ชุดโปรแกรมรักษาความปลอดภัยเครือข่าย

NETWORK SECURITY SUITE

นาย เกียรติศักดิ์ เหมกรณ์

นาย จักรพันธ์ ขวัญเริงใจ

นางสาว เอกภัทรา ขวัญเพชร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2549

ชุดโปรแกรมรักษาความปลอดภัยเครือข่าย

NETWORK SECURITY SUITE

โดย

นาย เกียรติศักดิ์ เหมกรณ์

นาย จักรพันธ์ ขวัญเริงใจ

นางสาว เอกภัทรา ขวัญเพชร

อาจารย์ที่ปรึกษา

อาจารย์ อัครเดช วัชระภูพงษ์

ผศ. ธนา หงษ์สุวรรณ

อาจารย์ ธนัญชัย ตรีภาค

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดโปรแกรมรักษาความปลอดภัยเครือข่าย

NETWORK SECURITY SUITE

ผู้จัดทำ

1. นาย เกียรติศักดิ์	เหมกรณ์	รหัสประจำตัว	46010064
2. นาย จักรพันธ์	ขวัญเริงใจ	รหัสประจำตัว	46010100
3. นางสาว เอกภัทรา	ขวัญเพชร	รหัสประจำตัว	46010993

_____อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

_____อาจารย์ที่ปรึกษา

(ผศ. ธนา หงษ์สุวรรณ)

_____อาจารย์ที่ปรึกษา

(อาจารย์ ธนัญชัย ตรีภาค)

ชุดโปรแกรมรักษาความปลอดภัยเครือข่าย

นาย เกียรติศักดิ์ เหมกรณ์ 46010064
นาย จักรพันธ์ ขวัญเริงใจ 46010100
นางสาว เอกภัทรา ขวัญเพชร 46010993
อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ผศ. ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษาร่วม
อาจารย์ ธนัญชัย ตรีภาค อาจารย์ที่ปรึกษาร่วม
ปีการศึกษา 2549

บทคัดย่อ

ในปัจจุบันการรักษาความปลอดภัยระบบเครือข่ายเป็นเรื่องที่สำคัญลำดับแรกสุดที่ต้องคำนึงถึง โดยเฉพาะอย่างยิ่งสำหรับองค์กรที่ต้องการความปลอดภัยสำหรับข้อมูล เครื่องมือแรก ๆ ที่เป็นที่นิยมใช้ในการป้องกันการบุกรุกเครือข่ายคงหนีไม่พ้น "ไฟร์วอล" ซึ่งถือได้ว่าเป็นอุปกรณ์รักษาความปลอดภัยเครือข่ายที่มีความจำเป็นอันดับต้น ๆ โดยโครงงานนี้จะมีการนำไฟร์วอลประเภทมาทำงานร่วมกันเพื่อให้การรักษาความปลอดภัยเครือข่ายเป็นไปได้อย่างมีประสิทธิภาพ ได้แก่ ไฟร์วอลส่วนบุคคล เกตเวย์ไฟร์วอลซึ่งทำงานอยู่บนเกตเวย์ และพร็อกซี่ไฟร์วอล ซึ่งคือซีเคียวรีเวิร์สเว็บพร็อกซี่ แต่เนื่องด้วยข้อจำกัดของไฟร์วอลที่จะจำแนกแพ็กเก็ตได้ตามกฎที่มีอยู่เท่านั้น ไม่สามารถจำแนกแพ็กเก็ตที่ไม่ตรงตามกฎใด ๆ ได้ อีกทั้งการบุกรุกในปัจจุบันมีความสลับซับซ้อนและมีกลวิธีใหม่ ๆ เกิดขึ้นตลอดเวลา จึงต้องมีการนำระบบจำแนกและบันทึกพฤติกรรมผู้บุกรุกหรือที่เรียกว่า "ฮันนี่พอต" เข้ามาช่วยในการป้องกันการโจมตีจากผู้บุกรุกอีกทางหนึ่ง โดยจะช่วยจำแนกแพ็กเก็ตที่ไม่ตรงกับกฎของไฟร์วอลว่าเข้าข่ายบุกรุกหรือไม่ หลังจากนั้นระบบจะทำการคัดแยกผู้ใช้งานที่มีพฤติกรรมที่น่าสงสัยออกจากระบบจริง เข้าสู่ระบบเสมือนเพื่อเฝ้าดูและศึกษาพฤติกรรมต่อไป

NETWORK SECURITY SUITE

Mr. Kiattisak Hamkorn	46010064
Mr. Jakkapan Khwanreungjai	46010100
Ms. Ekpatthra Khwanpetch	46010993
Mr. Akkradach Watcharapupong	Advisor
Asst.Prof. Thana Hongsuwan	Co-Advisor
Mr. Thananchai Treepak	Co-Advisor

Academic Year 2006

ABSTRACT

At the present time, network security has became the first priority which has to concern, especially for the organizations that their information sensitive. So the organizations have to keep it safe all the time. The most popular security system is "Firewall" as it is one of the must-have security systems. This project combines many types of firewall to work together in order to get the highest efficiency such as a personal firewall with an intrusion detection system, a gateway firewall and a secure reverse web proxy which is a proxy firewall.

However firewall has some exceptions which it works on the knowledge bases the rules that it is set. These make it cannot classify some connections which do not match with the rules. Moreover, nowadays the intruders have many new techniques and tool to intrude the organizations which perhaps firewall cannot protect or prevent. So it's is necessary to use Honeypot to solve this problems. Honeypot will detect the packet that does not match with the rules of firewall and classified the intruders from the users. After that it send the intruders to the virtual system, at this we can learn and observe about their techniques which will help us to approve our organization

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี อันเนื่องมาจากการได้รับโอกาส คำแนะนำ คำสั่งสอน และคำปรึกษาจากท่านอาจารย์อย่างดีเสมอมา โดยเฉพาะอย่างยิ่ง จากท่านอาจารย์อัครเดช วัชระภูพงษ์ ท่านอาจารย์ ธนัญชัย ดรีภาค และผู้ช่วยศาสตราจารย์ ธนา หงษ์สุวรรณ ซึ่งต้องขอขอบพระคุณท่านอาจารย์ทั้งสามเป็นอย่างสูง รวมถึงท่านอาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้ประสิทธิประสาทความรู้ในทุก ๆ แขนงเป็นอย่างดีเสมอมา

ขอขอบคุณห้องวิจัยและพัฒนาการรักษาความปลอดภัยข้อมูล (ISAG) ภาควิชาวิศวกรรมคอมพิวเตอร์ ที่เป็นแหล่งประสิทธิประสาทความรู้ความเข้าใจ ตลอดจนสนับสนุนเครื่องมือและข้อมูลเพื่อใช้ในการทำวิจัย อีกทั้งเป็นสถานที่ที่เต็มไปด้วยความอบอุ่นจากท่านอาจารย์ เพื่อน ๆ พี่ ๆ ที่คอยให้คำปรึกษาในการทำวิจัยเป็นอย่างดีเสมอมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกๆเรื่อง ทำให้ข้าพเจ้าสามารถทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี คุณค่าและประโยชน์อันพึงมาจากปริญญานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นาย เกียรดิศักดิ์ เหมกรณ์
นาย จักรพันธ์ ขวัญเริงใจ
นางสาวเอกภัทรา ขวัญเพชร

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

บทที่ 1
บทนำ

1.1 บทนำ

ในปัจจุบันการใช้งานคอมพิวเตอร์ผ่านระบบเครือข่ายมีบทบาทสำคัญต่อการดำเนินกิจกรรมต่าง ๆ เป็นอย่างมาก แต่ในการเชื่อมต่อผ่านทางเครือข่ายนั้นยังมีความเสี่ยงต่อการถูกบุกรุกทั้งจากภายในและภายนอกองค์กร จึงจำเป็นต้องมีการตรวจสอบป้องกันระบบเครือข่ายและเครื่องคอมพิวเตอร์อย่างเหมาะสม ซึ่งสิ่งที่สามารถช่วยลดความเสี่ยงจากการถูกบุกรุกนี้ได้อย่างหนึ่งคือไฟร์วอล (Firewall) โดยไฟร์วอลนั้นจะทำหน้าที่ป้องกันอันตรายต่าง ๆ จากภายนอกที่เข้ามายังเครือข่ายของเรา โดยไฟร์วอลแบ่งเป็นหลายประเภท แต่ละประเภทจะมีหน้าที่ในการทำงานแตกต่างกัน ดังนั้นในการป้องกันระบบเครือข่ายขนาดใหญ่จึงต้องมีการนำไฟร์วอลประเภทต่าง ๆ มาประกอบกันเพื่อการทำงานที่ได้ประสิทธิภาพมากขึ้น

แต่ในโลกแห่งความเป็นจริง ผู้บุกรุกมีกลวิธีและเครื่องมือใหม่ ๆ ในการบุกรุกอยู่เสมอ อีกทั้งความสามารถของไฟร์วอลในการจำกัดแพ็คเก็ตที่เข้ามาในระบบนั้นเป็นแบบ Rules-base ซึ่งจะอาศัยกฎที่ได้ตั้งไว้เท่านั้น รวมทั้งยังไม่สามารถป้องกันการบุกรุกที่มาจากภายในได้ (Internal Attack) ดังนั้น การบุกรุกรูปแบบใหม่ ๆ ที่ไม่มีในกฎหรือไม่สามารถจำแนกได้อย่างแน่ชัดจากกฎที่มีอยู่ของไฟร์วอลว่าเป็นผู้บุกรุก จึงควรถูกเปลี่ยนเส้นทางไปยังระบบจำลอง ที่เรียกว่า ฮันนี่พอต (Honeypot) เพื่อทำการตรวจสอบอีกครั้งว่าเข้าข่ายเป็นผู้บุกรุกหรือไม่ ถ้าเป็นผู้บุกรุก ระบบจะล่อหลอกให้เข้าไปในเครื่องกับดักเพื่อเฝ้าดูการกระทำ ความพยายามต่าง ๆ หรือ พฤติกรรมที่ผู้บุกรุกกระทำต่อระบบ เพื่อใช้เป็นกรณีศึกษาและสามารถนำไปใช้ในการกำหนดกฎให้กับไฟร์วอลได้ต่อไปในอนาคต โดยที่การจับตาดูพฤติกรรมของผู้บุกรุกจะกระทำโดยไม่ให้ผู้บุกรุกรู้ตัวว่าถูกจับตามองอยู่และไม่ส่งผลเสียหายใดๆต่อระบบจริง

โครงงานนี้เป็นโครงงานต่อเนื่องจากปี 2548 คือ ISAG Firewall Suite 2548 และ ISAG Honeypot Suite 2548 โดยพยายามพัฒนารวบรวมและปรับปรุงความสามารถให้แก่ระบบไฟร์วอลและระบบฮันนี่พอต โดยจะรวมเอาส่วนการทำงานต่าง ๆ ของทั้ง 2 โครงงาน อันได้แก่ ไฟร์วอลส่วนบุคคล (FireAlarm), เกตเวย์ไฟร์วอล (FireScreen), รีเวิร์สเว็บพร็อกซี (FireBreak), ส่วนควบคุมระบบไฟร์วอล (FireStation), ส่วนจำแนกของระบบฮันนี่พอต (Honeywall) และระบบหลอก (Cage) ให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ เพื่อให้เป็นชุดโปรแกรมรักษาความปลอดภัยเครือข่ายที่สมบูรณ์แบบ สามารถจำแนกการสื่อสารได้ว่าปรกติหรือผิดปรกติได้อย่างแม่นยำและเก็บพฤติกรรมหรือกระทั่งล่อหลอกผู้บุกรุกเข้าสู่ระบบหลอกได้อย่างแนบเนียน

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาการทำงาน, โครงสร้างและคุณสมบัติของไฟร์วอลสำหรับระบบเครือข่าย

1.2.2 เพื่อศึกษาการทำงานและโครงสร้างของระบบฮันนี่พอต เพื่อจำแนกผู้บุกรุกออกจากผู้ใช้งานจริง และบันทึกพฤติกรรมของผู้บุกรุกที่กระทำต่อระบบ

1.2.3 พัฒนาการทำงานของไฟร์วอลและฮันนี่พอตให้สามารถทำงานร่วมกันได้ เพื่อเพิ่มประสิทธิภาพในการรักษาความปลอดภัยให้กับระบบเครือข่าย

1.2.4 เพื่อสร้างต้นแบบชุดรักษาความปลอดภัยเครือข่าย

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้รับความรู้ความเข้าใจเกี่ยวกับการทำงานของระบบไฟร์วอลประเภทต่างๆ

1.3.2 ได้รับความรู้ความเข้าใจเกี่ยวกับการทำงานของระบบฮันนี่พอต

1.3.3 ได้รับความรู้ความเข้าใจเกี่ยวกับการจำแนกผู้บุกรุกออกจากผู้ใช้ทั่วไป

1.3.4 ได้รับความรู้ความเข้าใจเกี่ยวกับวิธีการทำงานของระบบตรวจจับผู้บุกรุก

1.3.5 ได้รับความรู้ความเข้าใจเกี่ยวกับวิธีการบุกรุกและการป้องกันการบุกรุกแบบต่าง ๆ

1.3.6 ได้รับความรู้ความเข้าใจในการติดต่อใช้งานระบบแอ็คทีฟไดเร็คทอรี

1.4 ขอบเขตของโครงงาน

1.4.1 โปรแกรมไฟร์สเตชั่นเป็นส่วนกำหนดกฎของโปรแกรมไฟร์อลาร์ม , ไฟร์สกรีน

1.4.2 โปรแกรมไฟร์สกรีนสามารถกรองแพ็คเก็ตที่ผ่านเข้าออกเครือข่ายได้ตามกฎที่รับมาจากส่วนไฟร์สเตชั่น รวมทั้งสามารถเปลี่ยนเส้นทางแพ็คเก็ตที่น่าสงสัยให้ไปถูกตรวจสอบโดยใช้ระบบตรวจจับผู้บุกรุก เพื่อจำแนกผู้บุกรุกออกจากผู้ใช้ระบบจริงอีกทีหนึ่ง

1.4.3 โปรแกรมไฟร์อลาร์มสามารถใช้งานภายใต้สิทธิของผู้ใช้งานได้

1.4.4 โปรแกรมไฟร์อลาร์มสามารถทำงานแบบวินโดวส์เซอร์วิส (Windows Service) ซึ่งสามารถป้องกันเครื่องของผู้ใช้ได้แม้ว่าผู้ใช้คนนั้นยังไม่ได้ล็อกอินเข้าระบบก็ตาม

1.4.5 ระบบตรวจจับผู้บุกรุกมีความสามารถในการตรวจจับการบุกรุกแบบใหม่ ๆ ได้

1.4.6 สามารถจัดเก็บล็อกไฟล์ที่ผิดปรกติจากส่วนของไฟร์สกรีน, ไฟร์แทรป และไฟร์เบรกไว้ที่ฐานข้อมูลของส่วนกลางและนำมาแสดงผลได้

1.4.7 สามารถทำการ Autoupdate Signature ของสนอร์ดอินไลน์เพื่อเพิ่มความสามารถในการจำแนกผู้บุกรุกให้มีความทันสมัยอยู่เสมอ

1.5 ขั้นตอนการดำเนินงาน

1.5.1 ศึกษาความรู้พื้นฐาน การทำงานส่วนต่าง ๆ ของระบบไฟร์วอล และ ระบบฮันนี่พอต

1.5.2 ศึกษาหาวิธีการและแนวทางในการพัฒนาให้ส่วนกรองแพ็คเก็ตของระบบไฟร์วอล (Firescreen) สามารถทำงานร่วมกับส่วนจำแนกผู้บุกรุกของระบบฮันนี่พอตได้

1.5.3 ศึกษารูปแบบวิธีการบุกรุก และการจำแนกผู้บุกรุกออกจากผู้ใช้จริง

1.5.4 ศึกษาวิธีการทำงานของแอคทีฟไดเรคทอรี่

1.5.5 ศึกษาวิธีการทำงานของไอพีเทเบิล และระบบตรวจจับผู้บุกรุก (Snort_inline) ที่จะนำมาใช้ในการพัฒนาส่วนตรวจสอบผู้บุกรุก

1.5.6 ทดลองการใช้ระบบไฟร์วอลให้สามารถกรองแพ็คเก็ตตามกฎที่ตั้งจากส่วนควบคุมไฟร์สเตชั่นและสามารถดักจับแพ็คเก็ตที่ไม่เข้าตามกฎใด ๆ ให้ถูกจำแนกด้วยระบบตรวจจับผู้บุกรุกอีกครั้งหนึ่ง

1.5.7 พัฒนาส่วนติดต่อกับผู้ใช้ (User Interface) ให้มีความสามารถมากขึ้น

1.5.8 ทำรายงานและสรุปผล

1.6 เนื้อหาของรายงาน

รายงานฉบับนี้มีทั้งหมด 12 บทดังนี้

- บทที่ 1 กล่าวถึงวัตถุประสงค์ ประโยชน์และขอบเขตของโครงงาน
- บทที่ 2 กล่าวถึงทฤษฎีและหลักการที่ใช้ในการพัฒนาระบบชุดโปรแกรมรักษาความปลอดภัยเครือข่าย
- บทที่ 3 กล่าวถึงการออกแบบพัฒนาโครงงานซึ่งจะอธิบายโปรแกรมในส่วนต่าง ๆ ของระบบว่ามีอะไรบ้าง สามารถทำอะไรได้บ้าง
- บทที่ 4 กล่าวถึงหลักการทำงาน การพัฒนาและเทคโนโลยีที่ใช้ในการพัฒนาระบบไฟร์สเตชั่น (FireStation)
- บทที่ 5 กล่าวถึงหลักการทำงาน การพัฒนาและเทคโนโลยีที่ใช้ในการพัฒนาระบบไฟร์สกรีน (FireScreen)
- บทที่ 6 กล่าวถึงหลักการทำงาน การพัฒนาและเทคโนโลยีที่ใช้ในการพัฒนาระบบไฟร์อลาร์ม (FireAlarm)
- บทที่ 7 กล่าวถึงหลักการทำงาน การพัฒนาและเทคโนโลยีที่ใช้ในการพัฒนาระบบไฟร์แทรป (FireTrap)
- บทที่ 8 กล่าวถึงหลักการทำงาน การพัฒนาและเทคโนโลยีที่ใช้ในการพัฒนาระบบไฟร์เบรก (FireBreak)

- บทที่ 9 กล่าวถึงหลักการทำงาน การพัฒนาและเทคโนโลยีที่ใช้ในการพัฒนาระบบล็อกมอนิเตอร์ (Log Monitor)
- บทที่ 10 กล่าวถึงโปรแกรม Tartarus Management (TM) ซึ่งเป็นโปรแกรม Front-end ในการควบคุมการทำงานของระบบจำแนกผู้บุกรุก
- บทที่ 11 กล่าวถึงการทดสอบการทำงานรวมของโครงงานชุดโปรแกรมรักษาความปลอดภัยเครือข่าย
- บทที่ 12 บทวิจารณ์และสรุป

บทที่ 10
การพัฒนาโปรแกรม Tartarus Management

10.1 แนวคิด

โปรแกรม Tartarus Management (TM) เป็นโปรแกรม Front-end ที่สร้างขึ้นเพื่อแก้ปัญหาความยุ่งยากในการดูแลและจัดการระบบจำแนกผู้บุกรุกที่อยู่ในส่วนไฟร์สกรีน การจัดการกับเครื่องกับดักของระบบไฟร์แทรป รวมถึงการแสดงล็อกไฟล์จากระบบตรวจจับผู้บุกรุก และล็อกไฟล์จากการบันทึกพฤติกรรมของผู้บุกรุกในเครื่องกับดัก

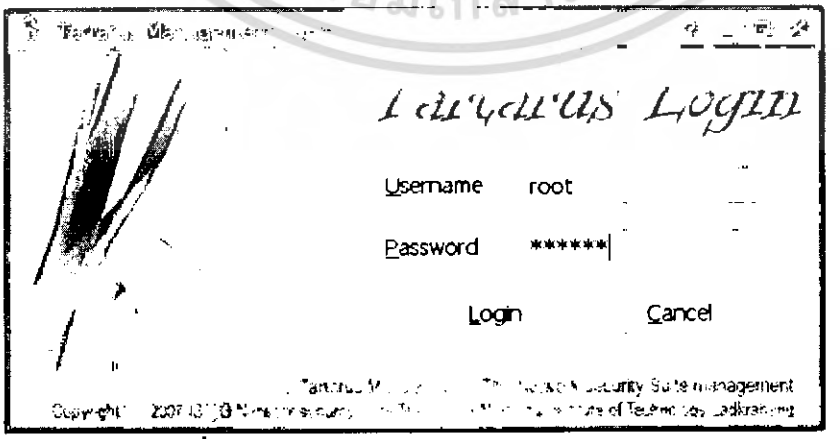
10.2 ขอบเขตและความสามารถ

โปรแกรม Tartarus Management (TM) สามารถควบคุม และกำหนดค่าการทำงานของเครื่องมือต่าง ๆ ได้ เช่น สามารถดูสถานะการทำงานของระบบจำแนกผู้บุกรุกส่วนไฟร์สกรีน , สั่งเริ่มหรือยกเลิกการทำงานของระบบ , ดูสถานะการทำงานของเครื่องกับดักในระบบไฟร์แทรป , สร้าง แก้ไข สั่งแช่แข็งหรือลบเครื่องกับดักได้ รวมทั้งยังสามารถเลือกใช้ สร้างหรืออัพเดตกฎของสนอร์ตอินไลน์ได้ พร้อมทั้งแสดงข้อมูลต่าง ๆ ที่อยู่ในดาต้าเบสเซิร์ฟเวอร์ (LogServer) ได้

