการแก้ไขหรือเปิดเผยข้อมูลสำคัญ เข้าใช้ทรัพยากรต่างๆ โดยไม่มีสิทธิ์ หรือกระทำการใดๆ เพื่อให้ระบบไม่สามารถให้บริการได้ตามปกติ
- **objective** หมายถึงเป้าหมายของการบุกรุกระบบเช่นต้องการสร้างความเสียหายหรือการได้รับสิทธิใดต่างๆ ในระบบ หรือเป้าหมายทางด้านการเงิน ชื่อเสียง

ต่อมา Lough [Lough, 2001] ได้ชี้ให้เห็นว่าแนวคิดของ Howard ไม่สามารถจัดกลุ่มของการบุกรุกระบบได้ทั้งหมด อีกทั้งแนวคิดข้างต้นไม่สามารถแบ่งกลุ่มได้อย่างชัดเจนดังนั้น

Lough จึงนำเสนอการจัดอนุกรมวิธานขึ้นมาใหม่โดยมีชื่อว่า Validation Exposure Randomness Deallocation Improper Conditions Taxonomy หรือ VERDICT

Lough เสนอแนวคิดว่า จุดอ่อนของระบบนั้นเกิดมาจากความผิดพลาดของการผลิตซอฟต์แวร์ตั้งแต่การออกแบบ พัฒนา และติดตั้งบำรุงรักษาระบบ Lough จึงจัดจำแนกจุดอ่อนของระบบตามสาเหตุของจุดอ่อนออกเป็น 4 ประเภทได้แก่ improper validation, improper exposure, improper randomness และ improper deallocation

- **improper validation** หมายถึงจุดอ่อนที่เกิดจากการตรวจสอบเงื่อนไขที่เป็นค่าวิกฤติหรือค่าพารามิเตอร์ของระบบไม่เพียงพอเป็นผลให้โปรเซสสามารถเข้าถึงหน่วยความจำบางส่วนของระบบ ซึ่งก่อให้เกิดความเสียหายแก่ระบบ หรือเกิดปัญหา buffer overflow เป็นต้น
- **improper exposure** หมายถึงจุดอ่อนที่เกิดจากระบบปฏิบัติการเปิดเผยการสร้างชุดคำสั่งแบบนามธรรม (abstract) ได้แก่ชุดคำสั่งเทียม (pseudo-instruction) ซึ่งอนุญาตให้ผู้ใช้สามารถเข้าใช้ทรัพยากรต่างโดยที่ผู้ใช้ไม่ต้องเข้าใจชุดคำสั่งภาษาเครื่องด้วยความเลินเล่อในบางกรณีผู้ใช้สามารถเข้าถึงข้อมูลบางอย่างที่ควรปกปิด เช่น บริการ finger เป็นโปรแกรมที่ให้บริการข้อมูลสำคัญซึ่งผู้บุกรุกอาจจะใช้ประโยชน์สำหรับการบุกรุกทั้งทางตรงและทางอ้อม
- **improper randomness** หมายถึงจุดอ่อนที่เกิดจากระเบียบวิธีการสุ่มของระบบปฏิบัติการไม่ดีพอ เนื่องจากการสุ่มตัวเลขของระบบปฏิบัติการไม่ใช่การสุ่มที่แท้จริง ระบบปฏิบัติการสุ่มตัวเลขโดยอาศัยการคำนวณตัวเลขโดยกำหนดค่าเริ่มต้น (seed) แล้วคำนวณค่านั้นด้วยสูตรพหุนามเพื่อที่จะสร้างตัวเลขชุดใหม่ และทำซ้ำจนได้ข้อมูลสุ่มออกมา จากวิธีการข้างต้นจะเกิดผลกระทบต่อการเข้ารหัสข้อมูล ผู้บุกรุกสามารถที่จะถอดรหัสข้อมูลได้ เช่น การเลือกกุญแจของ Kerberos (version 4) และ XDM (GUI ของ X-window) เป็นต้น
- improper deallocation หมายถึงจุดอ่อนที่เกิดจากโปรแกรมไม่คืนหน่วยความจำหลังจากการใช้งานเป็นผลให้ข้อมูลที่ใช้ในการประมวลผลบางส่วนยังถูกเก็บอยู่ในหน่วยความจำ สำหรับระบบปฏิบัติการยูนิกซ์ เมื่อโปรแกรมหยุดการทำงานกลางคัน ระบบปฏิบัติการจะสร้างแฟ้มชื่อ core ขึ้นมา แฟ้มดังกล่าวได้บรรจุตัวแปร รีจิสเตอร์ (register) ข้อมูลสำคัญบางอย่างไว้เพื่อใช้สำหรับการดีบัก (debug)ปัญหาที่เกิดขึ้น ผู้บุกรุกอาจจะใช้แฟ้มดังกล่าวเพื่อหาข้อมูลที่เป็นประโยชน์สำหรับการบุกรุกระบบ

จะเห็นว่ารายละเอียดบางข้อคล้ายคลึงกับงานวิจัยของ Bishop [Bishop, 1995] และพบว่าอนุกรมวิธานชุดนี้สามารถอธิบายการโจมตีระบบอย่างคร่าวๆ เท่านั้น ไม่สามารถจัดจำแนกการโจมตีระบบให้อยู่ในกลุ่มของ หนอนอินเทอร์เน็ต ไวรัส หรือโปรแกรมโทรจัน ออกจากกันได้ชัดเจน

Simon [Simon, 2005] เสนอแนวคิดของการจัดทำอนุกรมวิธานโดยใช้แนวคิดของการแบ่งเป็นมิติ (dimension) ซึ่งเป็นวิธีการจัดจำแนกรูปแบบการบุกรุกเป็นหมวดหมู่ สำหรับอนุกรมวิธานชุดนี้กำหนดให้มีการจัดหมวดหมู่ 4 มิติ ซึ่งได้แก่ first dimension, second dimension, third dimension และ forth dimension แต่ละมิติมีรายละเอียดดังนี้

First dimension

first dimension เป็นการจัดกลุ่มการบุกรุกตามชนิดของการบุกรุก เนื่องจากการกำหนดหมวดหมู่ของการบุกรุกให้ถูกต้องนั้นเป็นประเด็นที่สำคัญ ตัวอย่างเช่นการบุกรุกระบบผ่านโพรโตคอล TCP เพื่อติดตั้งโปรแกรมโทรจันในเครื่องเป้าหมายนั้น เมื่อพิจารณาเหตุการณ์ข้างต้นตามรูปแบบของการกระจายตัวของการบุกรุกถือว่า เหตุการณ์ดังกล่าวเป็นการบุกรุกด้วย หนอนอินเทอร์เน็ต แต่เมื่อพิจาณาเหตุการณ์ตามรูปแบบการทำงานถือว่าการบุกรุกดังกล่าวเป็นโปรแกรมโทรจัน ดังนั้นจึงเป็นเรื่องสำคัญที่จะต้องนิยามเหตุการณ์ที่เกิดขึ้นมาให้ถูกต้อง

นอกจากนี้เมื่อเกิดการบุกรุกขึ้นใหม่และไม่สามารถจัดกลุ่มของการบุกรุกได้ เหตุการณ์นั้นจะต้องได้รับการจัดกลุ่มที่ใกล้เคียงที่สุด ตัวอย่างกลุ่มของการบุกรุกระบบในมิตินี้ได้แก่ ไวรัส หนอนอินเทอร์เน็ต โปรแกรมโทรจัน ปัญหาการล้นของบัฟเฟอร์ การปฏิเสธการให้บริการ การโจมตีเครือข่าย การโจมตีรหัสผ่าน การโจมตีทางกายภาพ การขโมยข้อมูล เป็นต้น

Second dimension

เมื่อจัดกลุ่มการบุกรุกในมิติแรกได้แล้ว ในลำดับถัดไปต้องอธิบายถึงเป้าหมายของการบุกรุกระบบซึ่งแบ่งเป้าหมายของการบุกรุกออกเป็น 2 ระดับได้แก่ ระดับฮาร์ดแวร์และระดับซอฟต์แวร์ ซึ่งมีรายละเอียดดังนี้

- **เป้าหมายในระดับฮาร์ดแวร์** เป็นการบุกรุกในระดับกายภาพการเช่นการโจมตีอุปกรณ์เครือข่าย หรืออุปกรณ์ภายในคอมพิวเตอร์เพื่อทำลายข้อมูลที่บันทึกไว้ในดิสก์ เป็นต้น