10.3 การทำงานของโปรแกรม

โปรแกรม Tartarus Management มีรายละเอียดดังนี้

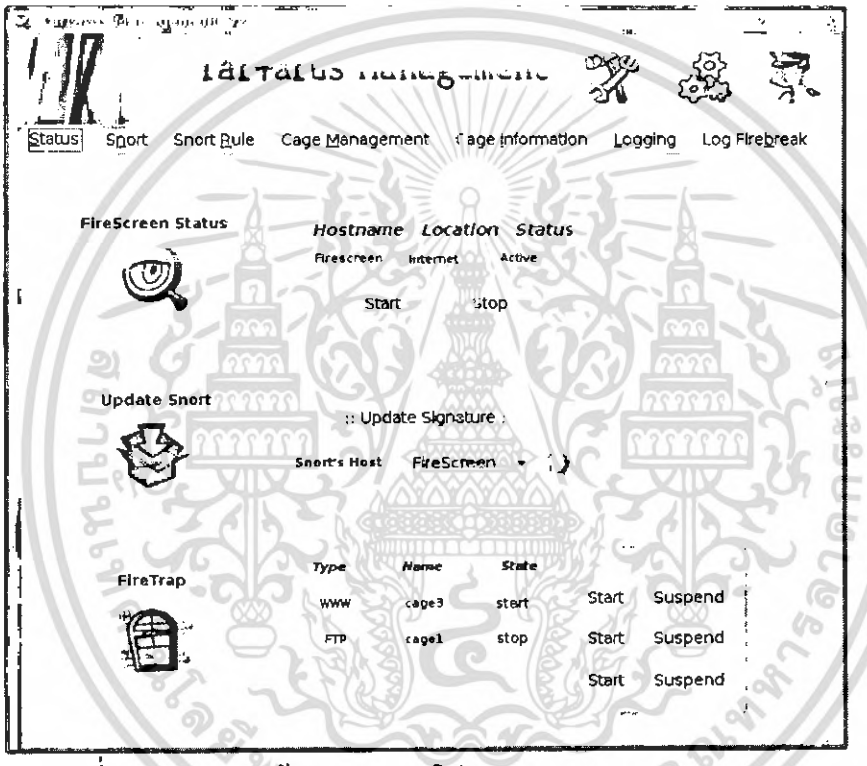
- **หน้าต่าง Login** เพื่อพิสูจน์ตัวตนก่อนใช้โปรแกรม



รูปที่ 10.1 แสดงหน้าจอล็อกอินเข้าโปรแกรม

- **หน้า Status**

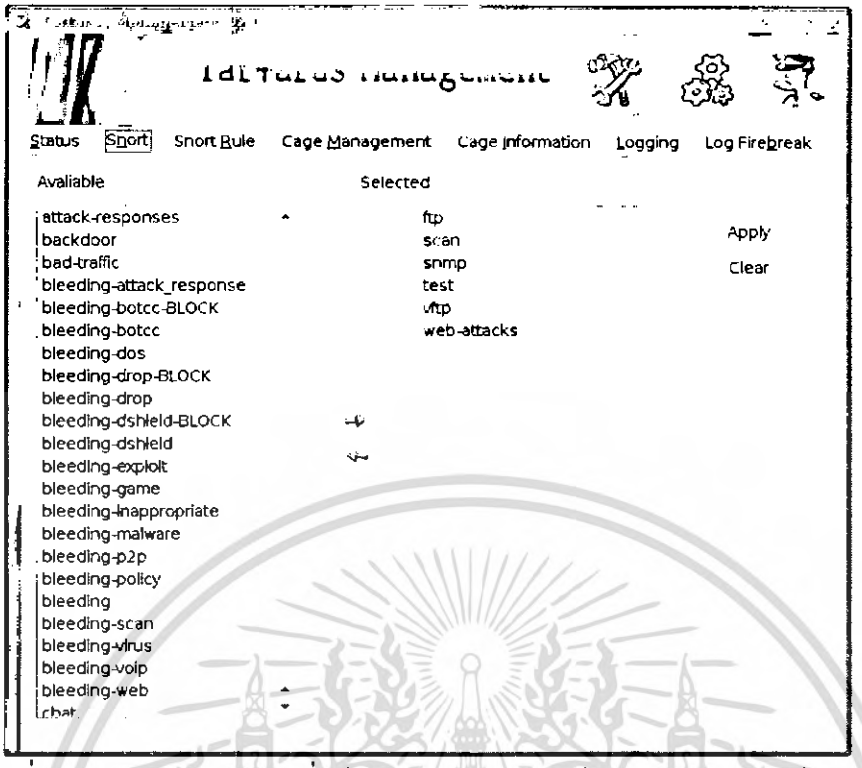
เป็นหน้าที่บอกถึงสถานะของเครื่องไฟร์สกรีนและเครื่องกับดักไฟร์แทรป ว่าทำงานอยู่หรือไม่ โดยสำหรับเครื่องกับดักนั้นจะแสดงถึงชื่อและประเภทของเครื่องกับดักด้วย อีกทั้งยังสามารถสั่งเริ่มหรือหยุดการทำงานของโปรแกรมสนอร์ตอินไลน์ในเครื่องไฟร์สกรีน สามารถทำการอัพเดต Signature ของสนอร์ตอินไลน์ที่อยู่ในไฟร์สกรีนและไฟร์เบรกได้ และยังสามารถสั่งเริ่มหรือหยุดการทำงานของเครื่องกับดักได้โดยการกดปุ่ม Start หรือ Suspend ซึ่งจะมีลักษณะดังในรูปถัดไป



รูปที่ 10.2 แสดงหน้า Status ของโปรแกรม Tartarus Management

- **หน้า Snort**

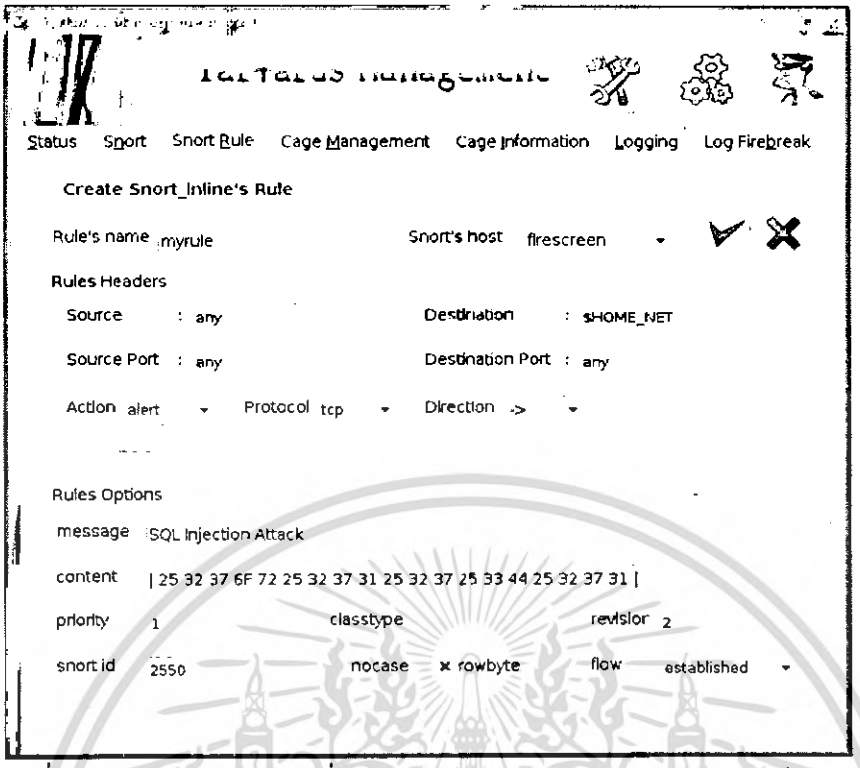
เป็นส่วนที่ใช้ในการเพิ่มหรือลดกฎของโปรแกรมสนอร์ตอินไลน์ที่เครื่องไฟร์สกรีนรวมถึงการอัพเดต Signature ของสนอร์ตอินไลน์โดยวิธีการเพิ่มกฎที่จะใช้จำแนกผู้บุกรุกสามารถทำได้โดยเลือกจากรายชื่อที่อยู่ด้านในกรอบAvaliable (กรอบด้านซ้าย) แล้วคลิ๊กที่ปุ่มลูกศร ⟶ แล้วกดปุ่ม Apply



รูปที่ 10.3 แสดงรูปในส่วนที่ใช้ในการกำหนดกฎ ให้กับระบบตรวจจับผู้บุกรุก

- **หน้า Create Snort Rule**

เป็นส่วนที่ให้ผู้ใช้สามารถทำการสร้างกฎสำหรับสนอร์ตอินไลน์ที่ใช้ในระบบได้ โดยผู้ใช้สามารถกำหนดส่วนเฮดเดอร์และออปชั่นต่าง ๆ ให้กับกฎของสนอร์ตอินไลน์ตามความต้องการ และเมื่อทำการกรอกข้อมูลเป็นที่เรียบร้อย ให้กดปุ่ม ✓ จากนั้นโปรแกรมจะทำการอัพเดตกฎที่สร้างใหม่ไปยังเครื่องไฟร์สกรีนหรือไฟร์เบรกในระบบ เพื่อใช้งานต่อไป



รูปที่ 10.4 แสดงรูปในส่วนที่ใช้ในการสร้างกฎให้กับระบบตรวจจับผู้บุกรุก

- **หน้า Cage Management**

เป็นส่วนที่ใช้ในการจัดการกับเครื่องกับดักทั้งในส่วนของการสร้างเครื่องกับดักใหม่หรือลบเครื่องกับดักเก่า และกำหนดเงื่อนไขในการตรวจสอบสภาพของเครื่องกับดัก ซึ่งจะช่วยให้ผู้ดูแลระบบไม่จำเป็นต้องเข้าไปตรวจสอบด้วยตัวเอง เพราะโปรแกรมจะทำการตรวจสอบให้โดยอัตโนมัติอยู่ตลอดเวลาในการใช้งาน โดยสามารถสร้างเครื่องกับดักใหม่ได้จากเครื่องกับดักต้นแบบที่เก็บไว้ใน Cage Path (กำหนดพาธ ได้ในหน้าการ Configuration) ส่วนพาธที่เก็บเครื่องกับดักใหม่ที่สร้างขึ้นสามารถกำหนดได้ในช่อง Path ของหน้าต่างนี้

เงื่อนไขในการตรวจสอบสภาพเครื่องกับดักที่ผู้ใช้สามารถเลือกให้โปรแกรมตรวจสอบได้แก่ จำนวนผู้ใช้ปัจจุบัน, จำนวนผู้ใช้ในกรุ๊ปรูท, เจ้าของไฟล์ /etc/passwd, password root ว่ามีการเปลี่ยนแปลงหรือไม่, และโหมดของไฟล์ /etc/passwd

<u>วิธีการสร้างไฟล์ config ของเครื่องกับดัก</u>

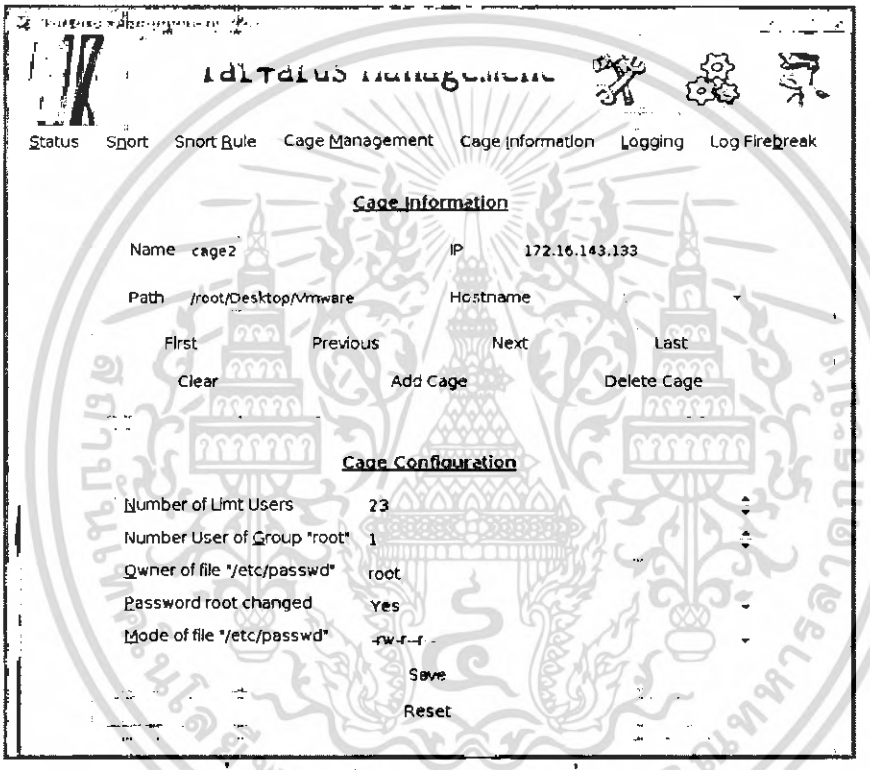
สามารถทำได้โดยเลือกเครื่องกับดักที่ต้องการโดยกดปุ่ม Next หรือ Previous แล้วเลือกเงื่อนไขที่ต้องการให้ตรวจสอบ จากนั้นกดปุ่ม save โปรแกรมก็จะทำการสร้างไฟล์ cage.conf ให้ โดยไฟล์นี้จะเก็บอยู่ในไดเรกทอรี่เดียวกับที่เก็บตัวอิมเมจของเครื่องกับดัก

<u>การสร้างเครื่องกับดักใหม่</u>

การสร้างเครื่องกับดักใหม่ สามารถทำได้โดยในขั้นแรกให้กดปุ่ม Clear จากนั้นให้พิมพ์ข้อมูลลงในช่องต่างๆ แล้วกดปุ่ม Add Cage

<u>การลบเครื่องกับดัก</u>

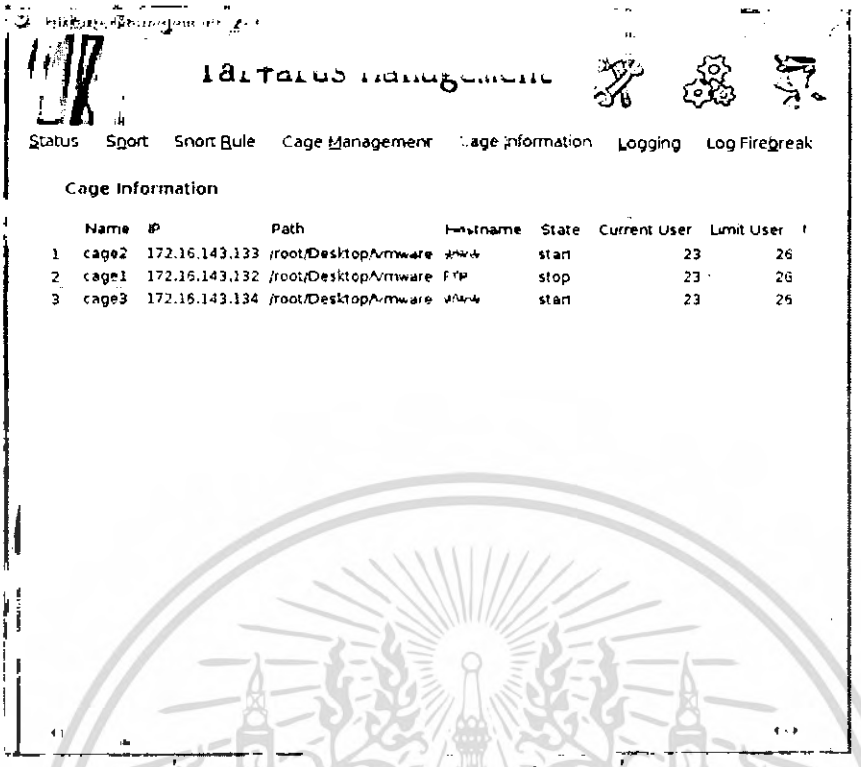
การลบเครื่องกับดักสามารถทำได้โดยการเลือกเครื่องกับดักที่ต้องการลบโดยกดปุ่ม next หรือปุ่ม Previous แล้วกดปุ่ม Delete Cage



รูปที่ 10.5 หน้าต่างส่วนจัดการเครื่องกับดัก

- หน้า **Cage Information**

ส่วน Cage Information เป็นส่วนที่ใช้แสดงข้อมูลของเครื่องกับดักที่เก็บอยู่ในฐานข้อมูลในเครื่องแม่ข่ายฐานข้อมูล (Log Server) ซึ่งจะแสดงถึงชื่อ, หมายเลขไอพีแอดเดรส, สถานะ, พาธที่เก็บเครื่องกับดัก และข้อกำหนดที่ใช้ตรวจสอบเครื่องกับดักนั้นๆ โดยในการติดต่อกับฐานข้อมูลภายในเครื่องแม่ข่ายฐานข้อมูลนั้นจะกระทำผ่านช่องทาง SSH ซึ่งสถานะของเครื่องกับดักที่แสดงในหน้าต่างนี้จะสอดคล้องกับสถานะของเครื่องกับดักจากหน้าต่าง Status



รูปที่ 10.6 หน้าต่างของส่วนแสดงข้อมูลเครื่องกับดัก

- **หน้า Logging**

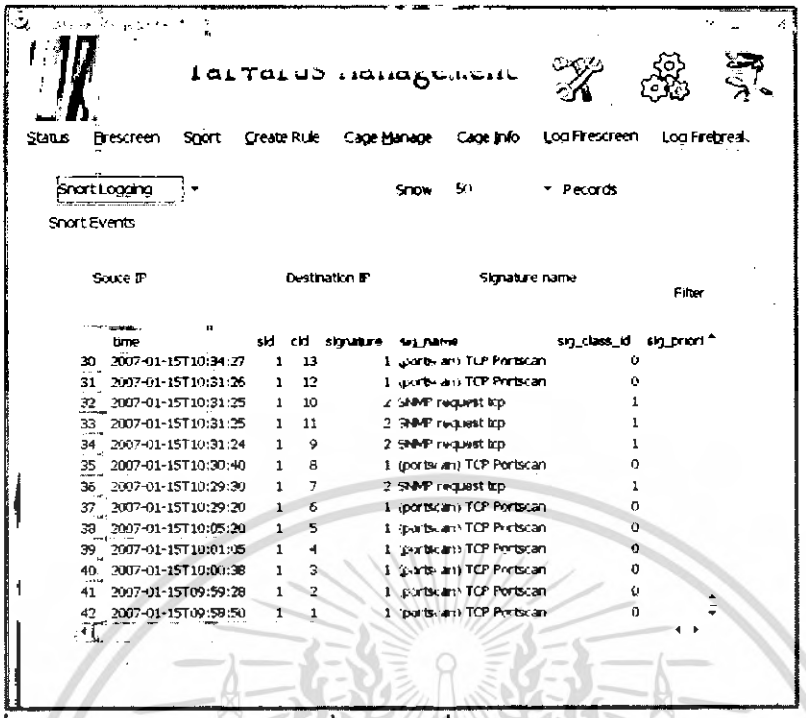
เป็นส่วนที่ใช้แสดงหน้าที่ใช้ในการดูข้อมูลล็อกไฟล์ ที่เก็บอยู่ในฐานข้อมูลภายในเครื่อง Logserver ซึ่งจะประกอบด้วยข้อมูลที่ได้จากโปรแกรมสนอร์ตอินไลน์, โปรแกรมเซมเฮน และโปรแกรมเซเบค ซึ่งทั้ง 3 โปรแกรมนี้จะทำการเก็บบันทึกที่แตกต่างกันดังนี้

Sebek จะทำการบันทึกการกระทำของผู้บุกรุก เช่น keystroke, ข้อมูลการนำเข้า หรือส่งออก, ไฟล์ที่ได้นำเข้าหรือส่งออก

Samhain จะทำการบันทึกการเปลี่ยนแปลงของไฟล์ต่างๆ ในเครื่องกับดัก ซึ่งจะทำให้ทราบผลจากคำสั่งของผู้บุกรุกว่าไปมีผลต่อไฟล์ใด หรือทำการแก้ไข หรือลบไฟล์ใด

Snort inline จะทำการบันทึกข้อมูลของชิ้นข้อมูลที่มีการตรวจพบว่าเป็นผู้บุกรุกจากเครื่องไฟร์สกรีน

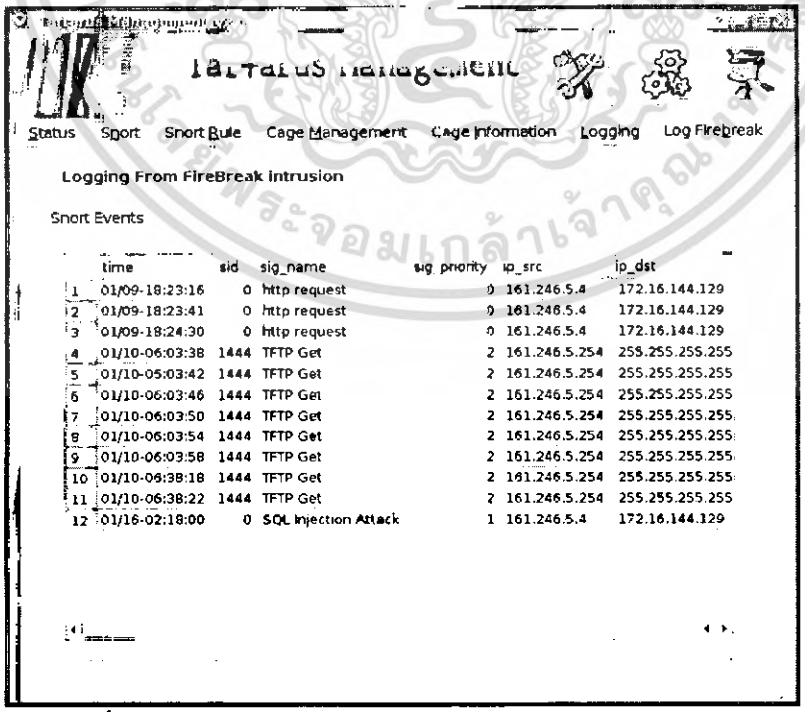
ผู้ใช้สามารถเลือกดูล็อกไฟล์จากแต่ละโปรแกรมโดยทำการเลือกที่ ComboBox ด้านบน แล้วโปรแกรมจะทำการดึงข้อมูลจากเครื่อง Logserver ออกมาแสดง นอกจากนั้นผู้ใช้สามารถค้นหาเฉพาะข้อมูลที่ต้องการโดยใส่ข้อมูลลงในช่องด้านบน แล้วกดปุ่ม Filter



รูปที่ 10.7 หน้าต่างแสดงข้อมูลที่เก็บในเครื่องคาต้าเบสเซิร์ฟเวอร์ (LogServer)

- **หน้า Log FireBreak**

แสดงล็อกไฟล์เกี่ยวกับการบุกรุกของผู้บุกรุกที่เข้ามาในระบบไฟร์เบรก โดยหลังจากที่โปรแกรมสนอร์ตอินไลน์ในไฟร์เบรกตรวจพบรูปแบบการบุกรุกที่เกิดขึ้นและทำการตัดการเชื่อมต่อนั้น ๆ โปรแกรมสนอร์ตอินไลน์จะทำการส่งล็อกไฟล์เหตุการณ์ที่เกิดขึ้นมาแสดงผล

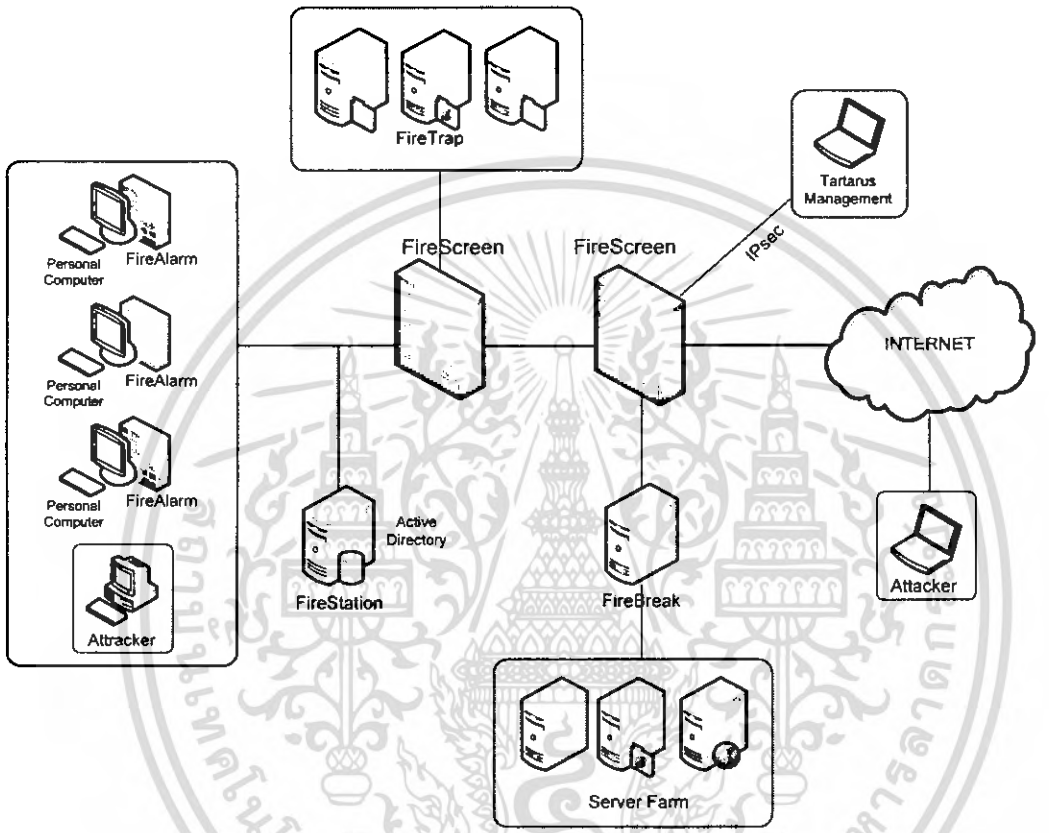


รูปที่ 10.8 หน้าต่างแสดงข้อมูลล็อกไฟล์จากระบบไฟร์เบรก

บทที่ 11

การทดลองและผลการทดลอง

11.1 โครงสร้างระบบ



รูปที่ 11.1 แสดงโครงสร้างระบบที่ใช้ในการทดลอง

11.2 ขั้นตอนการทดสอบชุดโปรแกรม

ในการทดลองจะแบ่งการทดลองออกเป็น

- การทดสอบการทำงานของไฟร์สเตชั่นในการกำหนดกฎให้กับไฟร์อลาร์มและไฟร์สกรีน
- การทดสอบการทำงานของไฟร์อลาร์มในการทำงานเป็นวินโดวส์เซอร์วิส
- การทดสอบการทำงานของไฟร์สกรีนในการทำหน้าที่เป็นเกตเวย์ไฟร์วอล ซึ่งกรองแพ็คเก็ตตามกฎที่รับมาจากไฟร์สเตชั่น
- การทดสอบการทำงานของไฟร์สกรีนในการทำหน้าที่จำแนกผู้บุกรุกออกจากผู้ใช้งานจริงและสร้างเส้นทางให้ผู้บุกรุกไปยังไฟร์แทรป
- การทดสอบการทำงานของการอัพเดต Signature ของสนอร์ตอินไลน์

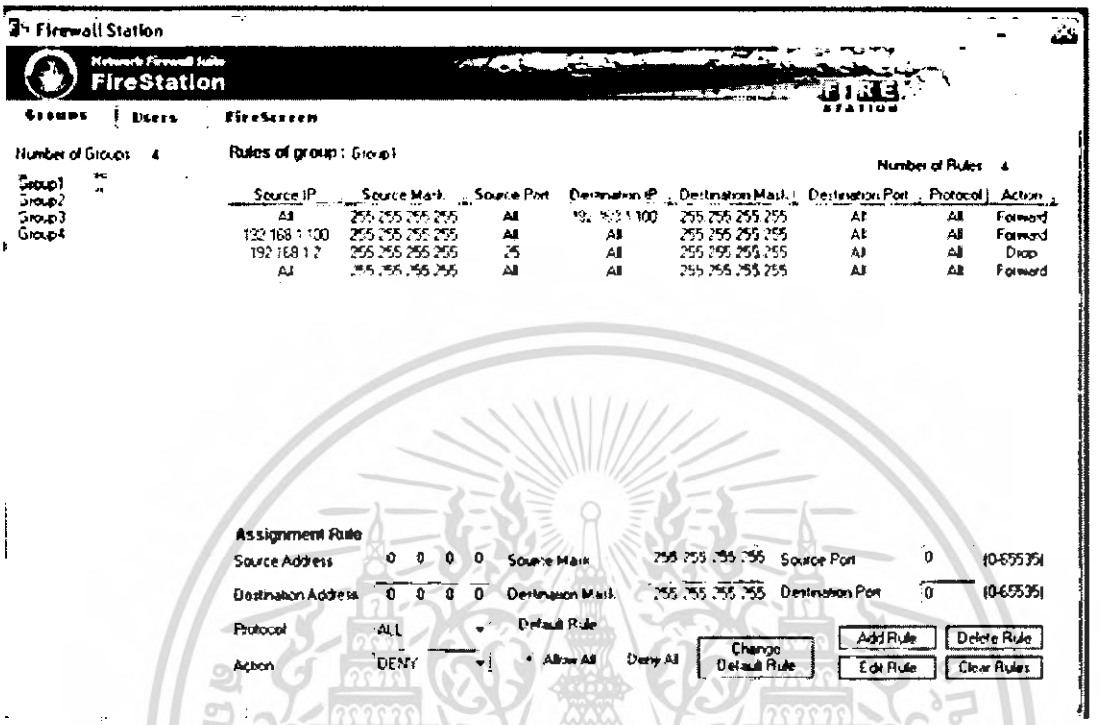
รายละเอียดของเครื่องในระบบ

บทบาท	ชื่อเครื่อง	IP ที่ eth0	IP ที่ eth1	IP ที่ eth2
Attacker (Linux)	Isag40	161.246.5.40	-	-
Attacker (Windows)	Attacker	192.168.1.70	-	-
FireScreen (Linux)	firescreen1	161.246.5.48	172.16.144.135	172.16.142.1
FireScreen (Linux)	firescreen2	172.16.142.2	172.16.143.135	192.168.3.254
FireBreak (Linux)	firebreak	172.16.144.129	-	-
FireStation (Windows)	firestation	192.168.1.100	-	-
FireAlarm (Windows)	firealarm	192.168.1.2	-	-
LogServer (Linux)	logserver	172.16.144.128	-	-
FireTrap (FTP) ตัวที่ 1	Cage1	172.16.143.132	-	-
FireTrap (www) ตัวที่ 2	Cage2	172.16.143.134	-	-
FireTrap (www) ตัวที่ 3	Cage3	172.16.143.133	-	-
Webserver (Linux)	WWW	172.16.144.131	-	-

ตารางที่ 11.1 ตารางกำหนดหมายเลขไอพีประจำเครื่องแต่ละเครื่อง

11.2.1 ทดสอบการทำงานของไฟร์สเตชั่น(FireStation)

เมื่อโปรแกรมไฟร์สเตชั่นเริ่มต้นการทำงานแล้วจะมีลักษณะดังต่อไปนี้



รูปที่ 11.2 โปรแกรมไฟร์สเตชั่น

โปรแกรมไฟร์สเตชั่นจะแบ่งออกเป็น 3 ส่วนด้วยกันคือ

- ส่วนการกำหนดกฎให้กับกลุ่มผู้ใช้
- ส่วนการกำหนดกฎให้กับผู้ใช้เป็นรายบุคคล
- ส่วนการกำหนดกฎให้กับไฟร์วอลบนเกต์เวย์หรือไฟร์สกรีน(FireScreen)

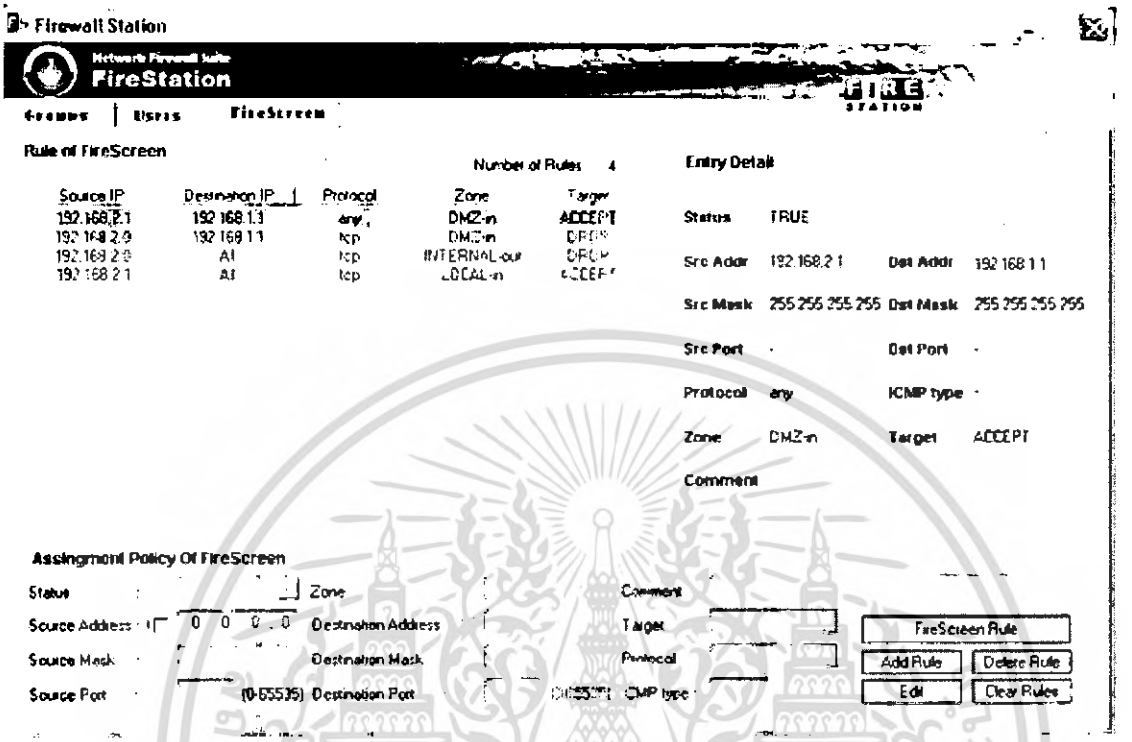
การทำงานของโปรแกรมนี้จะเป็นการกำหนดกฎให้กับไฟร์วอลส่วนต่างๆ ภายในระบบ สามารถทำการเพิ่ม แก้ไข และลบกฎได้โดยการนำค่าที่กำหนดไปใส่ในแอตทริบิวของอ็อปเจ็คนั้นๆ โดยกฎพื้นฐานจะเป็น Deny All คือไม่อนุญาตให้การติดต่ออื่นๆผ่านเข้ามาได้

ตัวอย่างการกำหนดกฎของไฟร์วอลให้กับผู้ใช้แบบรายบุคคลเพื่อให้สามารถขอใช้บริการเว็บไซต์จากเครื่องแม่ข่ายภายในองค์กรได้

Source IP	Source Mask	Source Port	Destination IP	Destination Mask	Destination Port	Protocol	Action
192.168.1.2	255.255.255.255	All	All	255.255.255.255	80	TCP	Drop

รูปที่ 11.3 แสดงการกำหนดกฎตัวอย่าง

จากรูป 11.4 เป็นการกำหนดกฎให้กับผู้ใช้รายบุคคลสำหรับป้องกันเครื่อง 192.168.1.2 ขอใช้บริการเว็บไซต์



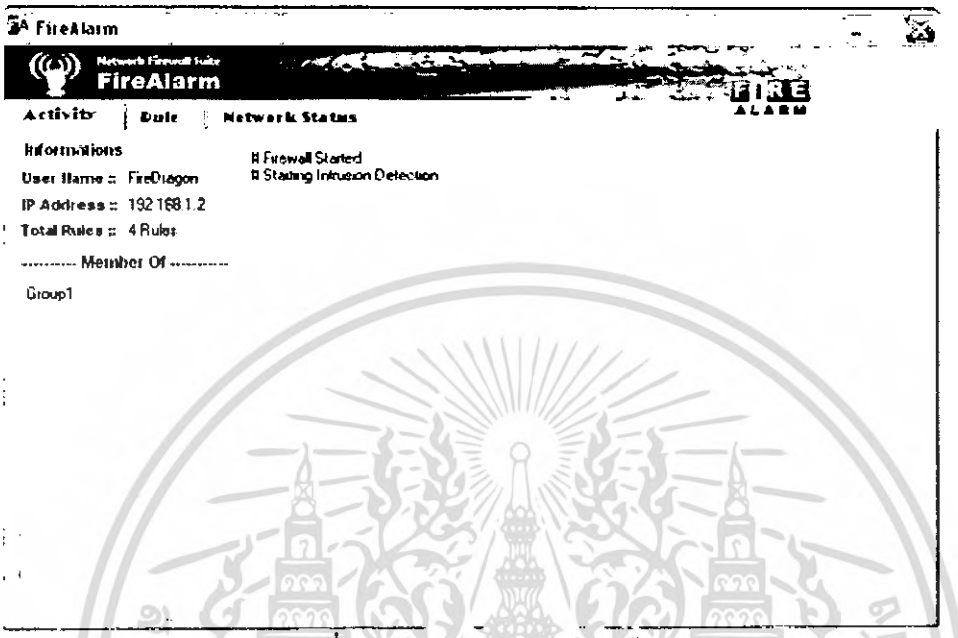
รูปที่ 11.4 โปรแกรมไฟร์สเตชั่นส่วนของการกำหนดกฎให้กับไฟร์สกรีน

กฎที่กำหนดให้กับแต่ละส่วนจะถูกเก็บไว้ในแอ็คทีฟไดเร็คทอรีในตำแหน่งที่ถูกต้องเพื่อรอไคลเอนต์ที่เป็นเจ้าของกฎนั้นมานำไปใช้ต่อไป ในส่วนของไฟร์สกรีนนั้นกฎที่ใส่ไว้ทั้งหมดนี้จะถูกนำไปเก็บในแอ็คทีฟไดเร็คทอรีในแอตทริบิวต์ firewallLog ของ DN คือ "cn=firescreen, cn=computers, dc=hephaestus, dc=com" และหลังจากกฎถูกเปลี่ยนแปลงแก้ไขอย่างไรก็ตามจะไปเปลี่ยนค่าในแอตทริบิวต์ update ของ DN เดียวกันนี้เป็น "YES" ด้วย

เช่นเดียวกัน กฎสำหรับไฟร์สกรีนจะถูกตีความหมายจากบนลงล่างเช่นกัน สำหรับตัวอย่างข้างต้น 2 กฎแรกสำหรับแพ็กเก็ตที่จะผ่านเข้าไปยังโซน DMZ กฎที่ 3 สำหรับแพ็กเก็ตที่ผ่านออกมาจากเครือข่ายภายใน และสุดท้ายจะเป็นสำหรับแพ็กเก็ตที่เข้ามายังเครื่องเกตเวย์เอง

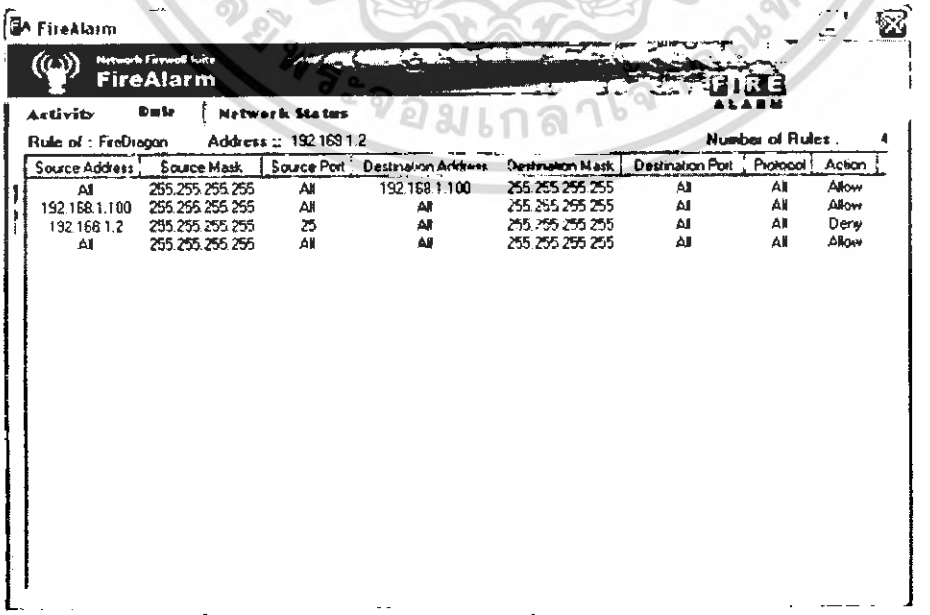
11.2.2 ทดสอบการทำงานของไฟร์อลาร์ม (FireAlarm)

เมื่อผู้ใช้ลงทะเบียนเข้าใช้ระบบปฏิบัติการ โปรแกรมเพอร์ซันนอลไฟร์วอลจะเริ่มทำงานทันที และจะมีลักษณะดังต่อไปนี้



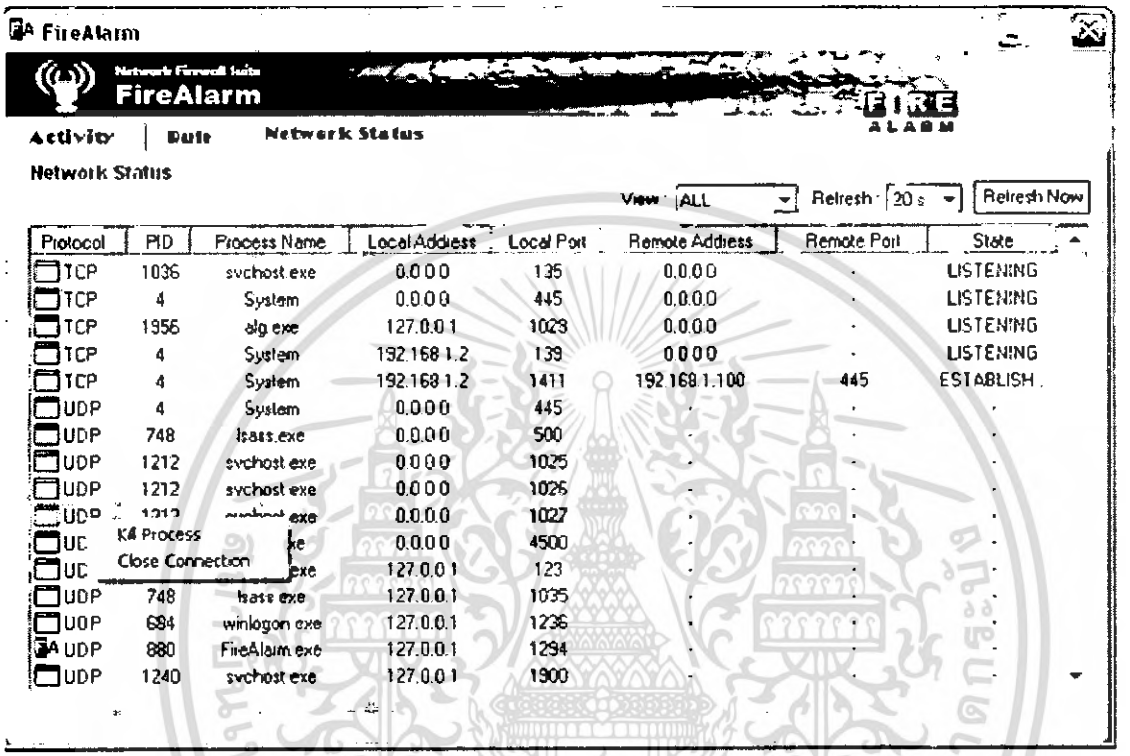
รูปที่ 11.5 โปรแกรมไฟร์อลาร์ม

เมื่อโปรแกรมเริ่มทำงานจะทำการรับเอากฎการฟิลเตอร์ของกลุ่มและผู้ใช้ที่ทำการล็อกอิน (Login) ตามที่กำหนดไว้ที่ส่วนกลางมาเป็นกฎการฟิลเตอร์ของตัวโปรแกรมไฟร์อลาร์มก่อนจะเริ่มทำงานด้วยการฟิลเตอร์ต่างๆ ตามกฎการฟิลเตอร์ที่รับมาจากส่วนกลาง และจะเริ่มการทำงานของส่วนตรวจจับผู้บุกรุกด้วยทันที โดยกฎที่รับมาจะแสดงดังรูป