- **เป้าหมายระดับของซอฟต์แวร์** เป้าหมายในระดับนี้ได้แบ่งออกเป็นส่วนย่อยได้แก่ การบุกรุกผ่านระบบปฏิบัติการ โปรแกรมเซิร์ฟเวอร์ เช่นหนอนอินเทอร์เน็ตชื่อ Code Red [CA-2001-19, 2001] ได้สร้างความเสียหายแก่โปรแกรมเว็บเซิร์ฟเวอร์

ชื่อ IIS ทำงานบนระบบปฏิบัติการวินโดวส์ เป็นต้น การบุกรุกผ่านโปรแกรมประยุกต์ซึ่งรวมไปถึงการบุกรุกผ่านจุดอ่อนของโพรโตคอลเครือข่าย

Third dimension

third dimension เป็นส่วนของการอธิบายจุดอ่อนของระบบ เนื่องจากจุดอ่อนของระบบปฏิบัติการหนึ่งจุด ผู้บุกรุกสามารถหาวิธีการโจมตีระบบได้มากกว่าหนึ่งวิธีการ ดังนั้นเมื่อตรวจพบจุดอ่อนในระบบจะต้องกำหนดกลุ่มของจุดอ่อนที่เกิดขึ้นมาให้ชัดเจน สำหรับการนิยามจุดอ่อนของอนุกรมวิธานชุดนี้มีสองแนวคิดคือ ในกรณีที่เป็นจุดอ่อนที่เคยเกิดขึ้นมาแล้วให้ใช้ระบบการอ้างอิงจุดอ่อนจาก CVE แต่ถ้าเป็นจุดอ่อนที่เกิดขึ้นใหม่ให้นิยามจุดอ่อนดังกล่าวตามแนวคิดของ Howard

Forth dimension

จากที่กล่าวมาแล้วในมิติก่อนหน้า จุดอ่อนของระบบอาจจะมีช่องทางการบุกรุกมากกว่าหนึ่งช่องทาง การจัดหมวดหมู่ในกลุ่มนี้จึงเป็นการนิยามวิธีการบุกรุกเพิ่มเติมจากมิติที่ 1 เนื่องจากในมิติที่ 1 นั้นเป็นการนิยามการบุกรุกตามเป้าหมายหรือลักษณะเด่นของการบุกรุก แต่บางการบุกรุกมีวิธีการบุกรุกมากกว่าหนึ่งวิธี ดังตัวอย่าง หนอนอินเทอร์เน็ตได้สั่งงานโปรแกรมโทรจันไว้เพื่อรอรับการเชื่อมต่อที่ไม่มีกระบวนการตรวจสอบตัวตน พบว่านอกจากการก่อกวนระบบเครือข่ายด้วยหนอนอินเทอร์เน็ตแล้วยังก่อกวนเครื่องเป้าหมายด้วยโปรแกรมโทรจันอีกด้วย จากที่กล่าวมาข้างต้น การจำแนกในมิติที่ 1 เป็นการจัดเหตุการณ์ดังกล่าวเป็นการบุกรุกด้วยหนอนอินเทอร์เน็ตและระบุว่าเหตุการณ์นั้นเป็นการบุกรุกด้วยโปรแกรมโทรจันในมิติที่ 4

ตัวอย่างการจัดอนุกรมวิธานตามแนวคิดของ Simon แสดงไว้ในตารางที่ 2.1 อธิบายได้ว่าการบุกรุกระบบด้วย Nimda ซึ่งเป็นหนอนอินเทอร์เน็ตแบบกระจายตัวด้วยอีเมล์ (mass-mailing worm) เป้าหมายของการบุกรุกคือโปรแกรม Internet explore รุ่น 5.5 SP1 นอกจากที่ทำหน้าที่เหมือนหนอนอินเทอร์เน็ตแล้วยังเป็นโปรแกรมไวรัสแบบกระจายด้วยแฟ้ม โปรแกรมโทรจันและก่อกวนระบบด้วยเทคนิค DoS (ศึกษารายละเอียดจาก [ศุภโชค, 2548])รายละเอียดทางเทคนิคอื่นๆ ของหนอนอินเทอร์เน็ตตัวนี้ศึกษาเพิ่มเติมจาก CVE-2001-0333 เป็นต้น

ตารางที่ 2.1 แสดงการจัดอนุกรมวิธานของการบุกรุกตามแนวคิดของ Simon [Simon, 2005]

attack	1st dimension	2nd dimension	3rd dimension	4th dimension
Blaster	network-aware worm	MS Windows family	CAN-2003-0352	TCP packet flooding DoS
Chernobyl	File infector virus	MS Windows 98		corruption of information
John the Ripper	guessing password attack	UNIX family	configuration	disclosure of information
Melissa	mass-mailing worm	MS Word 97	configuration	macro virus & TCP flooding DoS
Nimda	mass-mailing worm	MS IE 5.5 SP1	CVE-2001-0333	file infector virus, Trojan and Dos

จากการศึกษาเรื่องการจัดอนุกรมวิธานพบว่า รูปแบบการบุกรุกระบบมีมากมายหลายวิธีการ แต่เมื่อได้รับการจัดหมวดหมู่ออกเป็นกลุ่มและศึกษาถึงผลกระทบที่จะเกิดขึ้นแล้ว ทำให้ทราบว่าช่องทางของการบุกรุกระบบมีไม่กี่ช่องทางเพื่อให้ได้มาซึ่งสิทธิ์ในการเข้าใช้ทรัพยากรอย่างไม่จำกัด

2.4 ระบบตรวจจับการบุกรุก

Bace [Bace, 2001] ได้ให้นิยามระบบตรวจจับการบุกรุกไว้ว่าระบบตรวจจับการบุกรุกคือ ระบบที่ประกอบด้วยฮาร์ดแวร์หรือซอฟต์แวร์สำหรับทำงานในกระบวนการตรวจสอบเหตุการณ์ต่างๆ ที่เกิดขึ้นในระบบคอมพิวเตอร์และเครือข่ายเพื่อวิเคราะห์หาร่องรอยของการบุกรุกโดยอัตโนมัติ วิธีการตรวจจับการบุกรุกมีมากมายหลายวิธีการโดยแต่ละวิธีการมีความแตกต่างกัน แต่ระบบตรวจจับการบุกรุกแต่ละระบบมีองค์ประกอบพื้นฐานที่เหมือนกันได้แก่ information source, analysis และ response โดยแต่ละองค์ประกอบมีรายละเอียดดังนี้

- **information source** คือข้อมูลที่จะนำมาใช้เพื่อการวิเคราะห์กิจกรรมต่างๆ ที่เกิดขึ้นว่าเป็นการบุกรุกระบบหรือเหตุการณ์ปกติ ข้อมูลเหล่านี้อาจนำมาจากล็อกไฟล์ของระบบปฏิบัติการ ข้อมูลในหน่วยความจำในขณะที่เกิดเหตุการณ์ที่ต้องตรวจ

สอบ หรือข้อมูลในเครือข่ายในกรณีที่ต้องการติดตามการทำงานของการสื่อสาร ข้อมูลที่น่าสงสัย เป็นต้น

- **analysis** คือส่วนของการวิเคราะห์ตัดสินกิจกรรมของระบบโดยพิจารณาจากข้อมูลที่ได้มาจาก source เพื่อที่จะสรุปว่าเหตุการณ์หรือการกระทำใดที่ชี้บ่งว่าเป็นการบุกรุกหรือเกิดการบุกรุกแล้วในระบบ แนวทางของการวิเคราะห์เหตุการณ์นั้นมีสองรูปแบบคือ anomaly detection และ misuse detection ซึ่งจะกล่าวในลำดับถัดไป
- **response** คือส่วนของการตอบสนองต่อเหตุการณ์บุกรุกที่เกิดขึ้นซึ่งแบ่งรูปแบบของการตอบสนองออกเป็นสองแบบคือ การตอบสนองแบบ active และการตอบสนองแบบ passive โดยการตอบสนองแบบ active นั้นเป็นการตอบสนองต่อเหตุการณ์แบบทันทีทันใด เช่น ทำลายโปรเซสที่ทำการบุกรุก ตัดการเชื่อมต่อเครือข่ายที่เป็นที่มาของการบุกรุก หรือปิดพอร์ตหรือโพรโตคอลที่ได้รับผลกระทบ เป็นต้น สำหรับการตอบสนองแบบ passive นั้นเป็นการรายงานหรือแจ้งเตือนการบุกรุกไปยังผู้รับผิดชอบเพื่อแก้ไขปัญหา

2.4.1 แนวทางของการตรวจจับการบุกรุก

ศุภโชคและคณะ[ศุภโชค, 2548] ได้ศึกษาถึงแนวทางของการตรวจจับการบุกรุก ซึ่งกล่าวโดยสรุปคือ แนวทางการตรวจจับการบุกรุกแบ่งออกเป็นสองแบบ คือ anomaly detection และ misuse detection ซึ่งแนวทางแต่ละแบบนั้นมีรายละเอียดดังนี้

Anomaly detection

การตรวจจับการบุกรุกด้วยแนวทาง anomaly detection ถือว่ากิจกรรมการบุกรุกทั้งหมดเป็นกิจกรรมที่ผิดปกติ ในการตรวจจับการบุกรุกจึงต้องแยกกิจกรรมการทำงานปกติออกมาและกำหนดให้กิจกรรมอื่นนั้นเป็นกิจกรรมที่ผิดปกติซึ่งถือว่าเป็นการบุกรุก ดังนั้นในระบบตรวจจับการบุกรุกจะต้องมีส่วนที่เก็บกิจกรรมหรือพฤติกรรมการใช้งานปกติซึ่งจะเก็บข้อมูลกิจกรรมหรือพฤติกรรมปกติของผู้ใช้ เครื่องคอมพิวเตอร์ หรือการเชื่อมต่อเครือข่าย เป็นต้น ข้อมูลต่างๆ เหล่านี้ถูกสร้างมาจากข้อมูลหรือประวัติการใช้งานในช่วงการทำงานปกติ หลังจากนั้นตัวตรวจจับจะเก็บข้อมูลเหตุการณ์ต่างๆ ในขณะเวลาหนึ่งๆ และใช้เกณฑ์ในการชี้วัดต่างๆ เช่นตัวชี้วัดทางสถิติ ตัวชี้วัดด้านความถี่ เป็นต้น เพื่อบ่งบอกว่ากิจกรมใดที่พฤติกรรมมีความผิดปกติ นั่นคือหากมีการกระทำที่ต่างจากที่กำหนดไว้ในระบบที่ระดับนัยสำคัญทางสถิติที่ระบุไว้ก็จะตีความว่าเป็นการพยายามที่จะบุกรุก หากพิจารณาถึงการกระทำที่เป็นการบุกรุกและการกระทำที่ผิดปกติจะพบว่ามี

ความเป็นไปได้ที่การกระทำที่ผิดปกติแต่ไม่ใช่การบุกรุก ระบบตรวจจับการบุกรุกจะถือว่ากิจกรรมที่เกิดขึ้นนั้นเป็นการบุกรุกระบบ (เกิดปัญหา false positive) และการกระทำที่เป็นการบุกรุกแต่ระบบตรวจจับการบุกรุกไม่ได้ระบุว่าเป็นการกระทำที่ผิดปกติ (เกิดปัญหา false negative) ซึ่งเป็นปัญหาที่สำคัญมากของแนวทางนี้

ประเด็นที่สำคัญอีกประการหนึ่งสำหรับแนวทางของการตรวจจับการบุกรุกแนวทางนี้ คือ การเลือกระดับของการกระทำที่จะถือว่าเป็นการบุกรุก นอกจากนี้แนวทางดังกล่าวยังเป็นระบบที่มีค่าใช้จ่ายสูงเนื่องจากต้องทำการเก็บข้อมูลสำหรับการตรวจสอบและอาจต้องปรับปรุงเกณฑ์ของตัวชี้วัดพฤติกรรมของระบบบ่อยๆ เพื่อให้การตรวจสอบมีความถูกต้องมากขึ้น สำหรับเทคนิคของการตรวจจับการบุกรุกที่ใช้แนวทาง anomaly detection มีดังนี้ statistical approach, predictive pattern generation และ neural network เป็นต้น รายละเอียดของการบุกรุกแต่ละวิธีการศึกษาเพิ่มเติมจากรายงานของ ศุภโชค [ศุภโชค, 2548]

Misuse detection

หลักการของการตรวจจับการบุกรุกด้วยแนวทาง misuse detection คือ ตัวตรวจจับการบุกรุกจะวิเคราะห์กิจกรรมของระบบ โดยการพิจารณาเหตุการณ์หรือชุดของเหตุการณ์ที่ตรงกับรูปแบบของเหตุการณ์ที่ได้กำหนดไว้แล้วว่าเป็นการบุกรุก โดยอธิบายถึงการบุกรุกต่างๆ ที่รู้จัก รูปแบบของเหตุการณ์การบุกรุกที่รู้จักเหล่านี้จะเรียกว่า ร่องรอยการบุกรุก (signatures) ดังนั้นบางครั้งจึงเรียกแนวทาง misuse detection ว่า signature-based detection แนวทางนี้จะใช้ข้อมูลความรู้เกี่ยวกับพฤติกรรมที่เกี่ยวกับการบุกรุก และค้นหาเพื่อตรวจสอบพฤติกรรมเหล่านี้โดยตรงซึ่งตรงกันข้ามกับ anomaly detection ที่จะค้นหาเพื่อตรวจจับส่วนตรงกันข้ามกับพฤติกรรมที่ปกติ

ประเด็นสำคัญของการตรวจจับแนวทางนี้ก็คือ ระบบตรวจจับการบุกรุกจะนิยามรูปแบบหรือร่องรอยของการบุกรุกอย่างไรให้ครอบคลุมถึงความหลากหลายทั้งหมดที่เป็นไปได้ของการบุกรุก และเขียนรูปแบบหรือร่องรอยการบุกรุกอย่างไรที่จะไม่ตรงกับกิจกรรมที่ไม่ใช่การบุกรุก สำหรับเทคนิคของการบุกรุกตามแนวทางนี้ได้แก่ production/expert system, keystroke monitoring, model based intrusion detection และ state transition analysis เป็นต้น รายละเอียดของการบุกรุกแต่ละวิธีการศึกษาเพิ่มเติมจากรายงานของศุภโชค [ศุภโชค, 2548]

2.4.2 คุณลักษณะของระบบตรวจจับการบุกรุกที่ดี

วิธีการตรวจจับการบุกรุกมีหลายวิธีการแต่ไม่มีระบบใดที่สามารถตรวจจับการบุกรุกได้ครอบคลุมทุกปัญหา แต่ละระบบมีจุดเด่นและข้อจำกัดที่แตกต่างกัน Price จึงได้เสนอสมบัติของระบบตรวจจับการบุกรุกที่ดีโดยสรุปว่า ระบบตรวจจับการบุกรุกที่ดีควรมีความสามารถและคุณสมบัติดังต่อไปนี้ [Price, 1998]

- **run continually** หมายถึงระบบตรวจจับการบุกรุกจะต้องทำงานอยู่ตลอดเวลาโดยไม่ต้องมีการควบคุมของผู้ดูแลระบบ และต้องมีความน่าเชื่อถือเพียงพอที่จะทำงานในลักษณะอยู่เบื้องหลัง (background process) แต่ผู้ดูแลระบบต้องสามารถตรวจสอบการทำงานของระบบตรวจจับการบุกรุกได้
- **fault tolerant** หมายถึงระบบตรวจจับการบุกรุกยังคงสามารถที่จะทำงานต่อไปได้ในกรณีที่เครื่องคอมพิวเตอร์เกิดปัญหาหรือข้อผิดพลาด และไม่ต้องมีการสร้างฐานความรู้ทุกครั้งที่เริ่มงาน
- **subversion resistance** หมายถึงระบบตรวจจับการบุกรุกมีความสามารถในการตรวจสอบตัวเองเพื่อไม่ให้ถูกลบ ทำลาย แก้ไข หรือถูกแทนที่ด้วยโปรแกรมอื่น
- **minimal overhead** หมายถึงการทำงานของระบบตรวจจับการบุกรุกจะต้องไม่ส่งผลกระทบต่อการทำงานของอุปกรณ์คอมพิวเตอร์นั่นคือระบบดังกล่าวใช้ทรัพยากรของระบบคอมพิวเตอร์น้อยที่สุดเท่าที่จะทำได้ เพราะถ้าหากระบบตรวจจับการบุกรุกทำให้ระบบคอมพิวเตอร์ทำงานช้าลงแล้วก็จะส่งผลการทำงานของระบบตรวจจับการบุกรุก
- **observe deviations** หมายถึงระบบตรวจจับการบุกรุกจะต้องตรวจสอบการทำงานที่ผิดไปจากรูปแบบการทำงานปกติของระบบตรวจจับการบุกรุกเองได้
- **easily tailored** หมายถึงการปรับเปลี่ยนหรือแก้ไขระบบตรวจจับการบุกรุกให้เข้ากับระบบคอมพิวเตอร์ต้องทำได้ง่าย เนื่องจากคอมพิวเตอร์แต่ละแบบจะมีรูปแบบการใช้งานและกลไกในการป้องกันที่ต่างกัน
- **changing system behavior** หมายถึงระบบตรวจจับการบุกรุกสามารถปรับการทำงานให้สอดคล้องกับการเปลี่ยนแปลงพฤติกรรมการใช้งานของระบบคอมพิวเตอร์ได้ เช่นเมื่อมีการติดตั้งโปรแกรมชุดใหม่ให้แก่ระบบคอมพิวเตอร์ ระบบตรวจจับการบุกรุกก็ต้องสามารถปรับเปลี่ยนการทำงานให้เข้ากับระบบที่เปลี่ยนไปได้