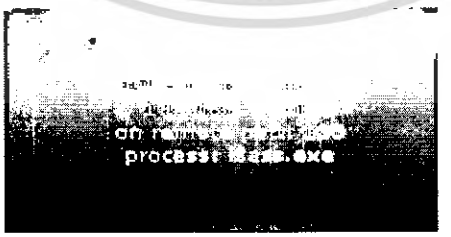
รูปที่ 11.6 แสดงกฎการฟิลเตอร์ที่รับมาจากเซิร์ฟเวอร์

โปรแกรมไฟร์อลาร์มในส่วนของสถานะทางเครือข่าย (Network Status) จะแสดงรายการของโปรแกรมที่ใช้งานเครือข่ายอยู่ในขณะนั้นโดยจะอัพเดตรายการตามเวลาที่ได้ตั้งเอาไว้ และในส่วนนี้สามารถเลือกจบโปรแกรมการทำงาน (Kill Process) หรือปิดการเชื่อมต่อของโพรเซส (Close Connection) ใดๆได้ดังแสดงในรูป 11.7



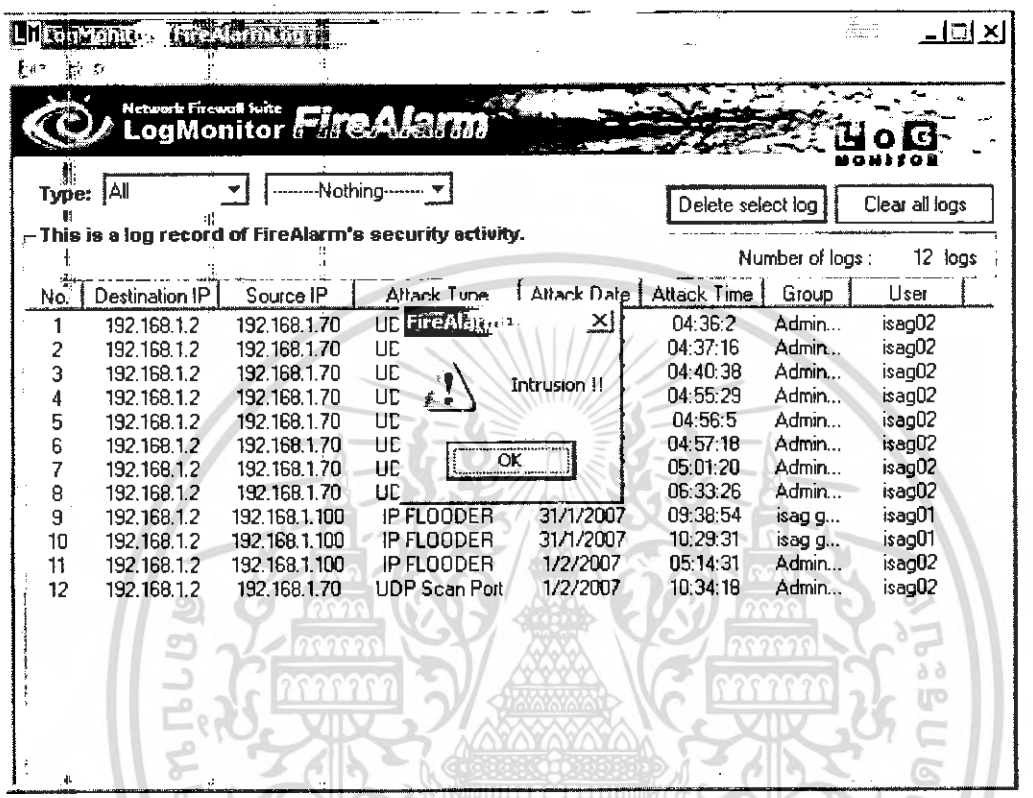
รูปที่ 11.7 แสดงสถานะของโพรเซสที่ใช้งานเครือข่าย

เมื่อมีโพรเซสใดที่เชื่อมต่อเครือข่ายจะป๊อปอัพ (Popup) ขึ้นมาบอกทิศทางและไอพีแอดเดรสที่โพรเซสนั้นเชื่อมต่อ ดังแสดงในรูป



รูปที่ 11.8 แสดงป๊อปอัพ (Popup) แสดงโพรเซสที่ใช้งานเครือข่าย

และเมื่อมีการโจมตีหรือมีการบุกรุกทางเครือข่ายเกิดขึ้น โปรแกรมจะแสดงรายละเอียดดังรูป11.9 และจะทำการส่งรายละเอียดต่างๆของการโจมตีหรือการบุกรุกนั้นๆไปเก็บไว้ที่แอตทริบิวต์ firewallLo g ของกลุ่มของผู้ใช้ในแอ็คทีฟไดเร็คทอรี (Active Directory)



รูปที่ 11.9 การแจ้งเตือนการบุกรุกหรือถูกโจมตี

11.2.3 ทดสอบการทำงานของไฟร์สกรีน (Firescreen)

หลังจากเครื่องเกตเวย์ซึ่งติดตั้งโปรแกรมไฟร์สกรีนไว้เริ่มต้นทำงานขึ้นมา ขั้นตอนแรกโปรแกรมจะทำการใส่กฎเริ่มต้นและกฎเดิมเข้ากับเกตเวย์เสียก่อน โดยในที่นี้จะเป็นการทำงานครั้งแรกของโปรแกรมไฟร์สกรีนจะไม่มีกฎเก่าก่อนอยู่ เหลือเพียงแค่กฎเริ่มต้นที่ตั้งไว้แล้วเท่านั้น เราสามารถใช้คำสั่ง iptables -L --line-number เพื่อตรวจสอบดูกฎในไอพีเทเบิลว่าถูกต้องตามต้องการหรือไม่

```
firescreen:~# iptables -L --line-number
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN
 limit: avg 5/sec burst 5
3    DROP       all  --  BASE-ADDRESS.MCAST.NET/4  anywhere
4    DROP       all  --  240.0.0.0/5          anywhere
5    DROP       all  --  127.0.0.0/8          anywhere
6    DROP       all  --  0.0.0.0/8            anywhere
7    DROP       all  --  169.254.0.0/16       anywhere
8    DROP       all  --  anywhere             anywhere            state INVALID
9    ACCEPT     all  --  firestation.hephaestus.com  firescreen.hephaestus.com state RELATE
D,ESTABLISHED

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  192.168.2.0/24       firestation.hephaestus.com tcp dpt:686
2    ACCEPT     tcp  --  firestation.hephaestus.com  192.168.2.0/24      state RELATED,ESTA
BLISHED
3    ACCEPT     all  --  192.168.1.0/24       anywhere            state RELATED,ESTABLISHED

4    ACCEPT     tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN
 limit: avg 5/sec burst 5
5    DROP       all  --  BASE-ADDRESS.MCAST.NET/4  anywhere
6    DROP       all  --  240.0.0.0/5          anywhere
7    DROP       all  --  127.0.0.0/8          anywhere
8    DROP       all  --  169.254.0.0/16       anywhere
9    DROP       all  --  anywhere             BASE-ADDRESS.MCAST.NET/4
10   DROP       all  --  anywhere             240.0.0.0/5
11   DROP       all  --  anywhere             127.0.0.0/8
12   DROP       all  --  anywhere             169.254.0.0/16
13   DROP       all  --  anywhere             255.255.255.255
14   DROP       all  --  anywhere             anywhere            state INVALID

Chain OUTPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    DROP       all  --  anywhere             255.255.255.255
3    DROP       all  --  anywhere             BASE-ADDRESS.MCAST.NET/4
4    ACCEPT     tcp  --  firescreen.hephaestus.com  firestation.hephaestus.com tcp dpt:ldap


Chain LOG-CHAIN (0 references)
num  target     prot opt source               destination
1    LOG        all  --  anywhere             anywhere            limit: avg 1/sec burst 5
LOG level info prefix `FIRESCREEN '
2    DROP       all  --  anywhere             anywhere
```

รูปที่ 11.10 แสดงกฎเริ่มต้นของไฟร์สกรีน

จะพบว่ากฎพื้นฐานที่กำหนดไว้ในเชลล์สคริปต์ init.sh ถูกกำหนดให้กับไอพีเทเบิลเรียบร้อยแล้ว รวมถึงการสร้างเชน LOG-CHAIN ซึ่งใช้สำหรับเก็บล็อกรายละเอียดของแพ็กเก็ตที่จะละทิ้งไป (DROP) อนึ่งสำหรับแพ็กเก็ตที่ละทิ้งไปในกฎพื้นฐานนี้จะไม่มีการเก็บล็อกไว้แต่อย่างใด เฉพาะส่วนที่ละทิ้งไปเนื่องจากกฎที่กำหนดไว้ที่ไฟร์สเตชั่นเท่านั้นที่จะมีการเก็บรายละเอียดลงล็อกไฟล์

ขั้นตอนต่อไปจะเป็นการนำกฎจากไฟร์สเตชั่นที่ได้กำหนดไว้ก่อนหน้ามาใช้ ไฟร์สกรีนจะคอยตรวจสอบไปยังไฟร์สเตชั่นเป็นระยะเพื่อดูว่ามีการเปลี่ยนแปลงแก้ไขหรือไม่ กรณีนี้เพิ่งได้ทำการเพิ่มกฎเข้าไปใหม่ ดังนั้นไฟร์สกรีนจะทำการนำกฎทั้งหมดมาจากไฟร์สเตชั่นต่อท้ายกฎพื้นฐานก่อนหน้านี้ เราสามารถตรวจสอบดูด้วยคำสั่ง iptables -L --line-number เพื่อดูว่าไอพีเทเบิลได้รับกฎตามที่กำหนดไว้โดยไฟร์สเตชั่นถูกต้องหรือไม่

```
Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  192.168.2.0/24       firestation.hephaestus.com tcp dpt:686
2    ACCEPT     tcp  --  firestation.hephaestus.com  192.168.2.0/24      state RELATED,ESTAB
LISHED
3    ACCEPT     all  --  192.168.1.0/24       anywhere            state RELATED,ESTABLISHED
4    ACCEPT     tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN
limit: avg 5/sec burst 5
5    DROP       all  --  BASE-ADDRESS.MCAST.NET/4  anywhere
6    DROP       all  --  240.0.0.0/5          anywhere
7    DROP       all  --  127.0.0.0/8          anywhere
8    DROP       all  --  169.254.0.0/16       anywhere
9    DROP       all  --  anywhere             BASE-ADDRESS.MCAST.NET/4
10   DROP       all  --  anywhere             240.0.0.0/5
11   DROP       all  --  anywhere             127.0.0.0/8
12   DROP       all  --  anywhere             169.254.0.0/16
13   DROP       all  --  anywhere             255.255.255.255
14   DROP       all  --  anywhere             anywhere            state INVALID
15   ACCEPT     all  --  hep01.hephaestus.com  www.hephaestus.com
16   LOG-CHAIN  tcp  --  192.168.2.0/24       www.hephaestus.com  tcp dpt:ssh
17   LOG-CHAIN  tcp  --  192.168.2.0/24       anywhere            tcp dpt:1863
```

รูปที่ 11.11 กฎจากไฟร์สเตชั่นที่เพิ่มเข้าไปในเชนฟอร์เวิร์ด

```
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     all  --  anywhere             anywhere
2    ACCEPT     tcp  --  anywhere             anywhere            tcp flags:SYN,RST,ACK/SYN
limit: avg 5/sec burst 5
3    DROP       all  --  BASE-ADDRESS.MCAST.NET/4  anywhere
4    DROP       all  --  240.0.0.0/5          anywhere
5    DROP       all  --  127.0.0.0/8          anywhere
6    DROP       all  --  0.0.0.0/8            anywhere
7    DROP       all  --  169.254.0.0/16       anywhere
8    DROP       all  --  anywhere             anywhere            state INVALID
9    ACCEPT     all  --  firestation.hephaestus.com  firescreen.hephaestus.com state RELATED
,ESTABLISHED
10   ACCEPT     tcp  --  hep01.hephaestus.com  anywhere           tcp dpt:ssh
```

รูปที่ 11.12 กฎจากไฟร์สเตชั่นที่เพิ่มเข้าไปในเชนอินพุท

จะสังเกตเห็นว่าในเชนฟอร์เวิร์ดจะพบว่ามีกฎเพิ่มมาอีก 3 กฎเพิ่มขึ้นมาจากกฎพื้นฐานจาก init.sh คือกฎที่ 15, 16 และ 17 ซึ่งเป็น 3 กฎแรกที่กำหนดไว้ในไฟร์สเตชั่น

ส่วนในเชนอินพุทก็มีเพิ่มขึ้น 1 กฎเช่นกัน คือกฎสุดท้ายที่ตั้งไว้ที่ไฟร์สเตชั่น อย่างไรก็ตาม ในส่วนของเชนเอาท์พุทและเชน LOG-CHAIN นั้นจะไม่ถูกทำการเปลี่ยนแปลงแก้ไข สำหรับส่วนของล็อกไฟล์ หากมีแพ็กเก็ตที่ตรงกับกฎที่ระบุทาร์เก็ตเป็นเชน LOG-CHAIN (ก่อนที่จะถูกละทิ้งไปที่ตอนท้ายของเชน LOG-CHAIN) จะถูกเก็บรายละเอียดลงยังล็อกไฟล์ซึ่งเก็บไว้ที่ /var/log/firescreen.log และจะมีรายละเอียดของล็อกแสดงขึ้นบนหน้าจอด้วย

```
FIRESCREEN IN=eth2 OUT=eth1 SRC=192.168.2.7 DST=192.168.1.1 LEN=64 TOS=0x00 PREC
=0x00 TTL=63 ID=30287 DF PROTO=TCP SPT=2524 DPT=22 WINDOW=1460 RES=0x00 ACK URGP
=0
FIRESCREEN IN=eth2 OUT=eth1 SRC=192.168.2.7 DST=192.168.1.1 LEN=64 TOS=0x00 PREC
=0x00 TTL=63 ID=30289 DF PROTO=TCP SPT=2524 DPT=22 WINDOW=1460 RES=0x00 ACK URGP
=0
FIRESCREEN IN=eth2 OUT=eth1 SRC=192.168.2.7 DST=192.168.1.1 LEN=64 TOS=0x00 PREC
=0x00 TTL=63 ID=30291 DF PROTO=TCP SPT=2524 DPT=22 WINDOW=1460 RES=0x00 ACK URGP
=0
FIRESCREEN IN=eth2 OUT=eth1 SRC=192.168.2.7 DST=192.168.1.1 LEN=64 TOS=0x00 PREC
=0x00 TTL=63 ID=30293 DF PROTO=TCP SPT=2524 DPT=22 WINDOW=1460 RES=0x00 ACK URGP
=0
```

รูปที่ 11.13 ล็อกที่แสดงขึ้นที่หน้าจอของไฟร์สกรีน

ล็อกดังกล่าวเป็นล็อกจากเครื่องภายในเครือข่ายภายในซึ่งมีไอพีแอดเดรสเป็น 192.168.2.7 พยายามใช้ซีเคียวเชลล์ (ssh) เข้าไปยังเครื่องเว็บเซิร์ฟเวอร์ซึ่งมีไอพีแอดเดรสเป็น 192.168.1.1 แต่ถูกปฏิเสธโดยตัวไฟร์สกรีนจากกฎที่ 16 ในเชนฟอร์เวิร์ด (รูปที่ 11.10)

ล็อกไฟล์จริงๆนั้นเก็บไว้ที่ /var/log/firescreen.log จะมีรูปแบบเช่นเดียวกันกับล็อกที่แสดงบนหน้าจอดังรูปที่ 11.14 เพียงแค่เพิ่มวันเวลาและชื่อโฮสต์ด้านหน้า ไฟร์สกรีนจะอ่านไฟล์นี้เป็นระยะเพื่อส่งล็อกในส่วนของไอพีเทเบิลไปเก็บยังแอ็คทีฟไดเร็คทอรีบนไฟร์สเตชั่น (ข้อมูลล็อกใน firescreen.log ไม่ได้มีเฉพาะล็อกจากไอพีเทเบิลแต่ประกอบด้วยล็อกอื่นๆในระดับเดียวกันด้วยเช่นกัน)

ข้อมูลภายใน firescreen.log เป็นดังแสดงในรูป

```
Jan 11 02:43:33 firescreen kernel: FIRESCREEN IN=eth2 OUT=eth1 SRC=192.168.2.7
DST=192.168.1.1 LEN=64 TOS=0x00 PREC=0x00 TTL=63 ID=30289 DF PROTO=TCP SPT=2524
 DPT=22 WINDOW=1460 RES=0x00 ACK URGP=0
Jan 11 02:43:57 firescreen kernel: FIRESCREEN IN=eth2 OUT=eth1 SRC=192.168.2.7
DST=192.168.1.1 LEN=64 TOS=0x00 PREC=0x00 TTL=63 ID=30291 DF PROTO=TCP SPT=2524
 DPT=22 WINDOW=1460 RES=0x00 ACK URGP=0
Jan 11 02:44:45 firescreen kernel: FIRESCREEN IN=eth2 OUT=eth1 SRC=192.168.2.7
DST=192.168.1.1 LEN=64 TOS=0x00 PREC=0x00 TTL=63 ID=30293 DF PROTO=TCP SPT=2524
 DPT=22 WINDOW=1460 RES=0x00 ACK URGP=0
```

รูปที่ 11.14 แสดงข้อมูลใน /var/log/firescreen.log

ล็อกจะถูกส่งไปเก็บยังแอ็คทีฟไดเร็คทอรีบนไฟร์สเตชั่นในแอดทริบิวต์ชื่อ firewallLog ภายใต้ DN = "cn=firescreen, cn=computers, dc=hephaestus, dc=com"

สำหรับการตรวจสอบแพ็กเก็ตที่ไม่ตรงตามกฎ เมื่อแพ็คเก็ตใด ๆ ไม่ตรงตามกฎจะถูกส่งเข้าโมดูล ip_queue ของไอพีเทเบิล แล้วถูกตรวจสอบด้วยโปรแกรมสนอร์ตอินไลน์หลังจากนั้นโปรแกรม S2I ในไฟร์สกรีนจะทำการสร้างเส้นทางให้ผู้บุกรุกไปยังเครื่องกับดักไฟร์แทรป โดยในการทดลองนี้ผู้บุกรุกจะทำการสแกนพอร์ตเข้ามายังเครื่องไฟร์เบรก ซึ่งสนอร์ตอินไลน์สามารถตรวจสอบได้ จากนั้นจึงทำการสร้างเส้นทางไปยังเครื่องไฟร์แทรปที่เตรียมไว้ พร้อมทั้งทำการตรวจสอบความอ่อนแอของเครื่องกับดักดังรูป

```
firescreen:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       all  --  isag40.ce.kmitl.ac.th  anywhere            to:172.16.143.133

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
SNAT       all  --  192.168.1.0/24       anywhere            to:161.246.5.48
SNAT       all  --  172.16.142.0/24      anywhere            to:161.246.5.48
MASQUERADE  all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
firescreen:~# _
```

รูปที่ 11.15 แสดงการสร้างเส้นทางให้ผู้บุกรุกเดินทางไปยังเครื่องกับดัก

```
open snort OK
ipsrc = 161.246.5.40
ipdst = 172.16.144.129
row = 1
open cage OK
name = cage2
user limit = 26
check user = 1
group limit = 1
check owner = 1
owner = root
check owner = 1
word[1] = Yes
check_pass = 1
mode = -rw-r--r--
check mode = 1
dbi_cage
cage2pass new = $1$WEk/uzIV$FDqKUSFhJpaSXGM9Lv3ro0
user current = 25group current = check_cage.......
check_cage OK
ipcage =
iptables v1.2.11: Unknown arg `--to-destination'
Try `iptables -h' or 'iptables --help' for more information.
iptables -t nat -A PREROUTING -s 161.246.5.40/32 -i eth0 -j DNAT --to-destinatio
n
pass_old = $1$os9oUeWW$c./cjKgFgwy3QehR5PuL11
```