2.4.3 คุณลักษณะของการตรวจจับการบุกรุกที่เหมาะสมสำหรับการพัฒนากลไกการตรวจจับการบุกรุกในระบบปฏิบัติการ

ในหัวข้อที่ 2.4.1 และ 2.4.2 เป็นการกล่าวถึงวิธีการตรวจจับการบุกรุกและคุณลักษณะของระบบตรวจจับการบุกรุกที่ดี หลังจากที่ศึกษาแนวคิดข้างต้นแล้วจึงพบว่าระบบตรวจจับการบุกรุกที่เหมาะสมสำหรับการพัฒนากลไกการตรวจจับการบุกรุกในระบบปฏิบัติการควรมีคุณลักษณะตามเงื่อนไขที่กล่าวมาแล้วในหัวข้อที่ 2.4.2 แต่การทำงานในระดับของระบบปฏิบัติการนั้นไม่จำเป็นต้องทำงานตลอดเวลาเหมือนกับการทำงานในระดับของโปรเซสแต่อาจจะตรวจสอบกิจกรรมอื่นทุกครั้งที่เป็นไปตามเงื่อนไข การทำงานของระบบตรวจจับการบุกรุกจะต้องคงทนไม่ถูกทำลายได้ง่าย เนื่องจากถ้ากระบวนการตรวจจับการบุกรุกถูกทำลายแล้ว ระบบปฏิบัติการจะได้รับผลกระทบที่รุนแรงกว่าระบบตรวจจับการบุกรุกในระดับโปรแกรมประยุกต์ อีกทั้งระยะเวลาของการตรวจจับการบุกรุกต้องเกิดขึ้นก่อนที่การบุกรุกจะประสบความสำเร็จ

วิธีการตรวจจับการบุกรุกดังที่กล่าวมาแล้วหลายวิธีการซึ่งแต่ละวิธีการมีข้อจำกัดที่แตกต่างกันออกไป เช่น

- **วิธีการวิเคราะห์ล็อกไฟล์** มีข้อจำกัดทางด้านข้อมูลนำเข้าสำหรับการวิเคราะห์เหตุการณ์เนื่องจากข้อมูลเหล่านั้นจะเกิดขึ้นเมื่อเกิดเหตุการณ์ไปแล้ว ดังนั้นจึงไม่เหมาะที่จะนำแนวคิดนี้มาพัฒนาในระดับของระบบปฏิบัติการ
- **วิธีการวิเคราะห์รูปแบบ** มีข้อจำกัดที่ว่ารูปแบบของเหตุการณ์บุกรุกจะต้องเป็นเหตุการณ์ที่เกิดขึ้นมาก่อน การตรวจจับการบุกรุกจึงจะสามารถตรวจจับได้ แม้ว่าจะประยุกต์แนวคิดของโครงข่ายประสาทเทียมมาตรวจจับการบุกรุกแต่ถ้ารูปแบบที่สอนไปนั้นไม่ครอบคลุม หรือเป็นรูปแบบที่ถูกสอนโดยผู้บุกรุก การตรวจจับการบุกรุกย่อมทำงานผิดพลาด
- **การวิเคราะห์การเปลี่ยนแปลงสถานะ** เป็นการแทนเหตุการณ์ให้อยู่ในรูปแบบสถานะ แล้วตรวจสอบการเปลี่ยนแปลงสถานะตามนโยบาย (policy) ที่วางไว้โดยผู้ดูแลระบบ ถ้าหากการนโยบายนั้นครอบคลุมกับวิธีการบุกรุกแล้ว การตรวจจับการบุกรุกตามแนวทางนั้นประสบความสำเร็จ แต่ถ้าหากการออกแบบนโยบายไม่รัดกุมแล้วการตรวจจับการบุกรุกจะไม่มีประสิทธิภาพ นอกจากนี้การทำงานของโปรเซสตรวจจับการบุกรุกต้องการทรัพยากรจำนวนมากในการติดตามการทำงานและพิจารณากิจกรรม

วิทยานิพนธ์ชุดนี้ได้ศึกษาและวิเคราะห์การตรวจจับการบุกรุกด้วยวิธีการวิเคราะห์การเปลี่ยนแปลงสถานะแล้วพบว่าแนวคิดนี้เหมาะที่จะพัฒนาในระดับของระบบปฏิบัติการ เนื่อง

จากวิธีการนี้ไม่จำเป็นต้องสร้างฐานความรู้หรือฐานข้อมูล การทำงานของขั้นตอนการตรวจจับสามารถทำงานแบบเวลาจริงได้ แม้ว่าการทำงานของระบบต้องใช้ทรัพยากรจำนวนมากแต่ถ้าหากการออกแบบและพัฒนาระบบนั้นถูกต้องและรัดกุมแล้ว ประสิทธิภาพของระบบปฏิบัติการที่เปลี่ยนแปลงไปจะอยู่ในระดับที่ยอมรับได้ นอกจากนี้การกำหนดนโยบายของการตรวจจับก็มีความสำคัญ นโยบายดังกล่าวต้องครอบคลุมและได้รับการทดสอบจนแน่ใจว่า นโยบายนั้นไม่ก่อให้เกิดผลกระทบและจุดอ่อนแก่ระบบปฏิบัติการ การกำหนดนโยบายของวิทยานิพนธ์ชุดนี้ได้ศึกษาแนวคิดและวิธีการตรวจจับเหตุการณ์จากงานวิจัยของ Nuansri [Nuansri, 1999] เรื่อง “Process State Transition Analysis Technique and its Application to Intrusion Detection System” สำหรับรายละเอียดของงานวิจัยชุดนี้จะกล่าวถึงในหัวข้อที่ 2.6

2.5 ระบบปฏิบัติการยูนิกซ์

ระบบปฏิบัติการยูนิกซ์เป็นเครื่องมือหลักในวิทยานิพนธ์ชิ้นนี้ ในรายละเอียดจะกล่าวถึงโปรแกรมและโปรเซส สถาปัตยกรรมของระบบปฏิบัติการยูนิกซ์ ซิซเท็มคอล (system call) และการควบคุมการเข้าถึงของระบบปฏิบัติการ สำหรับรายละเอียดของแต่ละหัวข้อนั้นกล่าวตามลำดับดังนี้

2.5.1 โปรแกรมและโปรเซสของระบบปฏิบัติการยูนิกซ์

โปรแกรม (program) [Bach, 1986] คือชุดคำสั่งที่พร้อมจะถูกสั่งงานโดยถูกบันทึกอยู่ในหน่วยความจำสำรองเช่น ดิสก์ เทป เป็นต้น ชุดคำสั่งเหล่านั้นอาจจะเขียนอยู่ในรูปของภาษาเครื่องหรือโปรแกรมสคริปต์ (script)

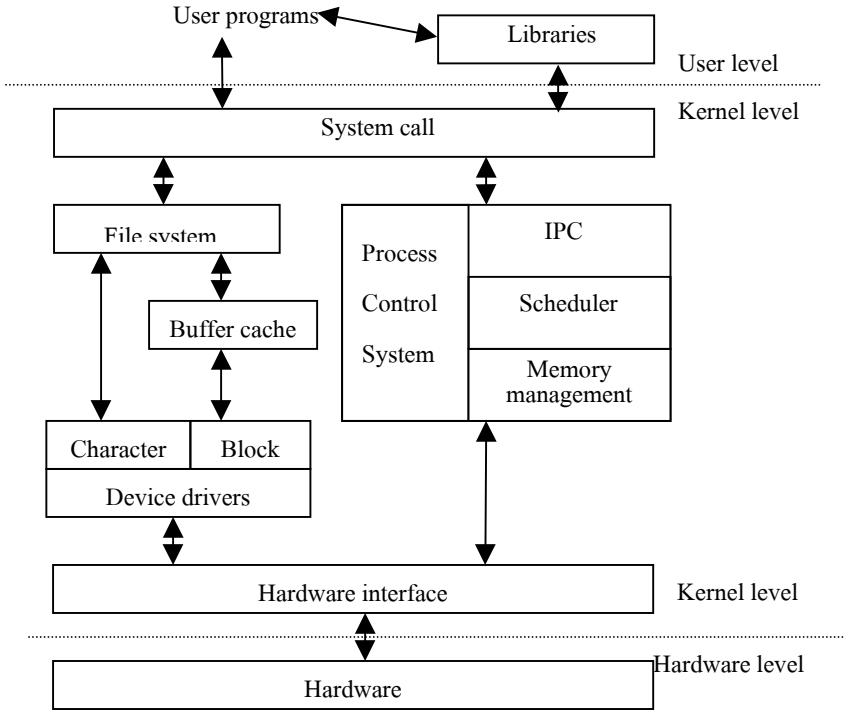
โปรเซส (process) [Bach, 1986] คือโปรแกรมที่ถูกสั่งงานและจัดเก็บอยู่ในหน่วยความจำหลัก ซึ่งประกอบไปด้วยส่วนต่างๆ ได้แก่

- **text segment** เป็นส่วนของชุดคำสั่งซึ่งเก็บอยู่ในรูปแบบบิตและพร้อมที่จะถูกดำเนินการโดยหน่วยประมวลผลกลาง
- **data segment** เป็นส่วนของหน่วยความจำที่ใช้สำหรับเก็บข้อมูลที่ใช้ในการประมวลผลเช่น การประกาศตัวแปรแบบโกลบอล (global) ในภาษาซี
- **stack segment** เป็นส่วนของหน่วยความจำที่ถูกจองไว้เพื่อดำเนินการเกี่ยวกับการเรียกใช้ฟังก์ชัน การส่งผ่านค่าฟังก์ชัน หรือใช้เก็บข้อมูลเมื่อมีการของหน่วยความจำแบบไดนามิค (dynamic) ของโปรเซส

ระบบปฏิบัติการยูนิกซ์รองรับการสั่งงานโปรเซสมากกว่าหนึ่งโปรเซสให้ทำงานพร้อมกันโดยมีระบบปฏิบัติการเป็นตัวจัดการลำดับการเข้าใช้หน่วยประมวลผลกลาง โปรแกรมหนึ่งโปรแกรมสามารถถูกสั่งงานซึ่งก่อให้เกิดโปรเซสได้มากกว่าหนึ่งโปรเซส แต่ละโปรเซสมีส่วนของชุดคำสั่ง ข้อมูล และแสตกของตัวเอง โปรเซสหนึ่งไม่สามารถเข้าถึงหน่วยความจำแสตกของอีกโปรเซสหนึ่งได้ แต่สามารถสื่อสารกันได้โดยการสื่อสารกันระหว่างโปรเซส (Inter Process Communication หรือ IPC)

2.5.2 สถาปัตยกรรมของระบบปฏิบัติการยูนิกซ์

ระบบปฏิบัติการยูนิกซ์ได้รับการออกแบบตามสถาปัตยกรรมแบบชั้น (layered architecture) โดยมีการทำงานขององค์ประกอบแต่ละชั้นประสานกันตามภาพประกอบที่ 2.3



ภาพประกอบ 2.3 สถาปัตยกรรมแบบชั้นของระบบปฏิบัติการยูนิกซ์ [Bach, 1986]

โดยภาพรวมระบบปฏิบัติการจะแบ่งออกเป็นสามชั้น คือชั้นของผู้ใช้ (user) ชั้นของเคอร์เนล (kernel) และชั้นของฮาร์ดแวร์ (hardware) โดยแต่ละชั้นมีส่วนประกอบและหน้าที่ดังนี้ [Bach, 1986]

User level

ในขณะที่โปรเซสกำลังทำงาน โปรเซสเหล่านั้นต้องการที่จะเข้าใช้ทรัพยากรของระบบ เช่น หน่วยประมวลผลกลาง หน่วยความจำหลัก หรือระบบแฟ้ม เป็นต้น โปรเซสจะต้องร้องขอทรัพยากรที่ต้องการผ่านฟังก์ชันของระบบปฏิบัติการ หลังจากนั้นฟังก์ชันของระบบจะติดต่อกับระบบปฏิบัติการอีกครั้งหนึ่ง ทุกโปรเซสในระบบไม่สามารถเข้าถึงทรัพยากรของระบบได้โดยตรง จำเป็นต้องขอใช้ทรัพยากรผ่านซิซเท็มคอล (ซึ่งจะอธิบายในหัวข้อที่ 2.5.3) หรือร้องขอผ่านชุดไลบรารี (library) หลังจากนั้นชุดไลบรารีจะติดต่อกับระบบปฏิบัติการผ่านซิซเท็มคอลอีกชั้นหนึ่ง

Kernel level

เคอร์เนล (kernel) เป็นส่วนหนึ่งของระบบปฏิบัติการ มีหน้าที่ในการจัดการทรัพยากรในระบบเช่น การจัดการระบบแฟ้มข้อมูล การจัดการหน่วยความจำ การจัดลำดับการทำงานของโปรเซส การจัดการการติดต่อกับ I/O โปรเซสสามารถเข้าถึงทรัพยากรได้ทุกชนิดถ้าหากได้รับสิทธิ์ แต่โปรเซสดังกล่าวต้องร้องขอทรัพยากรผ่านซิซเท็มคอลดังที่กล่าวมาแล้วก่อนหน้านี้ เคอร์เนลจะตรวจสอบสิทธิ์ของโปรเซส ถ้าหากโปรเซสดังกล่าวได้รับอนุญาตแล้ว เคอร์เนลจะส่งต่อคำร้องขอไปยัง hardware interface หรือ device driver

Hardware level

ชั้นของฮาร์ดแวร์เป็นชั้นที่อยู่ระดับล่างสุดของระบบคอมพิวเตอร์ เมื่อเคอร์เนลต้องการที่จะเข้าถึงทรัพยากรใดๆ เคอร์เนลจะติดต่อไปยังอุปกรณ์เหล่านั้นผ่านทาง device driver ของอุปกรณ์ เพื่อติดต่อกับ Hardware Control ซึ่งเป็นซอฟต์แวร์ที่ขึ้นอยู่กับอุปกรณ์

2.5.3 ซิซเท็มคอลของระบบปฏิบัติการ

ซิซเท็มคอล (system call) เป็นฟังก์ชันของระบบปฏิบัติการซึ่งเป็นตัวเชื่อม (interface) ระหว่าง user space และ kernel space ของระบบปฏิบัติการ สภาวะแวดล้อมของระบบปฏิบัติการแบ่งออกเป็นสถานะ คือ user space และ kernel space โดยแต่ละส่วนมีรายละเอียดดังนี้

- **user space** เป็นสภาวะแวดล้อมที่ถูกจำกัดสิทธิ์ โปรเซสของผู้ใช้ทุกโปรเซสทำงานในสภาวะนี้ โปรเซสเหล่านี้ไม่สามารถที่จะเข้าถึงทรัพยากรของระบบได้โดยตรง จำเป็นต้องร้องขอผ่านซิซเท็มคอล
- **kernel space** เป็นสภาวะที่โปรเซสสามารถเข้าใช้ทรัพยากรของระบบได้ โปรเซสในสภาวะนี้เป็นโปรเซสของเคอร์เนลมีหน้าที่คอยจัดการทรัพยากรต่างๆ เช่น ส่วนของการนำเข้า/ส่งออก (input/output) โปรเซสที่ทำงานในสภาวะนี้จะมีสิทธิ์พิเศษที่จะจัดการกับทรัพยากรที่ต้องการได้ ถ้าหากโปรเซสของผู้ใช้ต้องการทำงานในสภาวะนี้ โปรเซสดังกล่าวต้องร้องขอข้อมูลผ่านซิซเท็มคอล

เมื่อซิซเท็มคอลถูกเรียกใช้แล้ว ค่าพารามิเตอร์ของซิซเท็มคอลจะถูกตรวจสอบโดยเคอร์เนล ก่อนที่จะถูกนำไปใช้งาน ถ้าหากโปรเซสนั้นผ่านกระบวนการตรวจสอบแล้วค่าพารามิเตอร์จะถูกคัดลอกและส่งผ่านไปยัง kernel space เพื่อดำเนินงานตามชุดคำสั่งของซิซเท็มคอล หลังจากที่ดำเนินการเสร็จเรียบร้อยแล้วเคอร์เนลจะส่งผลการทำงานกลับไปยังโปรเซสที่เรียกใช้ซิซเท็มคอลซึ่งอยู่ใน user space ผู้พัฒนาระบบต้องตรวจสอบค่าที่ถูกส่งกลับมาว่าเกิดข้อผิดพลาดหรือทำงานสำเร็จตามวัตถุประสงค์

จากภาพประกอบที่ 2.3 แสดงให้เห็นว่า ซิซเท็มคอลเป็นช่องทางเดียวที่โปรเซสของผู้ใช้สามารถใช้ติดต่อกับระบบปฏิบัติการ ไม่ว่าผู้ใช้จะเรียกใช้โปรเซสใดๆ แต่ในที่สุดแล้วโปรเซสเหล่านั้นต้องร้องขอทรัพยากรผ่านซิซเท็มคอลเดียวกัน