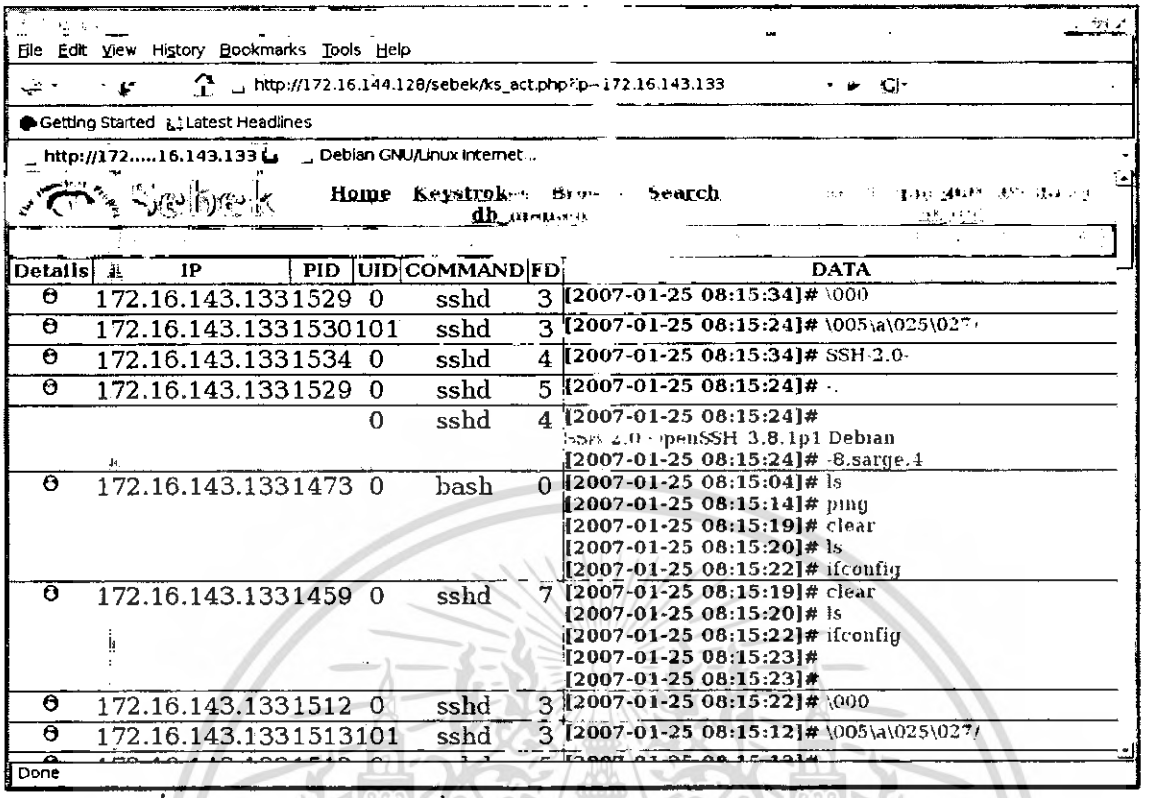
รูปที่ 11.16 แสดงเซอร์วิสที่โปรแกรม S2I ทำงานอยู่เพื่อตรวจสอบความอ่อนแอของเครื่องกับดัก

```
+---------+-------+---------------+---------------------+-----------+--------+
---------------+-------------+--------------+-------------+
|      50 | cage3 | 172.16.143.134 | /root/Desktop/Vmware | WWW       | start  |
             23 |          26 |            1 |           1 |
|      48 | cage1 | 172.16.143.132 | /root/Desktop/Vmware | FTP       | stop   |
             23 |          26 |            1 |           1 |
|      47 | cage2 | 172.16.143.133 | /root/Desktop/Vmware | WWW       | stop   |
             23 |          26 |            1 |           1 |
+---------+-------+---------------+---------------------+-----------+--------+
---------------+-------------+--------------+-------------+
3 rows in set (0.01 sec)

mysql> select * from information;
+---------+-------+---------------+---------------------+-----------+--------+
---------------+-------------+--------------+-------------+
| cage_id | cname | cip           | cpath               | chostname | cstate |
 cuser_current | cuser_limit | cgrp_current | cgrp_limit |
+---------+-------+---------------+---------------------+-----------+--------+
---------------+-------------+--------------+-------------+
|      50 | cage3 | 172.16.143.134 | /root/Desktop/Vmware | WWW       | start  |
             23 |          26 |            1 |           1 |
|      48 | cage1 | 172.16.143.132 | /root/Desktop/Vmware | FTP       | stop   |
             23 |          26 |            1 |           1 |
|      47 | cage2 | 172.16.143.133 | /root/Desktop/Vmware | WWW       | start  |
             23 |          26 |            1 |           1 |
+---------+-------+---------------+---------------------+-----------+--------+
---------------+-------------+--------------+-------------+
3 rows in set (0.00 sec)
```

รูปที่ 11.17 แสดงข้อมูลเกี่ยวกับการทำงานของเครื่องกับดัก

หลังจากที่ผู้บุกรุกถูกสร้างเส้นทางไปยังเครื่องกับดักแล้ว เมื่อผู้บุกรุกทำการติดต่อไปยังเครื่องไฟร์เบรก (172.16.144.129) ผู้บุกรุกจะถูกส่งไปยังเครื่องไฟร์แทรปโดยอัตโนมัติโดยที่ผู้บุกรุกไม่รู้ตัว (172.16.143.133) ซึ่งภายในเครื่องไฟร์เบรกจะมีโปรแกรมเซเบคทำงานอยู่เพื่อบันทึกพฤติกรรมของผู้บุกรุก ซึ่งผู้ดูแลระบบสามารถดูล็อกไฟล์ของเซเบคได้จากทั้งทางหน้าเว็บเพจและทางโปรแกรม Tartarus management



รูปที่ 11.18 แสดงล็อกไฟล์ที่บันทึกพฤติกรรมของผู้บุกรุกจากโปรแกรม Sebek

```
update on FireScreen
Loading /etc/oinkmaster.conf
Downloading file from http://www.bleedingsnort.com/bleeding.rules.tar.gz... done.
Archive successfully downloaded, unpacking... done.
Setting up rules structures... done.
Processing downloaded rules... disabled 0, enabled 0, modified 0, total=1991
Setting up rules structures... done.
Comparing new files to the old ones... done.
Updating local rules files... done.

[***] Results from Oinkmaster started 20070118 03:32:11 [***]

[*] Rules modifications: [*]
    None.

[*] Non-rule line modifications: [*]
    None.

[+] Added files (consider updating your snort.conf to include them if needed): [+]

    -> bleeding-attack_response.rules
    -> bleeding-botcc-BLOCK.rules
    -> bleeding-botcc.rules
    -> bleeding-dos.rules
    -> bleeding-drop-BLOCK.rules
```

รูปที่ 11.19 แสดงขั้นตอนการทำงานเมื่อทำการอัพเดต Signature ของสนอร์ตอินไลน์ ผ่านทางโปรแกรม Tartarus Management

บทที่ 12
บทวิจารณ์และสรุป

12.1 บทสรุป

ชุดโปรแกรมรักษาความปลอดภัยเครือข่ายที่พัฒนาขึ้นมานี้เป็นโครงงานที่เกิดจากการนำแนวคิดการรักษาความปลอดภัยเครือข่ายโดยใช้หลักการทำงานของไฟร์วอลประเภทต่าง ๆ ไม่ว่าจะเป็นเกตเวย์ไฟร์วอล, ไฟร์วอลส่วนบุคคล, รีเวิร์สเว็บพร็อกซี่ และการทำงานของฮันนี่พอตมาประยุกต์ใช้ร่วมกันเพื่อให้กลายเป็นชุดโปรแกรมรักษาความปลอดภัยที่สมบูรณ์แบบ สามารถป้องกันการบุกรุกได้จากทั้งภายในและภายนอกองค์กร สามารถกรั่นกองชิ้นข้อมูลที่เข้ามาในเครือข่ายตามกฎไฟร์วอลที่สร้างขึ้น ซึ่งกฎจะถูกควบคุมจากศูนย์กลางไฟร์สเตชั่น และสำหรับกฎของไฟร์วอลส่วนบุคคลจะไม่สามารถเปลี่ยนแปลงได้โดยผู้ใช้ทั่วไปต้องกระทำโดยผู้ดูแลระบบเท่านั้น อีกทั้งยังสามารถป้องกันเครื่องคอมพิวเตอร์ได้ถึงแม้ผู้ใช้จะยังไม่ล็อกอินเข้าเครื่องคอมพิวเตอร์ก็ตาม ซึ่งจะช่วยเพิ่มความปลอดภัยสำหรับเครื่องคอมพิวเตอร์ส่วนบุคคลมากยิ่งขึ้น

และสำหรับการเชื่อมต่อใด ๆ ที่ไม่ตรงตามกฎของไฟร์วอลที่กำหนดไว้ จะถูกตรวจสอบโดยระบบตรวจจับผู้บุกรุกอีกครั้งหนึ่ง เพื่อจำแนกผู้บุกรุกออกจากผู้ใช้งานจริง โดยถ้ามีพฤติกรรมเข้าข่ายเป็นผู้บุกรุก ระบบจะสร้างเส้นทางให้ผู้บุกรุกเข้าไปในระบบจำลองแล้วทำการบันทึกพฤติกรรมของผู้บุกรุกเพื่อนำไปใช้ในการศึกษารูปแบบการบุกรุกต่อไปได้ในอนาคต

12.2 วิจารณ์สิ่งที่ได้จากโครงงาน

- ไฟร์สกรีนสามารถกรั่นกรองแพ็คเก็ตตามกฎที่ได้รับมาจากไฟร์สเตชั่นและสามารถตรวจสอบแพ็คเก็ตที่ไม่ตรงตามกฎว่าเป็นผู้บุกรุกหรือผู้ใช้งานจริงได้
- แพ็คเก็ตที่ถูกจำแนกว่าเป็นผู้บุกรุกจะถูกส่งไปยังระบบจำลองเพื่อศึกษาพฤติกรรมต่าง ๆ ที่ผู้บุกรุกกระทำต่อระบบต่อไป
- ไฟร์สเตชั่นเป็นศูนย์กลางในการกำหนดกฎของไฟร์วอลชนิดต่าง ๆ ในระบบทั้งส่วนเกตเวย์ไฟร์วอล และ ไฟร์วอลส่วนบุคคล และรับล็อกไฟล์ที่มาจากทั้งสองส่วนนี้
- ไฟร์อลาร์มเป็นไฟร์วอลส่วนบุคคลที่รับกฎจากไฟร์สเตชั่นเท่านั้น โดยที่กฎจะอิงตามผู้ใช้และสามารถตรวจจับการบุกรุกต่าง ๆ ได้
- ไฟร์อลาร์มสามารถป้องกันเครื่องคอมพิวเตอร์ส่วนบุคคลได้ถึงแม้ผู้ใช้จะยังไม่ล็อกอินเข้าเครื่องนั้นก็ตาม
- การติดต่อระหว่างไฟร์สเตชั่นและไฟร์อลาร์มมีความปลอดภัยโดยใช้โพรโตคอล LDAPS ซึ่งใช้ TLS1.0 ในการเข้ารหัส

- โปรแกรม Tartarus Management สามารถควบคุมการได้อย่างปลอดภัยเนื่องจากมีการส่งข้อมูลต่าง ๆ ระหว่างระบบกับโปรแกรมผ่านทางช่องทาง IPsec
- ผู้ใช้งานสามารถเขียนกฎให้ระบบตรวจจับผู้บุกรุกได้ตามต้องการ เพื่อให้มีความเหมาะสมสำหรับการทำงานในแต่ละองค์กรโดยเฉพาะ

12.3 ปัญหาอุปสรรคและแนวทางแก้ไข

- ในการพัฒนาระบบบางส่วนทำงานอยู่คนละแพลตฟอร์ม ทำให้มีปัญหาในการพัฒนาและข้อจำกัดของเครื่องมือต่าง ๆ อยู่บ้าง

- ในการจำลองการทำงานบนเครื่องเสมือน (Virtual Machine) มีข้อจำกัดที่จำนวนอินเทอร์เฟสการ์ด จึงต้องมีการปรับเปลี่ยนโครงสร้างของระบบ

- ในการติดตั้งเครื่องมือและไลบรารี่ต่าง ๆ ที่จำเป็นสำหรับระบบ หากใช้เวอร์ชั่นที่เข้ากันไม่ได้กับระบบปฏิบัติการ จะไม่สามารถติดตั้งระบบได้

- ในขั้นตอนการรับกฎไฟร์วอลใหม่ ๆ ที่ผู้ใช้งานสร้างขึ้นจากส่วนควบคุมกลาง ต้องใช้เวลาซักพักในการเรียนรู้กฎที่รับมาใหม่

- หากผู้ใช้งานไม่มีความเข้าใจในการทำงานของไอพีเทเบิลและสนอร์ตอินไลน์แล้วอาจทำให้ระบบไม่สามารถทำงานได้ตามปกติ

12.4 แนวทางการพัฒนาต่อ

- เพิ่มระบบตรวจจับผู้บุกรุกในไฟร์อลาร์มให้ตรวจจับการบุกรุกได้ครอบคลุมมากขึ้น

- เพิ่มการส่งข้อมูลรับส่งระหว่างไฟร์สเตชั่นและไฟร์สกรีนให้มีความปลอดภัยมากขึ้น

- เมื่อจำแนกผู้บุกรุกใหม่ได้แล้ว ควรพัฒนาให้ไฟร์วอลมีความสามารถในการอัพเดตกฎเพื่อป้องกันผู้บุกรุกที่ตรวจพบนั้น ๆ ได้

- นำไปพัฒนาร่วมกับระบบป้องกันภายในเครื่องคอมพิวเตอร์ (host-based prevention) เพื่อความปลอดภัยของระบบมากยิ่งขึ้น

บทที่ 2
ทฤษฎีและหลักการที่ใช้ในการพัฒนา

2.1 ทฤษฎีและหลักการของไฟร์วอล (Firewall)

ไฟร์วอลเป็นเครื่องมือรักษาความปลอดภัยที่ทำงานในเชิงป้องกัน โดยทำหน้าที่ควบคุมการเข้าถึงเน็ตเวิร์ก ซึ่งแพ็คเก็ตที่สามารถผ่านเข้า-ออกเน็ตเวิร์กได้นั้น จะต้องเป็นแพ็คเก็ตที่ไฟร์วอลเห็นว่ามีความปลอดภัย โดยอาศัยการเปรียบเทียบคุณสมบัติของแพ็คเก็ตที่จะผ่านไฟร์วอลกับกฎที่ได้กำหนดไว้เป็นพื้นฐาน โดยเราสามารถแบ่งชนิดของไฟร์วอลได้หลายประเภทแต่ถ้าคำนึงถึงลักษณะการทำงานของไฟร์วอลที่ใช้ในการตรวจสอบและควบคุม เราสามารถแบ่งไฟร์วอลได้เป็น 3 ประเภท ดังนี้

1. แพ็คเก็ตฟิลเตอริง (Packet Filtering)
2. สเตตฟูลอินสเปกชัน (Stateful Inspection)
3. แอปพลิเคชั่นพร็อกซี่ (Application Proxy)

2.1.1 แพ็คเก็ตฟิลเตอริง (Packet Filtering)

เป็นลักษณะการทำงานโดยทั่วไปของไฟร์วอลที่ใช้ควบคุมทราฟฟิคจะอาศัยการตรวจสอบข้อมูลที่ปรากฏอยู่ในแพ็คเก็ตเฮดเดอร์ เช่น แอดเดรสต้นทาง (Source Address) แอดเดรสปลายทาง (Destination Address) พอร์ต (Port) โพรโตคอล (Protocol) เป็นต้น ซึ่งจากข้อมูลเหล่านี้จะสามารถนำมาใช้เป็นเงื่อนไขสำหรับควบคุมการเข้าออกของข้อมูลได้ โดยการพิจารณาข้อมูลทั้งหมดให้เป็นไปตามกฎที่ระบุไว้ ซึ่งเรียกว่า แอคเซสรูล (Access Rules) หรือ กฎของการควบคุมการผ่านการเข้าออกของแพ็คเก็ต โดยทั่วไปรูปแบบของแอคเซสรูลเบื้องต้นจะเป็นดังนี้

Source Address	Destination Address	Protocol	Service (Dst.Port)	Action

ตารางที่ 2.1 แสดงรูปแบบของแอคเซสรูลเบื้องต้น

โดยข้อมูลทั้งหมดจะเป็นเสมือนตัวแปรที่จะนำมาเปรียบเทียบกับค่าที่ระบุไว้ในแอคเซสรูลทีละค่าเงื่อนไข และในส่วนของฟิลด์สุดท้าย “Action” หมายถึงสิ่งที่ไฟร์วอลจะกระทำเมื่อค่าในแพ็คเก็ตนั้นตรงกับเงื่อนไข ตัวอย่างเช่น

Source Address	Destination Address	Protocol	Service (Dst.Port)	Action
ANY	161.246.5.50	TCP	80	ACCEPT

ตารางที่ 2.2 ตัวอย่างของการตั้งค่าแอคเซสรูลของไฟร์วอล

จากการตั้งแอคเซสรูลเช่นนี้จะหมายความว่าไฟร์วอลจะอนุญาตให้แพ็คเก็ตที่มีต้นทางหมายเลขไอพีแอดเดรส 161.246.5.50 และปลายทางใดๆ ที่ใช้โพรโตคอล TCP หมายเลขพอร์ตปลายทาง 80 ผ่านไฟร์วอลได้ หากไฟร์วอลมีแอคเซสรูลนี้เพียงข้อเดียว ก็เท่ากับอนุญาตให้โฮสต์เพียงโฮสต์เดียวคือ โฮสต์ที่มีไอพีแอดเดรสเป็นหมายเลข 161.246.5.50 เท่านั้นที่สามารถใช้บริการ HTTP (TCP พอร์ต 80) ไปยังโฮสต์อื่นที่อยู่อีกฟากหนึ่งของไฟร์วอลได้

2.1.2 สเตทฟูลอินสเปคชั่น (Stateful Inspection)

ในการทำงานของไฟร์วอลแบบแพ็คเก็ตฟิลเตอริงที่ได้กล่าวมานั้น จะเห็นได้ว่าเราสามารถตั้งค่าเพื่อกรองแพ็กเก็ตที่ไม่ต้องการออกไป แต่ไฟร์วอลดังกล่าวยังไม่ปลอดภัยมากพอ เพราะแม้ว่าเราสามารถจำกัดการเชื่อมต่อให้เหลือแต่พอร์ต 80 เท่านั้นที่ติดต่อเข้ามาได้ แต่ไฟร์วอลข้างต้นก็ไม่ได้ตรวจสอบว่า การเชื่อมต่อผ่านพอร์ต 80 นั้นเป็นการเชื่อมต่อตามปรกติหรือไม่ คือ หากเป็นพอร์ต 80 แล้วก็จะปล่อยผ่านหมด ทั้งที่อาจเป็นแพ็กเก็ตโจมตีก็ได้ เพราะแฮกเกอร์อาจเทลเน็ตไปยังพอร์ต 80 เพื่อรันโปรแกรม Backdoor ได้ นอกจากนั้นกรณีที่แพ็กเก็ตที่ผ่านไฟร์วอลมาเป็นแพ็กเกจที่มีการทำ Fragmentation มาด้วยแล้ว ไฟร์วอลประเภทนี้จะไม่สามารถตรวจสอบอะไรได้เลย

และหากเป็นกรณีที่เป็นการโจมตีโดยการกำหนดแพ็กเกจที่ผิดปรกติแล้วไฟร์วอลแบบแพ็คเก็ตฟิลเตอริงจะไม่สามารถตรวจสอบได้เช่นกัน เพราะหากหมายเลขไอพีและพอร์ตผ่านไฟร์วอลประเภทนี้จะยอมให้ผ่านได้ทันที และหากมีการปลอมหมายเลขไอพีแอดเดรสด้วยแล้วยิ่งทำให้การป้องกันของไฟร์วอลมีความสามารถลดลง นอกจากนั้นสำหรับพอร์ตที่มีหมายเลขมากกว่า 1024 ไฟร์วอลดังกล่าวจะต้องเปิดพอร์ตเหล่านั้นไว้ตลอดเวลา เพราะไฟร์วอลไม่รู้ว่าแอปพลิเคชั่นใดจะใช้งานพอร์ตหมายเลขใดบ้าง ซึ่งถือเป็นความไม่ปลอดภัยอีกเช่นกัน

ซึ่งข้อบกพร่องทั้งหมดนี้ ไฟร์วอลแบบสเตทฟูลอินสเปคชั่น สามารถป้องกันได้ (ต่อไปจะเรียกว่า สเตทฟูล) โดยไฟร์วอลประเภทนี้จะมีการตั้งกฎขึ้นมาเช่นเดียวกัน แต่ไฟร์วอลประเภทนี้จะมีความสามารถในการติดตามสถานะการเชื่อมต่อ โดยเฉพาะการเชื่อมต่อแบบ TCP ซึ่งในระหว่างการเชื่อมต่อ ไฟร์วอลจะทำการสร้างตารางสถานะ (State Table) ที่จะเก็บสถานะของการเชื่อมต่อในทุก ๆ การเชื่อมต่อเอาไว้ เช่น การเชื่อมต่อในแบบ TCP จะต้องเริ่มด้วย 3 Way Handshake ไฟร์วอลสเตทฟูลจึงต้องทำการตรวจสอบว่าแพ็กเก็ตแรกว่าเป็น SYN หรือไม่ และแพ็กเก็ตตอบกลับเป็น SYN/ACK หรือไม่ และแพ็กเก็ตขืนยันเป็น ACK หรือไม่ และยังตรวจสอบอีกว่าแต่ละแพ็กเกจมีลำดับของ Sequence Number ถูกต้องหรือไม่ นอกจากในระหว่างการเชื่อมต่อ ไฟร์วอลสเตทฟูลยังมีการตรวจสอบหมายเลขลำดับ หมายเลขตอบรับและแฟล็กต่าง ๆ ตลอดเวลา ดังนั้นจะเห็นได้ว่าการสร้างแพ็กเก็ตแปลกปลอมให้ผ่านไฟร์วอลแบบสเตทฟูลจะยากขึ้นมาก