2.5.4 โปรแกรมแบบ setuid/setgid

setuid เป็นชื่อย่อของ Set User ID เป็นการกำหนดสิทธิ์การเข้าถึงให้แก่แฟ้มบนระบบปฏิบัติการยูนิกซ์ การกำหนดสิทธิ์ดังกล่าวเป็นการอนุญาตให้ผู้ใช้บนระบบปฏิบัติการสามารถสั่งงานโปรแกรมด้วยสิทธิพิเศษชั่วคราวเพื่อที่จะดำเนินงานพิเศษบางอย่างที่ผู้ใช้ปกติไม่สามารถทำได้ เช่น คำสั่ง passwd ใช้สำหรับเปลี่ยนรหัสผ่านของผู้ใช้ เนื่องจากฐานข้อมูลผู้ใช้ ผู้ที่มีสิทธิ์ในการแก้ไขคือ root ผู้ใช้จึงต้องการสิทธิพิเศษสำหรับการแก้ไขข้อมูลดังกล่าว เป็นต้น

การทำงานของโปรแกรมแบบ setgid มีลักษณะที่คล้ายคลึงกันกับโปรแกรมแบบ setuid แต่ setgid ซึ่งย่อมาจาก Set Group ID เป็นการพิจารณาสิทธิ์ของโปรเซสด้วยเลขประจำกลุ่ม ดังนั้นเพื่อให้เข้าใจตรงกันวิทยานิพนธ์ชุดนี้จึงขอใช้คำว่า “โปรแกรม setuid” แทนคำว่า ”โปรแกรม setuid/setgid” ในที่นี้ถือว่าคำทั้งสองคำนี้มีความหมายเหมือนกัน

2.6 การตรวจจับการบุกรุกด้วยวิธีการวิเคราะห์การเปลี่ยนแปลงสถานะของโปรเซส

กิจกรรมต่างๆ บนระบบปฏิบัติการยูนิกซ์ถูกแทนด้วยโปรเซสหรือกลุ่มของโปรเซสของระบบ โปรเซสเหล่านั้นมีหมายเลขประจำตัว (User ID หรือ UID) และประจำกลุ่ม (Group ID หรือ GID) โดยที่ค่าทั้งสองค่านี้จะแสดงถึงความเป็นเจ้าของของโปรเซส นอกจากนี้โปรเซสยังมี

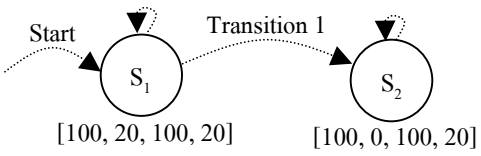
ค่า Effective user ID หรือ EUID และ Effective group ID หรือ EGID โดยค่าทั้งสองค่าเป็นค่าที่บอกถึงสิทธิ์ในการเข้าใช้ทรัพยากร ค่าประจำโปรเซสทั้งสี่ค่านี้เรียกว่า user credential สำหรับผู้ใช้ปกติแล้วจะถูกจำกัดสิทธิ์ของการเข้าใช้ทรัพยากร ยกเว้นผู้ใช้ชื่อ root ซึ่งมีค่า UID และ GID เป็น 0 มีสิทธิ์ในการเข้าใช้ทรัพยากรทั้งหมดของระบบปฏิบัติการ ถ้าหากโปรเซสของผู้ใช้ปกติมีค่า EUID หรือ EGID เป็น 0 แล้ว โปรเซสดังกล่าวมีสิทธิ์ในการเข้าใช้ทรัพยากรครึ่งหนึ่งของ root แต่ในบางกรณีผู้ใช้ปกติต้องการสิทธิ์พิเศษในการทำงานจึงจำเป็นที่จะต้องเปลี่ยนสิทธิ์ของโปรเซสให้มีสิทธิ์ที่เท่ากับ root ชั่วคราวเพื่อทำงานตามวัตถุประสงค์ หลังจากที่โปรเซสทำงานตามวัตถุประสงค์เรียบร้อยแล้ว โปรเซสต้องเปลี่ยนค่า user credential ให้กลับไปเป็นค่าเดิมเพื่อทำงานอื่นๆ ต่อไป

เมื่อโปรเซสถูกสั่งงาน (execute) ค่าประจำโปรเซสทั้ง 4 ค่า ได้แก่ UID, GID, EUID และ EGID จะถูกกำหนดโดยระบบปฏิบัติการ โดยที่ EUID มีค่าเท่ากับ UID และ EGID มีค่าเท่ากับ GID ในขณะที่โปรเซสปกติกำลังถูกกระทำการ ค่าประจำโปรเซสทั้งสี่ค่าจะไม่เปลี่ยนแปลง เว้นแต่โปรเซสดังกล่าวเป็นโปรเซสที่ต้องการสิทธิพิเศษสำหรับเข้าใช้ทรัพยากรที่ถูกจำกัดสิทธิ์ไว้เฉพาะ root เช่น ฐานข้อมูลผู้ใช้ โปรเซสดังกล่าวจำเป็นที่จะต้องเปลี่ยนสิทธิ์ของตัวเองให้เป็น root เมื่อโปรเซสนั้นดำเนินการด้วยสิทธิพิเศษเรียบร้อยแล้ว โปรเซสดังกล่าวจะต้องเปลี่ยนสิทธิ์ของตัวเองให้เท่ากับตอนเริ่มดำเนินการ โปรเซสที่ต้องเปลี่ยนแปลงสิทธิ์ในระหว่างการดำเนินงานนั้นเรียกว่าโปรเซสแบบ setuid

Nuansri [Nuansri, 1999] ได้ศึกษาและสรุปการเปลี่ยนแปลงค่าประจำโปรเซสโดยแบ่งโปรเซสออกเป็นสถานะโดยพิจารณาจากค่า user credential ของโปรเซส อีกทั้งกำหนดกฎสนับสนุนขึ้นมาเพื่อพิจารณากิจกรรมต่างๆ ที่เกิดขึ้นในระบบว่าเป็นเหตุการณ์ปกติหรือผิดปกติ สำหรับรายละเอียดของการนิยามสถานะและกฎสนับสนุนนั้นจะกล่าวในลำดับถัดไป

2.6.1 การนิยามสถานะ

ในขณะที่โปรเซสกำลังทำงานนั้น แต่ละโปรเซสมีค่า user credential ประจำโปรเซสจำนวน 4 ค่า ได้แก่ UID, GID, EUID และ EGID สำหรับวิทยานิพนธ์ชุดนี้จะแทนค่า user credential เหล่านี้ด้วยสัญลักษณ์ [UID, GID, EUID, EGID] และวิทยานิพนธ์ชุดนี้เรียกการแทนสัญลักษณ์ดังกล่าวว่า “ค่าประจำสถานะ” ตัวอย่างการนิยามสถานะของโปรเซสแสดงไว้ดังภาพประกอบที่ 2.4



ภาพประกอบ 2.4 การเปลี่ยนแปลงสถานะของโปรเซส

จากภาพประกอบที่ 2.4 เมื่อผู้ใช้ xyz ซึ่งมีค่า UID และค่า GID เป็น 100 และ 20 ตามลำดับ ได้สั่งงานโปรเซสขึ้นมาหนึ่งโปรเซส โดยโปรเซสนั้นจะได้รับการกำหนดค่าประจำสถานะเป็น [100, **20**, 100, 20] หมายถึงโปรเซสนั้นอยู่ในสถานะที่ 1 (S_1) ในขณะที่โปรเซสดังกล่าวทำงานตามชุดคำสั่งของโปรแกรมนั้น ค่าประจำสถานะของโปรเซสได้เปลี่ยนเป็น [100, **0**, 100, 20] ซึ่งถือว่าโปรเซสดังกล่าวอยู่ในสถานะที่ 2 (S_2) จากเหตุการณ์ข้างต้นเกิดการเปลี่ยนแปลงค่าประจำสถานะ ถือว่าโปรเซสดังกล่าวได้เปลี่ยนสถานะจากสถานะที่ 1 เป็นสถานะที่ 2

สำหรับค่าประจำสถานะของโปรเซสจะถูกแทนด้วยสัญลักษณ์ uid, sid หรือ oid ขึ้นอยู่กับค่าของ UID หรือ EUID ในขณะนั้น ในขณะเดียวกันค่าของ GID และ EGID จะถูกแทนด้วยสัญลักษณ์ gid, sgid หรือ ogid เช่นเดียวกันขึ้นอยู่กับค่าของ GID หรือ EGID สัญลักษณ์ uid, sid, oid, gid, sgid และ ogid มีความหมายตามการนิยามดังนี้

- **uid** หรือ User ID ใช้สำหรับอ้างถึงค่า UID และ EUID ของโปรเซส โดยที่ค่าดังกล่าวถูกกำหนดโดยระบบปฏิบัติการ ซึ่งมีค่าเท่ากับ UID ของผู้ใช้ที่สั่งงานโปรเซสนั้น
- **sid** หรือ Special ID ใช้สำหรับอ้างถึงค่า UID และ EUID ของโปรเซส โดยที่ค่าดังกล่าวมีค่าเท่ากับค่า UID ของผู้ใช้ที่มีสิทธิพิเศษในระบบ ได้แก่ root, daemon, bin เป็นต้น
- **oid** หรือ Other ID ใช้สำหรับอ้างถึงค่า UID และ EUID ของโปรเซสที่นอกเหนือจากที่กล่าวมาแล้วข้างต้น
- **gid** หรือ Group ID ใช้สำหรับอ้างถึงค่า GID และ EGID ของโปรเซส โดยที่ค่าดังกล่าวถูกกำหนดโดยระบบปฏิบัติการ ซึ่งมีค่าเท่ากับ GID ของผู้ใช้ที่สั่งงานโปรเซส
- **sgid** หรือ Special group ID ใช้สำหรับอ้างถึงค่า GID และ EGID ของโปรเซส โดยที่ค่าดังกล่าวมีค่าเท่ากับค่า GID ของผู้ใช้ที่มีสิทธิพิเศษในระบบ ได้แก่ wheel, root, daemon, kmem และ sys เป็นต้น

- **ogid** หรือ Other group ID ใช้สำหรับอ้างถึงค่า GID และ EGID ของโปรเซสที่นอกเหนือจากที่กล่าวมาแล้วข้างต้น

โปรเซสที่กำลังทำงานจะต้องอยู่ในสถานะใดสถานะหนึ่งซึ่งขึ้นอยู่กับค่าประจำสถานะของโปรเซส Nuansri [Nuansri, 1999] ได้แบ่งสถานะของโปรเซสออกได้เป็น 6 สถานะคือ สถานะปกติ สถานะพิเศษ สถานะผู้ใช้สูงสุด สถานะกลุ่มระบบ สถานะผู้ใช้อื่น และสถานะสิ้นสุด โดยที่แต่ละสถานะมีความหมายดังต่อไปนี้

สถานะปกติ (Normal state)

โปรเซสที่อยู่ในสถานะปกติคือโปรเซสที่มีค่าประจำสถานะเป็น [uid, uid, gid, gid] โปรเซสในสถานะนี้ถูกจำกัดสิทธิ์ในการเข้าถึงทรัพยากรซึ่งสามารถดำเนินกิจกรรมต่างๆ ได้ตามสิทธิที่ผู้ใช้ได้รับ

สถานะพิเศษ (Special privileged state)

โปรเซสที่อยู่ในสถานะพิเศษคือโปรเซสที่มีค่าประจำสถานะค่าใดค่าหนึ่งเป็น suid หรือ sgid โปรเซสในสถานะนี้มีสิทธิ์ในการเข้าถึงทรัพยากรครึ่งหนึ่งของผู้ดูแลระบบซึ่งสามารถดำเนินกิจกรรมต่างๆ ได้เช่น การแก้ไขฐานข้อมูลผู้ใช้ การสร้างโปรแกรมแบบ setuid หรือการแก้ไขโปรแกรมระบบ เป็นต้น สำหรับสถานะนี้ได้แบ่งออกเป็น 4 สถานะย่อยได้แก่

- **setuid state** หมายถึงโปรเซสที่มีค่าประจำสถานะเป็น [uid, **suid**, gid, gid]
- **setreuid state** หมายถึงโปรเซสที่มีค่าประจำสถานะเป็น [**suid**, uid, gid, gid]
- **setgid state** หมายถึงโปรเซสที่มีค่าประจำสถานะเป็น [uid, uid, gid, **sgid**]
- **setregid state** หมายถึงโปรเซสที่มีค่าประจำสถานะเป็น [uid, uid, **sgid**, gid]

สถานะผู้ใช้สูงสุด (Super user state)

โปรเซสที่อยู่ในสถานะผู้ใช้สูงสุด คือโปรเซสที่มีค่า UID และ EUID เป็น suid พร้อมกัน ซึ่งแทนค่าประจำสถานะด้วย [**suid**, **suid**, gid, gid] โปรเซสในสถานะนี้มีสิทธิ์ในการเข้าถึงทรัพยากรเทียบเท่ากับของผู้ดูแลระบบ โปรเซสดังกล่าวสามารถดำเนินกิจกรรมทุกอย่างด้วยสิทธิ์ของผู้ดูและระบบ ซึ่งปกติแล้วโปรเซสปกติจะไม่เปลี่ยนสถานะมาสู่ในสถานะนี้

สถานะกลุ่มระบบ (System group state)

โปรเซสที่อยู่ในสถานะกลุ่มระบบ คือโปรเซสที่มีค่า GID และ EGID เป็น sgid พร้อมกัน ซึ่งแทนค่าประจำสถานะด้วย [uid, uid, **sgid, sgid**] โปรเซสในสถานะนี้มีสิทธิ์ในการเข้าถึงทรัพยากรเทียบเท่ากับของผู้ใช้ในกลุ่มระบบ ซึ่งปกติแล้วโปรเซสปกติจะไม่อยู่ในสถานะนี้

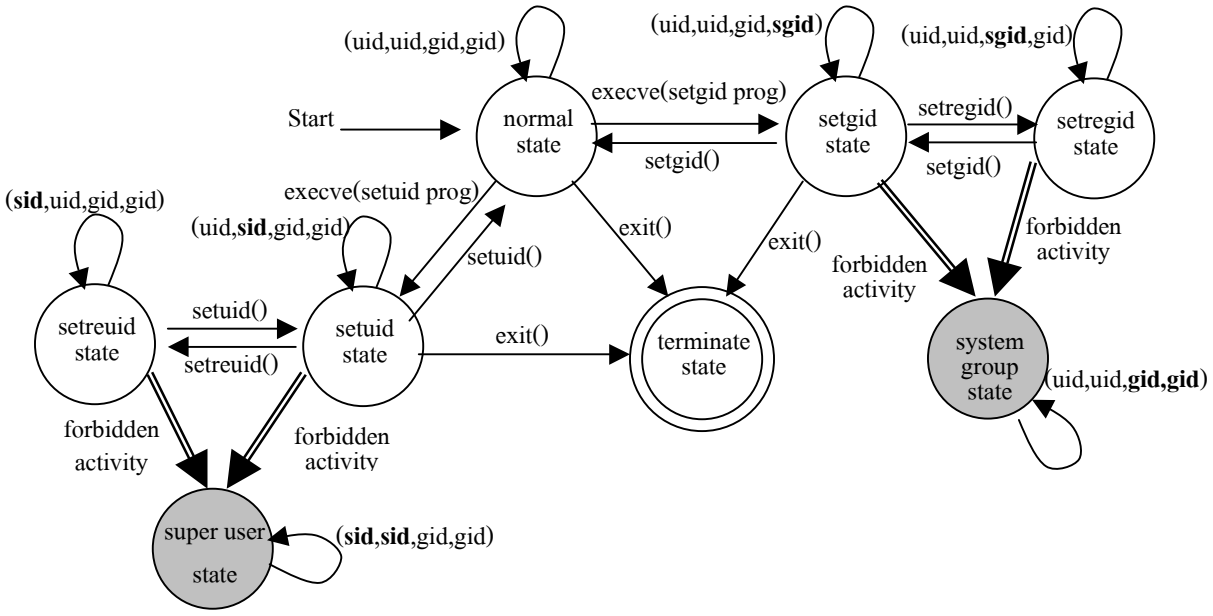
สถานะผู้ใช้อื่น (Other user state)

โปรเซสอยู่ในสถานะผู้ใช้อื่นแบ่งออกเป็น 2 กรณีคือ โปรเซสที่มีค่า UID และค่า EUID เป็น oid ซึ่งแทนค่าประจำสถานะด้วย [**oid, oid,** gid, gid] สำหรับอีกกรณีคือ โปรเซสที่มีค่า GIDและค่า EGID เป็น ogid ซึ่งแทนค่าประจำสถานะด้วย [uid**,** uid**, ogid**, **ogid**] สำหรับโปรเซสที่อยู่ในสถานะนี้มีสิทธิ์ในการเข้าใช้ทรัพยากรตามที่ oid และ ogid ได้รับเท่านั้นซึ่งไม่ก่อให้เกิดผลกระทบทางด้านความปลอดภัยของระบบ