สำหรับกรณีของแพ็กเก็ตที่มีการทำ Fragmentation มานั้น ไฟร์วอลสเตทฟูลจะรอจนครบทั้ง ตาด้าแกรมแล้วจึงทำการจัดเรียงแพ็คเก็ตใหม่ (Reassemble) แล้วจึงตรวจสอบว่าถูกต้องหรือไม่

ดังนั้นการโจมตีโดยวิธีการแบ่งเป็น Fragment ย่อย ๆ เพื่อหลอกไฟร์วอลก็จะทำได้ยากขึ้น และไฟร์วอลแบบสเตทฟูลไม่จำเป็นต้องเปิดพอร์ตหมายเลข 1024 ขึ้นไป ทิ้งเอาไว้ด้วย เพราะเมื่อไฟร์วอลสามารถติดตามสถานะการเชื่อมต่อได้แล้ว ก็ย่อมจะรู้ว่าการเชื่อมต่อนั้น ๆ ฝั่งไคลเอนต์มีการใช้งานพอร์ตใดเพื่อจะได้เปิดพอร์ตนั้น "เฉพาะ" สำหรับไคลเอนต์ และเมื่อการเชื่อมต่อจบลงก็จะปิดพอร์ตนั้นไว้เหมือนเดิมทำให้ระบบมีความปลอดภัยเพิ่มขึ้นมาก

อย่างไรก็ตามไฟร์วอลแบบสเตทฟูลนั้น ไม่ได้ทำงานเหมือนกันไปหมด ไฟร์วอลสเตทฟูลบางตัวจะขยับไปทำงานที่ชั้นแอปพลิเคชั่นในบางโพรโตคอล เช่น ในโพรโตคอล FTP นั้นจะมีการใช้พอร์ต 2 พอร์ต คือ พอร์ต 21 ใช้งานเป็นพอร์ตควบคุม และพอร์ต 20 เป็นพอร์ตข้อมูล ดังนั้นในการทำงานแบบสเตทฟูลนั้น สมมติว่าในครั้งแรกเครื่องไคลเอนต์ 161.246.5.10 ติดต่อมายังพอร์ต 21 ของเซิร์ฟเวอร์ 161.246.4.3 โดยจะสุ่มค่าพอร์ตฝั่งไคลเอนต์มาด้วย เช่น 1234 ซึ่งไฟร์วอลก็จะกำหนดในตารางสถานะ (State Table) ว่า ไอพี 161.246.4.3 พอร์ต 21 กับไอพี 161.246.5.10 พอร์ต 1234 มีการเชื่อมต่อกัน หลักจากนั้นไฟร์วอลจะเปิดพอร์ตทั้ง 2 เอาไว้ แต่เมื่อมีการส่งข้อมูล เครื่องไคลเอนต์กลับติดต่อมายังพอร์ต 20 ซึ่งไฟร์วอลที่ไม่เข้าใจการทำงานของโพรโตคอล FTP จะบล็อกข้อมูลทำให้ไม่สามารถเชื่อมต่อผ่านพอร์ต 20 เพื่อส่งข้อมูลได้ หรือในการส่งข้อมูลประเภทมัลติมีเดีย เช่น H.323 นั้น พอร์ตควบคุมกับพอร์ตที่ใช้ในการส่งข้อมูลจะเป็นคนละพอร์ต เช่นเดียวกับ FTP และบางครั้งยังเป็นการใช้พอร์ต UDP อีกด้วย ดังนั้นหากไฟร์วอลไม่มีการทำงานในชั้นแอปพลิเคชั่น หรือไม่เข้าใจโพรโตคอลนั้น ๆ แล้ว จะทำให้ใช้งานแอปพลิเคชั่นนั้น ๆ ไม่ได้ หรือหากต้องการจะทำให้ได้ ก็ต้องเปิดพอร์ตนั้นทิ้งเอาไว้ถาวร

สำหรับสเตทฟูลไฟร์วอลบางตัวจะสามารถทำงานในชั้นแอปพลิเคชั่นได้ แต่ยังไม่สามารถป้องกันการโจมตีที่แทรกซึมมากับการเชื่อมต่อตามปรกติได้ เช่น โพรโตคอล FTP นั้น แม้ว่าไฟร์วอลแบบสเตทฟูลบางตัวจะมีการติดตามให้มีการเปิดพอร์ต 20 และ 21 อย่างถูกต้อง แต่ถ้าหากในเนื้อหาที่ส่งมาเป็นสิ่งที่ประสงค์ร้าย ก็จะไม่สามารถรับรู้และป้องกันการโจมตีแบบนี้ได้

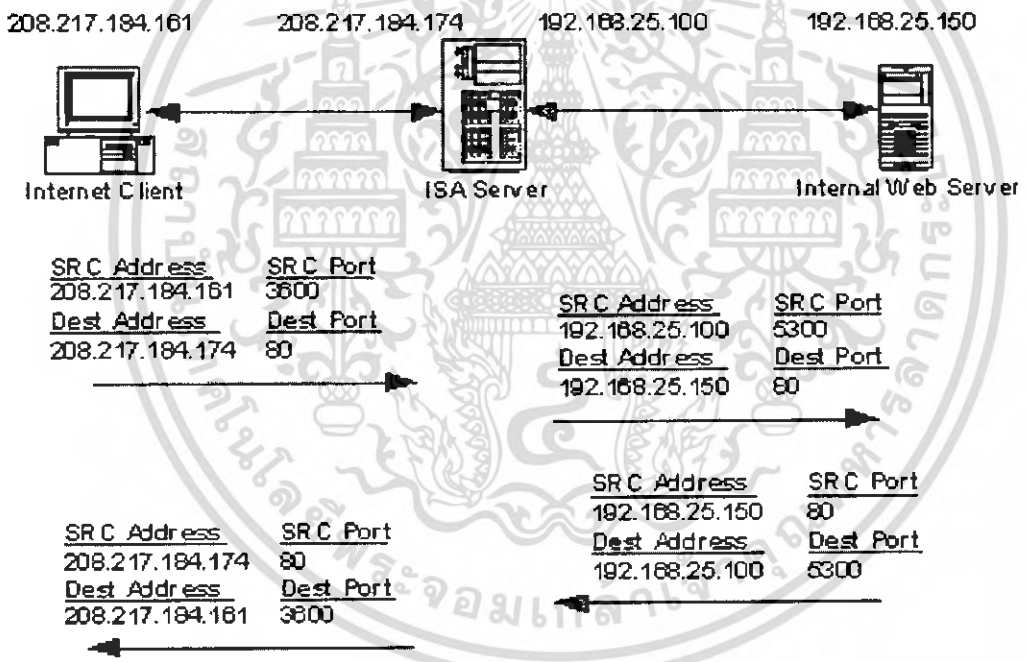
นอกจากนั้นในไฟร์วอลหลายตัว จะมีการติดตามสถานะการทำงานของ HTTP เป็นพิเศษ เพราะเป็นที่ทราบกันดีว่าข้อมูลในโลกนี้ มีบทบาทของ HTTP อยู่ไม่น้อย และมีการโจมตีผ่านทางเว็บมาก ดังนั้นการติดตามสถานะของ HTTP จะทำให้ระบบเครือข่ายมีความปลอดภัยมากขึ้น

2.1.3 แอปพลิเคชั่นพร็อกซี่ (Application Proxy)

แม้ว่าไฟร์วอลแบบสเตทฟูลจะมีความปลอดภัยเพิ่มขึ้นมากแล้วก็ตาม แต่ไฟร์วอลประเภทนี้ยังไม่สามารถป้องกันการโจมตีที่แทรกซึมมากับการเชื่อมต่อตามปรกติได้ เช่น โพรโตคอล FTP นั้น แม้ว่าไฟร์วอลแบบสเตทฟูลจะมีการติดตามให้มีการเปิดพอร์ต 20 และ 21 อย่างถูกต้อง แต่หากมีการส่งข้อมูลอื่น ๆ ที่ไม่ใช่ข้อมูล FTP แทรกมาในระหว่างการเชื่อมต่อ สเตทฟูลไฟร์วอลจะไม่รู้ ดังนั้นหากต้องการให้ไฟร์วอลมีความสามารถในการติดตามการทำงานของชั้นแอปพลิเคชั่นมาก

ขึ้นโดยทราบว่าการติดต่อเป็นการติดต่อตามรูปแบบของโพรโตคอลนั้น ๆ อย่างถูกต้องหรือไม่ ไฟร์วอลจะต้องก้าวขึ้นไปทำงานในชั้นแอปพลิเคชั่น โดยจะเรียกไฟร์วอลชนิดนี้ว่า Application Proxy Firewall

ไฟร์วอลประเภทนี้ จะทำงานในระดับชั้นแอปพลิเคชั่นเป็นสำคัญ โดยไฟร์วอลประเภทนี้จะทำหน้าที่เป็นตัวแทน (Proxy) ในการส่งต่อการเชื่อมต่อใด ๆ ไปยังเซิร์ฟเวอร์ เช่น เมื่อไคลเอนต์ 161.246.5.10 ติดต่อไปยังเว็บเซิร์ฟเวอร์ 161.246.4.7 ผ่านทางไฟร์วอล 161.246.5.1 ไฟร์วอลทำหน้าที่รับแพ็กเก็ตที่ขอเชื่อมต่อจาก 161.246.5.10 เอาไว้ จากนั้นจะทำการถอดแพ็กเก็ตเดิมออก แล้วสร้างแพ็กเก็ตใหม่ โดยอาศัยข้อมูลร้องขอเดิมไปยัง 161.246.4.7 เสมือนกับว่าไฟร์วอลเป็นผู้ร้องขอเอง ดังนั้นข้อดีประการแรกของไฟร์วอลประเภทนี้ คือ บุคคลภายนอกจะไม่รู้หมายเลขไอพีของเครือข่ายภายในที่ขอเชื่อมต่อออกมาภายนอก และเมื่อเว็บเซิร์ฟเวอร์ตอบกลับก็จะส่งผลลัพธ์กลับไปยังเครื่อง 161.246.5.10



รูปที่ 2.1 แสดงการทำงานของแอปพลิเคชั่นพร็อกซี่

และเนื่องจากทำหน้าที่เป็นตัวแทนในการเชื่อมต่อนี่เอง ทำให้ในการติดต่อผ่านไฟร์วอลทุกครั้ง ไฟร์วอลจะทำหน้าที่ในการตรวจสอบรูปแบบการติดต่อว่ามีรูปแบบที่ถูกต้องตามโพรโตคอลนั้น ๆ หรือไม่ ทำให้มีความปลอดภัยเพิ่มขึ้น นอกจากนั้นไฟร์วอลแบบนี้ยังป้องกันการปลอมไอพีได้โดยเด็ดขาดเนื่องจากไฟร์วอลจะทำหน้าที่ในการส่งต่อเสียเอง แต่การติดต่อแบบนี้ยังมีข้อเสียเช่นกัน คือ ไฟร์วอลแบบนี้จะทำงานได้ช้ากว่าเพราะมีการตรวจสอบมากกว่า อีกทั้งยังต้องสร้างแพ็กเก็ตใหม่ในทุก ๆ ครั้งด้วย และหากกรณีที่ในการเชื่อมต่อหนึ่งมีการแบ่งเป็นเซกเมนต์

(Segment) หลาย ๆ เซกเมนต์ ด้วยแล้ว ไฟร์วอลประเภทนี้ต้องรอให้ทุกเซกเมนต์ ส่งมาจนครบก่อนจึงจะส่งต่อได้ ทำให้เกิดความล่าช้าในการทำงานขึ้น นอกจากนั้นการที่ไฟร์วอลประเภทนี้ทำงานในระดับชั้นแอปพลิเคชั่น หมายความว่า มันจะส่งต่อเฉพาะโพรโตคอลที่รู้จักเท่านั้น หากโพรโตคอลใดที่ไม่รู้จักจะไม่สามารถส่งต่อได้เลย ในขณะที่หากเป็นไฟร์วอลแบบสเตทฟูลแล้ว เราสามารถตั้งค่าให้เปิดพอร์ตต่าง ๆ ทำให้สามารถเชื่อมต่อได้ แต่ไฟร์วอลแบบแอปพลิเคชั่นพร็อกซี่จะไม่สามารถติดต่อได้เลยหากไฟร์วอลไม่รู้จักกับโพรโตคอลนั้น

จากข้อดีของไฟร์วอลแบบสเตทฟูลที่มีความรวดเร็วในการทำงาน และข้อดีของไฟร์วอลแบบแอปพลิเคชั่นพร็อกซี่ที่มีความปลอดภัยสูงจึงทำให้มีผู้สร้างไฟร์วอลที่ผสมผสานความสามารถของไฟร์วอลทั้งสองขึ้น และเรียกไฟร์วอลแบบใหม่นี้ว่า Hybrid Firewall หรือบางผู้ผลิตจะเรียกว่า Adaptive Firewall โดยไฟร์วอลแบบใหม่นี้จะแตกต่างกันไปตามผู้ผลิต บางผลิตภัณฑ์ทำงานในแบบพร็อกซี่สำหรับโพรโตคอลหลัก ๆ และแบบสเตทฟูลสำหรับโพรโตคอลทั่ว ๆ ไป บางผลิตภัณฑ์ทำงานในแบบพร็อกซี่ในช่วงแรกของการเชื่อมต่อเพราะมีความปลอดภัยสูง และต่อมาหากเชื่อว่าการเชื่อมต่อนั้นเป็นการเชื่อมต่อตามปรกติ ก็จะขยับลงมาทำงานในแบบสเตทฟูลเพราะมีความรวดเร็วในการทำงานมากกว่า

ไฟร์วอลในปัจจุบันมีการพัฒนาไปมากจนอาจกล่าวได้ว่าไม่มีผลิตภัณฑ์ไฟร์วอลใด ที่เป็นแบบใดแบบหนึ่ง นอกจากนั้นผลิตภัณฑ์ไฟร์วอลมักจะมีการทำงานร่วมกับแอปพลิเคชั่นด้านความปลอดภัยอื่น ๆ เช่น ทำงานร่วมกับ Antivirus ทำให้ไฟร์วอลส่งข้อมูลของเมลล์ไปตรวจสอบไวรัสก่อนจะส่งต่อ หรือทำงานร่วมกับ IDS เพื่อตรวจสอบการบุกรุก ไฟร์วอลบางตัวสามารถตั้งให้ทำงานตามเวลา ตามผู้ใช้ บางตัวสามารถกำหนดทราฟฟิกให้กับแอปพลิเคชั่นไม่เท่ากันได้ด้วย

2.2 ทฤษฎีและหลักการของฮันนี่พอต (Honeypot)

แนวคิดของฮันนี่พอตคือ ต้องการทราบการกระทำ และความพยายามต่างๆที่ผู้บุกรุกกระทำต่อระบบ ดังนั้นเพื่อให้ได้มา จำเป็นจะต้องสร้างระบบบางอย่างให้เหล่าผู้บุกรุกได้เข้ามาและระบบต้องสามารถจับตาดูได้ โดยที่ผู้บุกรุกไม่รู้ตัวว่ากำลังโดนจับตามองอยู่และโดยที่ไม่ส่งผลเสียหายใดๆ ต่อระบบจริง เปรียบเสมือนล่อหลอกให้ขโมยพยายามงัดเข้ามาในบ้าน แล้วใช้กล้องวงจรปิดถ่ายภาพไว้ทุกๆ การกระทำตั้งแต่วิธีที่ใช้ในการงัดเข้ามาในบ้าน พยายามทำอะไรบ้าง สนใจสำรวจอะไรบ้าง หรือได้นำอะไรออกไป โดยที่บ้านและของมีค่าใดๆ เป็นเพียงภาพลวงตาเท่านั้น ทำให้ไม่ต้องเสียทรัพย์สินหรือก่อให้เกิดอันตรายใด ๆ ซึ่งทำให้เราสามารถเห็นวิธีที่ผู้บุกรุกใช้ ตลอดจนเครื่องมือใหม่ๆ ที่ผู้บุกรุกอาจนำมาใช้ ซึ่งเมื่อเราทราบวิธีและเครื่องมือของผู้บุกรุก ทำให้เราสามารถนำข้อมูลที่ได้ไปประยุกต์เพื่อปรับปรุงและป้องกันระบบจริงให้มีความปลอดภัยมากยิ่งขึ้น ตั้งแต่อดีตจนถึงปัจจุบันระบบฮันนี่พอตได้มีวิวัฒนาการและการพัฒนาระบบมาตามลำดับ โดยเราสามารถแบ่งชนิดของฮันนี่พอตได้ดังต่อไปนี้

2.2.1 ระบบฮันนี่พอตแบบ Low-interaction

ในยุคแรกนี้ ระบบฮันนี่พอตเป็นเพียงการสร้างการบริการเสมือนขึ้นมา (เซอร์วิสที่จำลองขึ้นมาให้เหมือนเซอร์วิสจริงๆ) เพื่อจำลองบริการต่างๆ ขึ้นทำให้ดูเสมือนว่ามีเครื่องที่เปิดให้บริการต่างๆ เช่น FTP อยู่เป็นต้น ซึ่งฮันนี่พ็อตชนิดนี้จะมีข้อจำกัดในส่วนของการโต้ตอบการโจมตีของผู้บุกรก ซึ่งการโต้ตอบของฮันนี่พ็อตรูปแบบนี้จะขึ้นอยู่กับการทำส่วนจำลองบริการและขึ้นอยู่กับระบบปฏิบัติการที่ใช้ เช่น อาจทำให้สามารถล็อกอินได้ หรือทำให้ใช้คำสั่งต่างๆ ได้เหมือนกับบริการจริงๆ ตัวอย่างฮันนี่พอตที่เป็นแบบ Low-interaction เช่น Specter, Honeyd และ KFSensor

ข้อดีของฮันนี่พ็อตแบบนี้คือมีความยืดหยุ่น ง่ายต่อการเปลี่ยนแปลงและการบำรุงรักษาตัวฮันนี่พ็อต และมีอัตราเสี่ยงเล็กน้อย ซึ่งโดยทั่วไปแล้วมีระบบ Plug & Play ซึ่งจะทำให้การเปลี่ยนแปลงเป็นไปได้โดยสะดวก การจำลองบริการนั้นจะช่วยลดความเสี่ยงโดยสามารถบรรจุสิ่งที่จะใช้ตอบโต้ผู้บุกรุก ผู้บุกรุกจะไม่สามารถเข้ายึดครองเครื่องฮันนี่พอตนั้นเพื่อใช้ในการโจมตีผู้อื่นต่อไปได้

ข้อเสียของฮันนี่พ็อตแบบ Low-interaction นี้คือสามารถออกแบบได้เฉพาะการตรวจจับการกระทำที่ทราบอยู่แล้วเท่านั้น ซึ่งมีข้อจำกัดมากมายในการเฝ้าดูพฤติกรรมผู้บุกรุก เพราะสิ่งที่เห็นจะแค่เพียงส่วนที่เกี่ยวข้องกับบริการนั้นเท่านั้น ไม่เกี่ยวกับเรื่องของระบบปฏิบัติการเลย

2.2.2 ระบบฮันนี่พอตแบบ High-interaction

ต่อมามีการสร้างเป็นเครื่องให้บริการเสมือน (เสมือนเป็นเครื่องคอมพิวเตอร์ที่ทำงานจริงซึ่งจะมีเรื่องของระบบปฏิบัติการเข้ามาเกี่ยวข้อง) เรียกระบบฮันนี่พอตแบบนี้ว่าเป็นแบบ High-interaction ซึ่งฮันนี่พอดแบบนี้จะมีทั้งระบบปฏิบัติการ แอปพลิเคชั่น และบริการจริงๆ เพื่อตอบโต้กับผู้บุกรุก นั่นคือถ้าต้องการให้มีบริการใดก็ทำการลงโปรแกรม หรือเปิดบริการต่างๆ ที่เครื่องจริงๆ อย่างเช่น ถ้าจะเปิดบริการ FTP ก็ลงโปรแกรมอย่างเช่น Ftpd เป็นต้น

ข้อดีของฮันนี่พ็อตแบบ High-interaction คือข้อมูลที่ได้จากการกระทำของผู้บุกรุกนั้นมีความครอบคลุมมากกว่าเพราะว่ามีการโต้ตอบทุกอย่างที่เหมือนจริง ดังนั้นเราจึงสามารถสรุปผลข้อมูลได้จากการโจมดีจริง ระบบจะตอบโต้กับผู้บุกรุกจริงๆ เราจึงสามารถเรียนรู้ขอบเขตของพฤติกรรมของเหล่าผู้บุกรุกได้ดี และข้อดีอีกข้อหนึ่งคือฮันนี่พ็อดแบบ High-interaction นั้นไม่ได้จำลองการตอบโต้กลับไปยังผู้บุกรุกว่าควรจะดอบกลับแบบใด เพราะใช้เป็นตัวระบบจริงที่ติดตั้งอยู่บนฮันนี่พ็อต สามารถจับตาดูการกระทำต่อระบบซึ่งอาจส่งผลต่อบริการที่เครื่องนั้นๆ ให้บริการอยู่ หรือข้อมูลที่ผู้บุกรุกสนใจ และทำการสำรวจหรือนำออกไปทำให้สามารถเฝ้าดูพฤติกรรมได้ครอบคลุมกว่าแบบเดิมมาก

ข้อเสียของฮันนี่พ็อตแบบ High-interaction คือการเพิ่มอัตราเสี่ยงให้กับระบบ เนื่องจากผู้บุกรุก อาจจะสามารถเข้ายึดครองเครื่องที่ทำหน้าที่เป็นฮันนี่พ็อต แล้วใช้เครื่องนั้นๆ ในการโจมตีระบบอื่นๆ ก็เป็นไปได้ ดังนั้นการใช้งานฮันนี่พ็อตที่เป็นแบบ High-interaction นั้นจึงต้องหาทาง