สถานะสิ้นสุด (Terminate State)

โปรเซสจะอยู่ในสถานะสิ้นสุดเมื่อกระบวนการทำงานของคำสั่งนั้นกระทำจนสิ้นสุดกระบวนการและจบคำสั่งอย่างสมบูรณ์

ลำดับการทำงานของโปรเซสแต่ละโปรเซสสามารถเขียนแทนได้ด้วยแผนภาพการเปลี่ยนแปลงสถานะได้ สืบเนื่องจากการนิยามสถานะข้างต้น พบว่าโปรเซสทุกโปรเซสที่กำลังดำเนินงานจะต้องอยู่ในสถานะใดสถานะหนึ่ง เมื่อพิจารณาตามแผนภาพที่แสดงไว้ในภาพประกอบที่ 2.5



ภาพประกอบ 2.5 การเปลี่ยนแปลงสถานะของโปรเซสทั้ง 6 สถานะ [Nuansri, 1999]

จากภาพประกอบที่ 2.5 อธิบายได้ว่า เมื่อโปรเซสถูกสั่งงาน โปรเซสจะอยู่ในสถานะปกติและดำเนินกิจกรรมต่างๆ จนกระทั่งโปรเซสเรียกใช้ซิซเท็มคอล exit() เพื่อจบการทำงาน บางกรณีโปรเซสดังกล่าวได้สั่งงานโปรแกรมแบบ setuid ด้วยซิซเท็มคอล execve() ซึ่งเป็นผลให้ค่าประจำสถานะของโปรเซสเปลี่ยนแปลงเป็น [uid, **sid**, gid, gid] โปรเซสจึงเปลี่ยนแปลงสถานะเข้าสู่สถานะ setuid ซึ่งเป็นสถานะที่โปรเซสมีสิทธิ์ในการเข้าใช้ทรัพยากรเป็นครึ่งหนึ่งของผู้ดูแลระบบ ในขณะที่โปรเซสอยู่ในสถานะนี้ โปรเซสสามารถดำเนินกิจกรรมต่างๆ ด้วยสิทธิพิเศษ ถ้าหากโปรเซสนั้นเรียกใช้ซิซเท็มคอล exit() ในขณะที่อยู่ในสถานะนี้ โปรเซสก็จะหยุดการทำงาน แต่ถ้าหากโปรเซสนั้นเรียกใช้ซิซเท็มคอล setuid() แล้วโปรเซสเปลี่ยนสถานะไปยังสถานะปกติหรือสถานะ setreuid ได้ขึ้นอยู่กับชุดคำสั่งของโปรเซสนั้น แต่อย่างไรก็ตามโปรเซสดังกล่าวจะอยู่ในสถานะพิเศษเพียงชั่วคราวเท่านั้น เมื่อสำเร็จตามวัตถุประสงค์แล้ว โปรเซสดังกล่าวจะกลับเข้าสู่สถานะปกติ แต่ถ้าโปรเซสนั้นเปลี่ยนสถานะเข้าสู่สถานะผู้ใช้สูงสุดแล้วถือว่าโปรเซสดังกล่าวพยายามที่จะเปลี่ยนสิทธิ์ของโปรเซสให้เท่าเทียมกับผู้ดูแลระบบซึ่งไม่ควรจะเป็นและเป็นการกระทำที่ผิดจากการทำงานปกติของโปรเซสทั่วๆ ไป สำหรับวิทยานิพนธ์ถือว่าโปรเซสนั้นเป็นโปรเซสที่มีแนวโน้มในการบุกรุก

จากที่กล่าวมาข้างต้นจะเห็นได้อย่างชัดเจนว่าโปรเซสปกติสามารถเปลี่ยนสถานะเป็นสถานะพิเศษได้กรณีเดียวคือการสั่งงานโปรแกรมแบบ setuid อีกทั้งการเปลี่ยนสถานะของโปรเซสนั้นจะเปลี่ยนแปลงตามลำดับนั่นคือ เริ่มต้นที่สถานะปกติ หลังจากนั้นเข้าสู่สถานะ setuid แล้วอาจจะเปลี่ยนเป็นสถานะ setreuid หรือสถานะกลุ่มผู้ใช้ ดังนั้นเมื่อโปรเซสอยู่ในสถานะพิเศษจำเป็นที่จะต้องติดตามการทำงานของโปรเซส โดยพิจารณากิจกรรมต่างๆ ที่เกิดขึ้นว่าเป็นกิจกรรมที่ผิดปกติหรือไม่ Nuansri [Nuansri, 1999] จึงได้กำหนดกฎสนับสนุนออกมาเพื่อพิจารณากิจกรรมเหล่านั้น รายละเอียดของกฎสนับสนุนนั้นจะกล่าวในหัวข้อถัดไป

2.6.2 กฎสนับสนุน

เมื่อโปรเซสอยู่ในสถานะพิเศษโปรเซสเหล่านั้นจะถูกติดตามและพิจารณากิจกรรมเหล่านั้นด้วยกฎสนับสนุนเพื่อที่จะตัดสินว่ากิจกรรมที่กำลังทำอยู่นั้นว่าเข้าข่ายการบุกรุกระบบหรือไม่ กฎที่ใช้สนับสนุนระบบตรวจจับการบุกรุกที่จะกล่าวถึงนี้ มีทั้งหมด 6 ข้อ ได้แก่

กฎข้อที่ 0 อนุญาตให้โปรเซสเรียกใช้ซิซเท็มคอล setreuid() และซิซเท็มคอล setregid() เท่านั้นเพื่อเปลี่ยนค่า UID หรือค่า GID ถ้าหากโปรเซสเรียกใช้ซิซเท็มคอลตัวอื่นให้ถือว่าโปรเซสดังกล่าวเข้าข่ายการบุกรุก

กฎข้อที่ 1 เมื่อโปรเซสอยู่ในสถานะพิเศษจะไม่อนุญาตให้โปรเซสนั้นเรียกใช้ซิซเท็มคอล execve()

กฎข้อที่2 ในขณะที่โปรเซสอยู่ในสถานะพิเศษไม่อนุญาตให้โปรเซสดังกล่าวสร้างโปรแกรมแบบ setuid โดยที่งานประเภทนี้อนุญาตเฉพาะผู้ใช้ที่มีสิทธิสูงสุดเท่านั้น

กฎข้อที่ 3 ไม่อนุญาตให้โปรเซสที่อยู่ในสถานะพิเศษแก้ไขโปรแกรมระบบ

กฎข้อที่ 4 ผู้ใช้ที่มีสิทธิสูงสุดเท่านั้นที่มีสิทธิในการสร้างบัญชีผู้ใช้ใหม่ได้

กฎข้อที่ 5 ซิซเท็มคอลดังต่อไปนี้ mount(), unmount(), nfssvc(), quotactl(), reboot (), settimeofday() และ swapon() อนุญาตให้เรียกใช้โดยผู้ใช้ที่มีสิทธิสูงสุดเท่านั้น

สำหรับรายละเอียดของกฎสนับสนุนแต่ละข้อนั้นจะกล่าวถึงในบทถัดไป

2.7 บทสรุป

ผู้บุกรุกอาศัยจุดอ่อนของระบบเพื่อเป็นช่องทางสำหรับการบุกรุกระบบ ถ้าหากการบุกรุกประสบความสำเร็จย่อมก่อให้เกิดผลกระทบแก่ระบบ แม้ว่าภายในระบบนั้นได้ติดตั้งระบบตรวจจับการบุกรุกไว้ แต่การทำงานของระบบตรวจจับการบุกรุกนั้นอาจจะไม่สามารถป้องกันได้ 100% เนื่องจากระบบตรวจจับการบุกรุกทำงานในระดับของโปรเซสของระบบย่อมถูกทำลายได้โปรเซสอื่นๆ ดังนั้นวิทยานิพนธ์ชุดนี้จึงเสนอแนวทางสำหรับการเพิ่มสมรรถนะแก่ระบบปฏิบัติการยูนิกซ์ด้วยการแก้ไขซิซเท็มคอลของระบบปฏิบัติการ โดยซิซเท็มคอลที่ต้องแก้ไขนั้นได้จากการวิเคราะห์กฎสนับสนุนการตรวจจับการบุกรุก ซึ่งในบทถัดไปจะกล่าวถึงการวิเคราะห์กฎสนับสนุนเพื่อให้ได้มาซึ่งซิซเท็มคอลที่ต้องแก้ไขและทดสอบผลของการวิเคราะห์โดยการพัฒนาโปรแกรมตรวจจับการบุกรุกหลังจากนั้นหาอัลกอริทึมสำหรับการแก้ไขระบบปฏิบัติการต่อไป