ควบคุมพฤติกรรมของผู้บุกรุกให้อยู่ภายในขอบเขตที่เรากำหนดและไม่สามารถใช้ระบบฮันนี่พ็อตเป็นฐานที่มั่นในการเข้าโจมตีระบบอื่นๆได้ ตัวอย่างของตัวฮันนี่พ็อตแบบ High-interaction ได้แก่ Honeynets เป็นต้น โดยการกระทำที่สามารถจับตาดูได้ในยุคนี้จะเป็นการกระทำที่ไม่มีเรื่องของการเข้ารหัสลับเข้ามาเกี่ยวข้อง เช่นการทำงานผ่าน telnet ซึ่งไม่มีการเข้ารหัสข้อมูลก่อนทำการส่ง ดังนั้นเมื่อมีเทคโนโลยีหรือเครื่องมือใหม่ๆที่มีการเข้ารหัส เช่น SSL, SSH, SCP ก็จะไม่สามารถดักดูข้อมูลได้

ต่อมาได้มีการนำเครื่องมือที่มีลักษณะของ Rootkits ซึ่งมีการทำงานแบบ Kernel-base tools เข้ามาใช้งานเพื่อให้สามารถเข้าถึงข้อมูลที่ทำการเข้ารหัส โดยที่เครื่องมือเหล่านี้ไม่ได้ไปนำข้อมูลมาถอดรหัส แต่จะเป็นการดูข้อมูลก่อนที่จะถูกเข้ารหัส(กรณีที่เป็นฝั่งส่ง) และอ่านข้อมูลหลังจากที่ถอดรหัสเรียบร้อยแล้ว (ในฝั่งรับ) ทำให้สามารถจับตาดูการกระทำของผู้บุกรุกได้ แม้จะใช้ SSH เข้ามายังเครื่องกับดัก ดังนั้นสิ่งที่เราสามารถบันทึกได้จะครอบคลุมพฤติกรรมของผู้บุกรุกทั้งหมด เช่น การกด keyboard การใช้คำสั่ง การถ่ายโอนไฟล์ การนำข้อมูลเข้าออก หรือการเรียกใช้โปรแกรม อีกทั้งระบบยังได้มีการเพิ่มคุณสมบัติในการซ่อนตัวเองไม่ให้ผู้บุกรุกตรวจพบว่ากำลังทำงานอยู่ และในยุคนี้ได้เริ่มนำฮันนี่พอตเข้ามาเป็นส่วนหนึ่งของระบบจริง นั้นคือมีการทำการคัดแยกผู้บุกรุกออกจากผู้ใช้งานทั่วไป โดยประกอบขึ้นด้วยหลายๆส่วน ซึ่งทำหน้าที่ต่างๆกันไป เช่น Honeywall, Log server, Cage ซึ่งเครื่องกับดักนั้นจะถูกแยกออกจากเครื่องให้บริการจริง เพื่อไม่ให้การกระทำต่อเครื่องกับดักเกิดความเสียหายต่อเครื่องให้บริการจริง

2.3 ความรู้พื้นฐานเกี่ยวกับการบุกรุก

ประเภทของผู้บุกรุก

- ผู้บุกรุกจากภายนอก (Outsider intruder) หมายถึง บุคคลภายนอกเครือข่ายที่พยายามเจาะเข้ามาในระบบหรือพยายามโจมตีระบบจากภายนอก เช่น การเจาะเข้ามายังเครื่องเว็บเซิร์ฟเวอร์แล้วเปลี่ยนหน้าเว็บไซต์ของเรา เป็นต้น ซึ่งการบุกรุกนี้อาจมาจากอินเตอร์เน็ต, การ dial-up หรือการบุกเข้าไปยังเครือข่ายของคู่ค้าแล้วเชื่อมต่อมายังเครือข่ายของเรา

- ผู้บุกรุกจากภายใน (Insider intruder) บุคคลภายในเครือข่าย รวมทั้งผู้ใช้ที่ใช้สิทธิ์ในทางที่ผิด หรือการลักลอบใช้สิทธิ์ของผู้ใช้คนอื่นๆ ที่มีสิทธิ์เหนือกว่า

ประเภทของการบุกรุก

- การบุกรุกทางกายภาพ (Physical Intrusion) ผู้บุกรุกพยายามบุกรุกที่เครื่องโดยตรง อาจเข้ามาใช้สิทธิ์พิเศษจากการทำงานที่คอนโซล หรือถอดย้ายอุปกรณ์ เช่น ฮาร์ดดิสก์ ซึ่งอาจนำไปเขียนหรืออ่านภายหลัง หรือบายพาสไบออสได้

- การบุกรุกทางระบบ (System Intrusion) ผู้บุกรุกที่เข้ามาในระบบ โดยปรกติมักเป็นผู้ใช้ที่มีสิทธิ์ต่ำ ถ้าระบบไม่ได้ใส่แพตช์ (patch) ที่สามารถแก้ช่องโหว่ของแอปพลิเคชั่นแล้ว จุดนี้อาจจะเป็นช่องโหว่ที่ทำให้ผู้ใช้คนนั้นสร้างสิทธิ์ของตัวเองให้มากขึ้น จนเทียบเท่าผู้ดูแลระบบได้ เนื่องจากแอปพลิเคชั่นที่ใช้งานเกือบทุกอย่างมีช่องโหว่อยู่ ถ้ายังไม่สามารถทำให้ช่องโหว่นั้นหมดไปหรือลดลงได้ จุดนี้จะกลายเป็นช่องทางสำหรับการบุกรุกระบบ

- การบุกรุกระยะไกล (Remote Intrusion) ผู้บุกรุกติดต่อผ่านทางเน็ตเวิร์ก มีหลายเทคนิคในการบุกรุกระบบแบบนี้ ปัจจุบันมีแอปพลิเคชั่นประเภทไฟร์วอล (Firewall) ทำหน้าที่เป็นด่านแรกในการป้องกันการบุกรุกทางเน็ตเวิร์ก

ตัวอย่างการบุกรุก

- Non-blind spoofing เป็นการโจมตีที่เกิดขึ้นได้เมื่อผู้บุกรุกอยู่บนซับเน็ตเดียวกันกับเครื่องเป้าหมายซึ่งผู้บุกรุกสามารถมองเห็นลำดับของหมายเลขลำดับ (Sequence number) และ acknowledgement ของแพ็คเก็ตต่างๆได้ เมื่อผู้บุกรุกรู้ลำดับของแพ็คเก็ตซึ่งอาจทำได้โดยการใช้โปรแกรมดักจับแพ็คเก็ตที่วิ่งไปมาอยู่ในเครือข่ายแล้วนำมาพิจารณาแล้ว ผู้บุกรุกสามารถทำ session hijacking ได้จากนั้นผู้บุกรุกจะทำการดักเอาส่วนของการยืนยันตนต่างๆ ที่ใช้ในการสถาปนาการเชื่อมต่อไว้ ทำให้การสถาปนาเชื่อมต่อนั้นล้มเหลว จากนั้นจึงทำการสถาปนาการเชื่อมต่อใหม่โดยการแก้ไข sequence number และ acknowledgement number ให้ตรงกับเครื่องของผู้บุกรุกก็จะถือว่าการโจมตีนี้ทำได้สำเร็จ

- Blind spoofing เป็นการโจมตีที่เกิดขึ้นจากภายนอกซึ่งผู้บุกรุกไม่อาจรู้ลำดับ sequence และ acknowledgement ดังนั้นผู้บุกรุกจึงพยายามส่งแพ็คเก็ตจำนวนมากไปยังเครื่องเป้าหมายอย่างสม่ำเสมอ ตามลำดับของหมายเลขลำดับ (Sequence number) ที่สุ่มขึ้นมา แต่ในปัจจุบันนี้ระบบปฏิบัติการได้มีการพัฒนาเรื่องของการสุ่มหมายเลขลำดับ ทำให้การคาดเดาให้ถูกต้องนั้นเป็นไปได้ยาก แต่ถึงอย่างไรก็ตามถ้าหมายเลขลำดับ ที่ผู้บุกรุกสุ่มขึ้นมานั้นถูกต้อง ข้อมูลจากเครื่องผู้บุกรุกก็จะถูกส่งไปยังเครื่องเป้าหมายเช่นกัน

- Man in the Middle Attack เรียกได้อีกอย่างหนึ่งว่า connection hijacking ซึ่งการโจมตีแบบนี้จะเป็นการเข้าไปขัดขวางการสื่อสารระหว่างเครื่องโฮสต์ 2 เครื่องและเข้าควบคุมการสื่อสารเพื่อสับเปลี่ยนแก้ไข หรือเพิ่มข้อมูลที่เครื่องทั้ง 2 ส่งถึงกันได้ โดยการที่ผู้บุกรุกจะทำเช่นนี้ได้ผู้บุกรุกเองต้องอยู่ในสถานที่เดียวกับเครื่องทั้ง 2 เครื่องที่ติดต่อกันโดยหัวใจหลักของการโจมตีแบบนี้คือการทำให้การติดต่อระหว่างเครื่องโฮสต์อยู่ในสถานะที่เรียกว่า "desynchronized" นั่นคือการทำให้เลขลำดับ sequence ของแพ็คเก็ตที่ได้รับไม่ตรงกับเลขลำดับลำดับที่ต้องการนั่นเอง

- Denial of Service Attack คือ IP spoofing ที่ใช้บ่อยที่สุด ในการโจมตีแบบ denial of service หรือ DoS ผู้บุกรุกจะทำการใช้แบนด์วิทธ์และทรัพยากรเครือข่ายให้หมดไปโดยการส่งแพ็คเก็ตจำนวนมากเท่าที่จะทำได้ไปยังเครื่องเป้าหมายในเวลาอันสั้น โดยผู้บุกรุกจะทำการเปลี่ยนไอพีต้นทางไปเรื่อยเพื่อทำให้การหยุดDoSทำได้ยากยิ่งขึ้น

2.4 ระบบตรวจจับผู้บุกรุกผ่านเครือข่าย (NIDS)

ระบบตรวจจับผู้บุกรุกทางเครือข่าย (Network Intrusion Detection System : NIDS) เป็นระบบตรวจจับแพ็กเก็ตที่วิ่งอยู่ในเครือข่ายโดยพยายามตรวจสอบว่ามีผู้บุกรุกพยายามบุกรุกเข้าสู่ระบบหรือไม่ ตัวอย่างของระบบนี้ เช่น ระบบที่ตรวจสอบว่ามีแพ็คเก็ตของการเชื่อมต่อแบบทีซีพี/ไอพี (TCP/IP) ชื่อว่า SYN ส่งเข้ามายังระบบเป็นจำนวนมากอย่างผิดปรกติในเครื่องเป้าหมายหรือไม่ ซึ่งการกระทำแบบนี้สามารถวิเคราะห์ได้ว่ามีผู้บุกรุกพยายามสแกนพอร์ตทีซีพี (TCP port) ของเครื่องอยู่ ระบบตรวจจับผู้บุกรุกทางเน็ตเวิร์กสามารถทำงานที่เครื่องเป้าหมาย หรือเครื่องที่ทำหน้าที่เฉพาะที่ใช้ในการเฝ้าดูเหตุการณ์ไม่ปรกติในเครือข่ายก็ได้ ดังนั้นระบบตรวจจับผู้บุกรุกทางเครือข่ายสามารถตรวจสอบเครื่องใด ๆ ก็ได้ในเครือข่าย

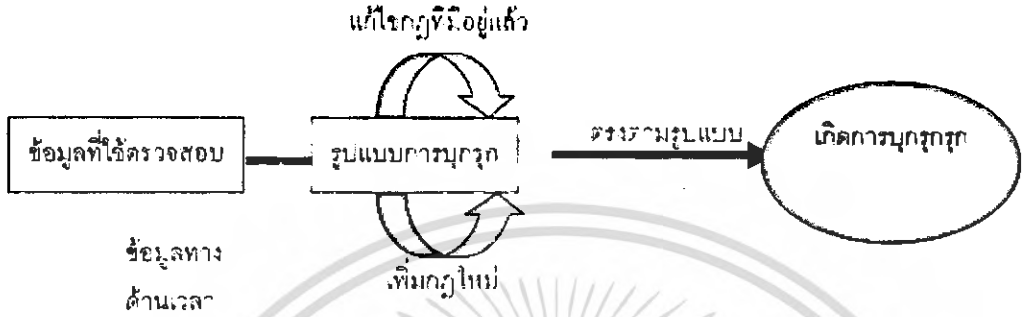
ระบบตรวจจับผู้บุกรุกมีหลักการทำงานต่างไปจากไฟร์วอล คือในขณะที่ไฟร์วอลทำหน้าที่กรองแพ็คเก็ต แต่ NIDS จะทำหน้าที่ตรวจสอบแพ็คเก็ต หรือถ้าจะเปรียบว่าระบบคอมพิวเตอร์คือบ้านหลังหนึ่ง ไฟร์วอลจะทำหน้าที่เป็นเซฟที่จะเปิดให้เฉพาะผู้ที่ได้รับอนุญาต ในขณะที่ NIDS จะทำหน้าที่เป็นอุปกรณ์ตรวจจับการบุกรุก ซึ่งหากขโมยเข้ามาในบ้าน ไฟร์วอลจะไม่ทราบว่าขโมยเข้ามา หากเซฟนั้นสร้างมาดีพอสิ่งที่อยู่ในเซฟก็จะปลอดภัย แต่หากเซฟสร้างมาไม่ดี หรือขโมยมีความเก่งกาจมากๆ ก็อาจจะเปิดเซฟออกมาได้ แต่หากบ้านของคุณมีอุปกรณ์ตรวจจับการบุกรุกแล้วคุณก็จะทราบได้ก่อนว่าขณะนี้เซฟกำลังถูกแกะอยู่ ซึ่งทำให้คุณสามารถหาทางรับมือหรือหาทางป้องกันได้อย่างทันท่วงที นี่คือหน้าที่ของ NIDS ซึ่งจะเป็นเครื่องมือที่ช่วยเสริมการทำงานของไฟร์วอลได้เป็นอย่างดี โดยไฟร์วอลจะเน้นการทำงานแบบตั้งรับ แต่ NIDS จะเน้นการทำงานในเชิงรุก ซึ่งจะทำให้ระบบมีความปลอดภัยสูงขึ้น

โปรแกรม NIDS ถือได้ว่าเป็นโปรแกรมที่มีความซับซ้อนมาก ทั้งนี้เนื่องจากโปรแกรม NIDS ที่ดีจะต้องเป็นโปรแกรมที่มีความฉลาด เพราะจะต้องตรวจสอบแพ็คเก็ตที่วิ่งไปมาในระบบเครือข่าย เพื่อหาแพ็คเก็ตที่มีลักษณะไม่ปรกติ นอกจากนั้นยังจะต้องวิเคราะห์อีกด้วยว่า ความไม่ปรกตินั้นเป็นความพยายามที่จะเจาะระบบหรือไม่

2.4.1 วิธีการตรวจจับผู้บุกรุกของ NIDS

- วิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก (Misuse Intrusion Detection)

วิธีนี้ทำการตรวจจับการบุกรุกที่อยู่ในรูปแบบที่รู้จัก หลักสำคัญของการตรวจจับวิธีนี้คือต้องระบุถึงสัญญาณการเกิดการบุกรุกทั้งหมดที่เป็นไปได้



รูปที่ 2.2 การทำงานของระบบตรวจจับผู้บุกรุกโดยวิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก

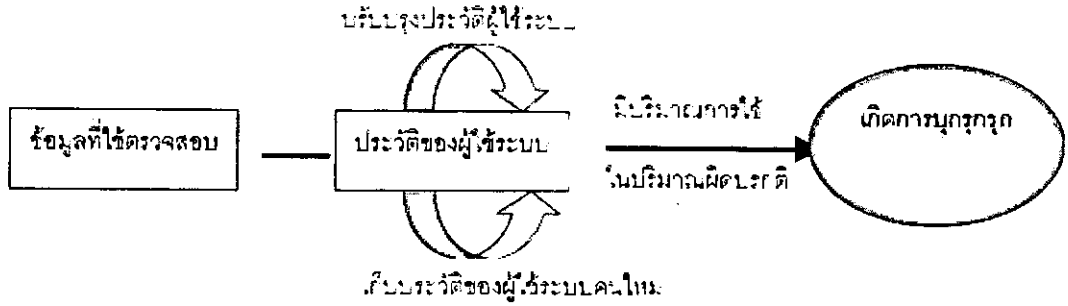
จากรูป ข้อมูลที่ใช้ตรวจสอบถูกนำมาเปรียบเทียบกับกฎ (Rules) ที่มีอยู่ ซึ่งมีการนำข้อมูลทางเวลาเข้ามาพิจารณาด้วย การตรวจจับการบุกรุกโดยวิธีนี้สามารถมีการแก้ไขกฎหรือเพิ่มกฎได้ ในระบบที่เป็นปัญญาประดิษฐ์ ระบบอาจทำการแก้ไขกฎหรือเพิ่มกฎได้ด้วยตัวเอง

จุดอ่อนของวิธีการเปรียบเทียบพฤติกรรมผู้ใช้กับวิธีการบุกรุกที่ระบบรู้จักนั้นคือ

1. ประสิทธิภาพในการตรวจจับผู้บุกรุกนั้นจะขึ้นอยู่กับการเลือกใช้กฎที่ใช้เปรียบเทียบ
2. หากตั้งกฎได้ไม่รัดกุมพอ เมื่อผู้ใช้ปรกติทำพฤติกรรมที่ตรงกับกฎของระบบตรวจจับผู้บุกรุกโดยไม่ได้ตั้งใจระบบตรวจจับอาจเข้าใจผิดว่าผู้ใช้นั้นๆ เป็นผู้บุกรุกได้
3. รูปแบบการบุกรุกที่ไม่รู้จักจะไม่สามารถถูกตรวจพบได้

- วิธีตรวจสอบการใช้งานทรัพยากรระบบที่ผิดปรกติ (Anomaly Intrusion Detection)

วิธีตรวจจับการบุกรุกโดยตรวจสอบการใช้งานทรัพยากรระบบที่ผิดปรกตินี้ตั้งอยู่บนสมมติฐานที่ว่าการกระทำใด ๆ ที่เป็นการบุกรุกจะต้องมีการใช้งานทรัพยากรระบบอย่างผิดปรกติ หมายความว่าในการตรวจจับวิธีนี้ต้องมีการเก็บพฤติกรรมปรกติของผู้ใช้ระบบไว้ เพื่อเปรียบเทียบว่าพฤติกรรมใดเป็นพฤติกรรมที่ผิดปรกติ การเปรียบเทียบจะใช้สถิติศาสตร์เข้าช่วย ดังแสดงในรูป



รูปที่ 2.2 การทำงานของการตรวจจับผู้บุกรุกโดยวิธีตรวจสอบการใช้งานระบบที่ผิดปรกติ

ข้อมูลที่ใช้ตรวจสอบ (Audit data) ซึ่งเก็บบันทึกโดยระบบจะถูกนำมาปรับปรุงไฟล์ประวัติ (Profile) ของผู้ใช้ พร้อมกันนั้นจะนำข้อมูลนี้มาเปรียบเทียบกับสถิติการใช้งานของผู้ใช้แต่ละคน หากพบว่ามีค่าผิดไปจากค่าปรกติก็จะระบุว่าอยู่ในสถานะที่มีการบุกรุกเกิดขึ้น นอกจากนี้ถ้ามีการตรวจสอบโดยใช้ข้อมูลใหม่ ๆ ก็สามารถเพิ่มในไฟล์ประวัติได้

ในการตรวจจับของ NIDS ส่วนใหญ่ เนื่องจากการตรวจจับเป็นการตรวจจับทางเครือข่าย ซึ่งข้อมูลจะอยู่ในรูปของแพ็กเก็ต จึงมักจะใช้วิธีการ Misuse กล่าวคือ ตรวจสอบรูปแบบการโจมตีโดยตรง โดยอาจเรียกรูปแบบการโจมตีว่า Pattern Attack หรือบางครั้งอาจเรียกว่า Signature โดยตัวอย่างข้างล่างนี้เป็นรูปแบบของการตรวจสอบการโจมตีของ Nimda (เป็นหนึ่งในหลาย ๆ บรรทัดของสคริปต์ที่ใช้โจมตี)

GET /script/...%c%af../winnt/system32/cmd.exe?/c+dir

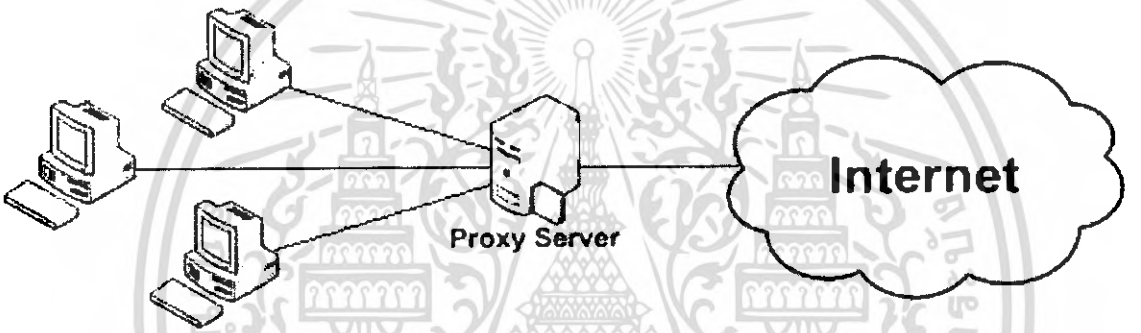
โดย %c%af เป็นข้อมูลที่เป็น Unicode ที่มีความหมายเท่ากับเครื่องหมาย Slash ดังนั้นคำสั่งข้างต้นจึงเป็นคำสั่งที่สั่งให้วิ่งไปที่Root Directory แล้วเรียกโปรแกรมใน /winnt/system32/cmd.exe เพื่อโจมตีจุดอ่อนของระบบ ดังนั้นเราจึงสามารถสร้างรูปแบบการตรวจสอบได้ว่าข้อมูลในแพ็กเก็ตนี้เป็นการโจมตีของ Nimda ซึ่งข้างต้นนี้เป็นรูปแบบการโจมตีแบบง่าย ๆ โดยในการโจมตีบางครั้งอาจมีความซับซ้อนมากกว่านี้ ซึ่งรูปแบบของการตรวจสอบการโจมตีก็จะต้องซับซ้อนมากขึ้นตามไปด้วย

การใช้วิธีตรวจจับการโจมตีโดย Signature นี้มีข้อดีที่มีความแม่นยำสูง และสามารถ อัพเดตไฟล์ Signature ให้มีความทันสมัยได้เรื่อย ๆ ซึ่งต่างจากวิธีการ Anomaly ซึ่งไม่แม่นยำ อย่างไรก็ตาม เนื่องจากการสร้างรูปแบบให้มีความจำเพาะเจาะจงลงไปถึงการโจมตีเป็นเรื่องยาก ดังนั้นในบางครั้งอาจมีการตรวจพบว่าเป็นการโจมตีจากข้อมูลทั่วไป ซึ่งจะเรียกการตรวจจับที่ผิดพลาดในลักษณะนี้ว่า False Negative และในบางครั้งเมื่อผู้โจมตีมีการปรับเปลี่ยนสคริปต์การโจมตีเพียงเล็กน้อยก็สามารถผ่านการตรวจสอบไปได้ ซึ่งจะเรียกกรณีแบบนี้ว่า False Positive

2.5 พร็อกซี่

2.5.1 เว็บพร็อกซี่ (Regular Webproxy)

เว็บพร็อกซี่มีลักษณะการใช้งานคือ การอนุญาตให้เครื่องไคลเอ็นต์ภายในเครือข่ายสามารถติดต่อกับอินเตอร์เน็ตได้โดยผ่านเว็บพร็อกซี่ โดยเว็บพร็อกซี่จะรอรับการร้องขอจากเครื่องไคลเอ็นต์ภายในเครือข่าย แล้วทำการส่งต่อการร้องขอนี้ไปยังเครื่องเซิร์ฟเวอร์ จากนั้นจึงรอรับการตอบกลับมาของเซิร์ฟเวอร์แล้วทำการส่งต่อการตอบรับนั้นให้กับไคลเอ็นต์ ซึ่งเว็บพร็อกซี่นี้สามารถทำการแคชข้อมูล คือการเก็บสำเนาข้อมูลจากอินเตอร์เน็ตมาไว้ที่เว็บพร็อกซี่ ทำให้เว็บพร็อกซี่นั้นไม่ต้องทำการร้องขอข้อมูลที่เคยทำการการร้องขอไปแล้วซ้ำ สามารถที่จะส่งข้อมูลเหล่านี้ให้กับเครื่องไคลเอ็นต์ที่ทำการร้องขอข้อมูลได้ทันที ทำให้การใช้งานอินเตอร์เน็ตนั้นทำได้รวดเร็วมากขึ้น



รูปที่ 2.3 แสดงตำแหน่งของเว็บพ็อกซี่ในเครือข่าย

ความสามารถและการใช้งานเว็บพร็อกซี่

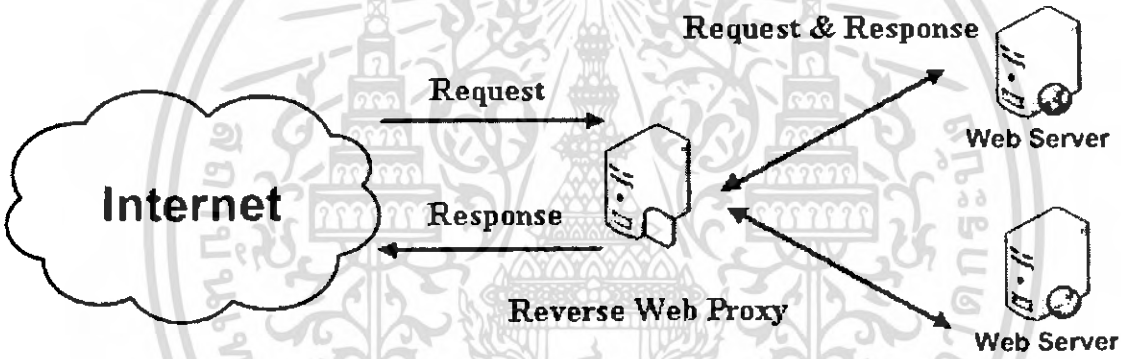
1. อนุญาตและจำกัดการใช้งานอินเตอร์เน็ตของเครื่องไคลเอนต์โดยวิธีการตรวจสอบไอพีแอดเดรส
2. แคชข้อมูลจากภายนอกให้เครื่องภายในเครือข่ายที่ต้องการข้อมูลที่เหมือนกันเรียกใช้งานได้เร็วขึ้น สามารถควบคุมให้การเข้าใช้งานอินเตอร์เน็ตและซับเน็ตที่ต้องการได้โดยกำหนด URL เป็นหลัก
3. อนุญาตหรือปฏิเสธการร้องขอจากไคลเอนต์โดยมีรูปแบบเงื่อนไขในการตัดสิน
4. กำหนด URL ที่จะใช้ในการกรองหรืออนุญาตให้เรียกใช้ผ่านเครื่องเซิร์ฟเวอร์ได้

2.5.2 รีเวิร์สเว็บพร็อกซี่ (Reverse Webproxy)

รีเวิร์สเว็บพร็อกซี่ เป็นพร็อกซี่ที่ทำงานบนฝั่งเซิร์ฟเวอร์แทนที่จะทำงานบนฝั่งไคลเอ็นต์จึงเรียกว่าเป็นการทำงานแบบรีเวิร์ส รีเวิร์สเว็บพร็อกซี่เป็นแอปพลิเคชั่นพร็อกซี่สำหรับเซิร์ฟเวอร์ที่ใช้โพรโตคอล HTTP รีเวิร์สเว็บพร็อกซี่จะทำงานอยู่ระหว่างไคลเอ็นต์กับเว็บเซิร์ฟเวอร์ทำหน้าที่

เป็นเกตเวย์ของเว็บเซิร์ฟเวอร์ หรือ ไอพีแอดเดรสที่ใช้สำหรับรับการร้องขอจากภายนอก ทำให้ไคลเอ็นต์เห็นรีเวิร์สเว็บพร็อกซี่เป็นเครื่องเว็บเซิร์ฟเวอร์ และทำหน้าที่ดูแลทราฟฟิกก่อนที่จะเข้าถึงเว็บเซิร์ฟเวอร์ โดยตรวจสอบการร้องขอที่ส่งเข้ามานั้นว่าเคยมีการแคชไว้หรือไม่ถ้ามีอยู่ในแคชก็จะทำการตอบกลับข้อมูลในแคชให้กับไคลเอ็นต์ ทำให้เว็บเซิร์ฟเวอร์ไม่ต้องสร้างข้อมูลตอบกลับหรือผลลัพธ์ทุกครั้งที่ได้รับการร้องขอ เว้นแต่ข้อมูลประเภท Dynamic ที่จะต้องมีการส่งการร้องขอไปยังเว็บเซิร์ฟเวอร์เพื่อประมวลผลทุกครั้ง โดยก่อนใช้งานรีเวิร์สเว็บพร็อกซี่ต้องมีการกำหนด prefix mapping ให้แก่รีเวิร์สเว็บพร็อกซี่ก่อนซึ่ง prefix mapping มี 2 ชนิด คือ

1) regular mapping ใช้สำหรับบอกรีเวิร์สเว็บพร็อกซี่ว่า URL prefix ไหนถูกแทนที่และบอก URL ปลายทางจริง
2) reverse mapping เป็นส่วนแปลง URL prefix ไปเป็น URL ของรีเวิร์สเว็บพร็อกซี่ซึ่งเป็นชื่อเว็บเซิร์ฟเวอร์จริงที่ใช้ในการติดต่อสื่อสารกับไคลเอนต์

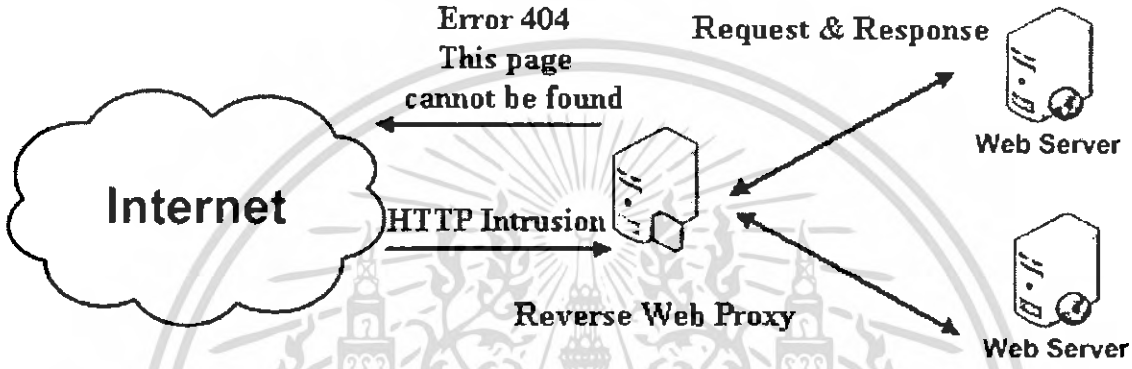


รูปที่ 2.4 การทำงานของรีเวิร์สเว็บพร็อกซี่

ในการใช้งานรีเวิร์สเว็บพร็อกซี่นั้นจะมีการใช้งานร่วมกับไฟร์วอลโดยมีการกำหนดกฎให้มีเพียงแค่ไอพีแอดเดรสของรีเวิร์สเว็บพร็อกซี่เท่านั้นที่สามารถติดต่อกับเว็บเซิร์ฟเวอร์ได้ ซึ่งหากมีการติดต่อสื่อสารเป็นการร้องขอจากไคลเอ็นต์มายัง URL ของรีเวิร์สเว็บพร็อกซี่ซึ่งเป็นทางที่ใช้ในการติดต่อกับเว็บเซิร์ฟเวอร์ รีเวิร์สเว็บพร็อกซี่ก็จะทำการตรวจสอบการร้องขอนั้นกับ prefix mapping ว่าจะต้องส่งการร้องขอนั้นให้กับเว็บเซิร์ฟเวอร์ใด จากนั้นจึงจะส่งการร้องขอนั้นไปยังไฟร์วอลเพื่อทำการส่งการร้องขอนั้นให้กับเว็บเซิร์ฟเวอร์ ทางฝั่งของเว็บเซิร์ฟเวอร์เมื่อประมวลผลของการร้องขอที่ได้รับจากไคลเอ็นต์เสร็จแล้ว จะทำการส่งผลลัพธ์กลับไปยังไคลเอ็นต์ โดยจะต้องทำการส่งผ่านไปยังรีเวิร์สเว็บพร็อกซี่ก่อนเช่นกัน ซึ่งเมื่อรีเวิร์สเว็บพร็อกซี่ได้รับผลลัพธ์ที่จะส่งกลับไปยังไคลเอ็นต์ ก็จะนำไปตรวจสอบกับ reverse mapping และทำการเปลี่ยน URL และ HTTP header กลับเป็น URL ที่ใช้ในการติดต่อกับไคลเอ็นต์ ก่อนที่จะส่งผลลัพธ์นั้นกลับไปยังไคลเอ็นต์ ซึ่งไคลเอ็นต์จะมองเห็นว่าการร้องขอนั้นถูกตอบรับโดยรีเวิร์สเว็บพร็อกซี่ แต่แท้จริงแล้วถูกตอบโดยเว็บเซิร์ฟเวอร์

2.5.3 รีเวิร์สเว็บพร็อกซี่แบบปลอดภัย (Secure Reverse Web Proxy)

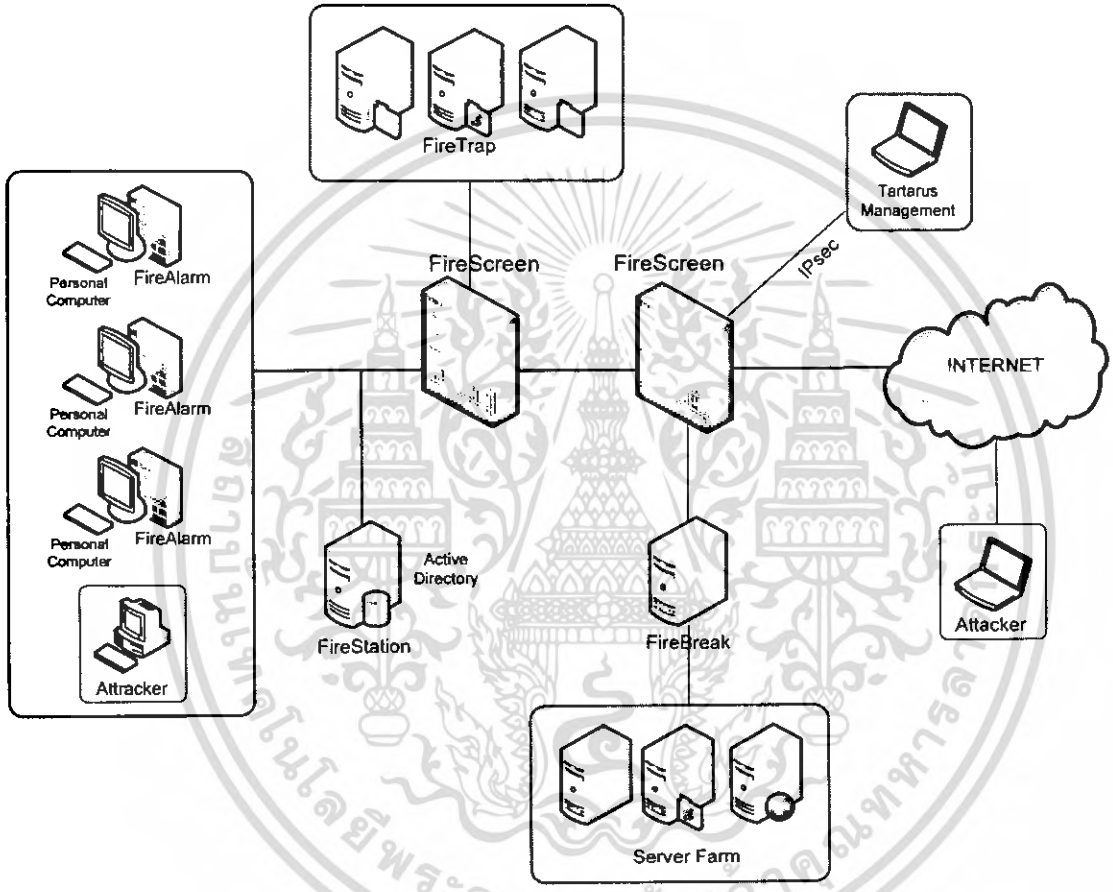
เป็นการใช้งานรีเวิร์สเว็บพร็อกซี่ให้มีความปลอดภัยในการใช้งาน โดยทำการตรวจสอบการร้องขอที่เข้ามาก่อนที่จะส่งต่อให้กับเว็บเซิร์ฟเวอร์ ว่าการร้องขอนั้นตรงกับรูปแบบการโจมตีหรือไม่ หากพบว่ามีการร้องขอที่ตรงกับรูปแบบการโจมตีก็จะทำการสกัดการร้องขอนั้นไว้ โดยอาจจะทำการเก็บบันทึกลงล็อกไฟล์และทำการส่งค่า page ที่เหมือน page ปรกติเช่น page Error 404 This Page can not be found เป็นการตอบสนองที่เรียกว่า Silent Killer Operation



รูปที่ 2.5 การทำงานของรีเวิร์สเว็บพร็อกซี่แบบปลอดภัย

บทที่ 3
การออกแบบและพัฒนา

3.1 โครงสร้างของโครงงาน



รูปที่ 3.1 รูปภาพแสดงโครงสร้างของชุดโปรแกรมรักษาความปลอดภัยเครือข่าย

จากภาพเป็นลักษณะโครงสร้างโดยรวมของชุดโปรแกรมรักษาความปลอดภัยเครือข่ายซึ่งมีการทำงานที่ติดต่อทั้งกับส่วนของผู้ใช้ที่มาจากอินเตอร์เน็ตและผู้ใช้ที่อยู่ภายในองค์กร และเป็นชุดโปรแกรมที่สามารถดักกรองแพ็คเก็ตที่เข้ามาในระบบพร้อมทั้งดักจับแยกแยะผู้บุกรุกออกจากผู้ใช้งานจริงเพื่อทำการศึกษาพฤติกรรมของผู้บุกรุกที่ทำกับระบบได้ต่อไป

3.2 การออกแบบและพัฒนาซอฟแวร์

โครงสร้างของชุดโปรแกรมจะแบ่งออกเป็น 7 ส่วนใหญ่ ๆ ดังนี้

3.2.1 ไฟร์สเตชั่น (FireStation)

เป็นโปรแกรมที่ทำงานอยู่บนระบบปฏิบัติการวินโดวส์เซิร์ฟเวอร์ 2003 ทำหน้าที่ในการกำหนดกฎของเกตเวย์ไฟร์วอล (FireScreen) และเพอร์ซันนอลไฟร์วอล (FireAlarm) ซึ่งกฎของเพอร์ซันนอลไฟร์วอลสามารถที่จะกำหนดกฎให้กับกลุ่มผู้ใช้และผู้ใช้แต่ละคนได้ นอกจากนี้ยังทำหน้าที่รับรายละเอียดการโจมตีจากเครื่องต่าง ๆ ที่อยู่ในระบบ ซึ่งเป็นข้อมูลจากส่วนไฟร์อลาร์ม (FireAlarm) และ ไฟร์สกรีน (FireScreen) ซึ่งข้อมูลเหล่านี้ ได้แก่ ข้อมูลการ Allow , Deny หรือ Redirect แพ็คเก็ตที่ผ่านเข้ามายังส่วนไฟร์สกรีนและไฟร์อลาร์ม รวมถึงการบุกรุกที่ตรวจจับได้จากไฟร์อลาร์ม พร้อมทั้งจัดการเกี่ยวกับการนำล็อกไฟล์จากฐานข้อมูลขึ้นมาแสดงผล

ไฟร์สเตชั่นจะพัฒนาบนระบบปฏิบัติการวินโดวส์โดยใช้ความสามารถของเทคโนโลยีที่ชื่อแอ็คทีฟไดเร็คทอรีในการเก็บกฎของโปรแกรมส่วนต่าง ๆ ภายในระบบ และใช้เครื่องมือที่พัฒนา คือ Microsoft Visual C++ 6.0

3.2.2 ไฟร์อลาร์ม (FireAlarm)

ทำงานบนระบบปฏิบัติการวินโดวส์ XP โดยจะทำหน้าที่ในการป้องกันเครื่องคอมพิวเตอร์ตามกฎที่ได้รับมาจากไฟร์สเตชั่น ส่วนระบบตรวจจับผู้บุกรุกจะทำหน้าที่ตรวจจับการโจมตีและทำการแจ้งเตือนโดยการแสดงรูปแบบการโจมตีไว้ในล็อกไฟล์ และสามารถทำการส่งล็อกไฟล์นี้ไปยังเซิร์ฟเวอร์กลางเพื่อแจ้งให้ผู้ดูแลระบบได้ทราบ นอกจากนี้ยังสามารถทำงานแบบวินโดวส์เซอร์วิส (Windows Service) ทำให้ป้องกันเครื่องของผู้ใช้ได้แม้ว่ายังไม่ได้ทำการล็อกอินเข้าระบบก็ตาม

ไฟร์อลาร์มจะพัฒนาบนระบบปฏิบัติการวินโดวส์ โดยใช้เครื่องมือที่พัฒนา คือ Microsoft Visual C++ 6.0

ส่วนประกอบต่าง ๆ ของไฟร์อลาร์ม มีดังนี้

- ในส่วนของการกรองแพ็คเก็ต จะใช้ Firewall Hook Driver ซึ่งเป็นการนำ API (Application Program Interface) ที่มีใน Microsoft SDK มาใช้ในการพัฒนาโปรแกรม โดย API นี้จะทำงานในส่วนของเคอร์เนลโหมด
- ในการดักจับแพ็คเก็ตจะใช้ความสามารถของ WinPcap ดักจับแล้วนำมาวิเคราะห์
- ส่วนที่ใช้ในการับกฎจากไฟร์สเตชั่น จะใช้ LDAP ในการติดต่อกับแอคทีฟไดเรคทอรี

3.2.3 ไฟร์สกรีน (FireScreen)

เป็นการรวมความสามารถของส่วนไฟร์สกรีนจากโครงงาน ISAG Firewall Suite 2548 และความสามารถในส่วนเกตเวย์ฮันนี่วอล จากโครงงาน ISAG Honeypot Suite 2548 เข้าไว้ด้วยกัน

โดยจะพัฒนาและออกแบบให้เป็นเกตเวย์ไฟร์วอลทำหน้าที่ป้องกันการโจมตีตามกฎที่รับมาจากเครื่องไฟร์สเตชั่นซึ่งเก็บไว้ในแอคทีฟไดเรคทอรีดาต้าเบสได้ โดยสามารถเข้าถึงผ่านทางโพรโตคอล LDAP โดยอาศัยไลบรารี่ OpenLDAP ซึ่งจะทำการตรวจสอบแพ็คเก็ตที่ผ่านเข้ามาในระบบเครือข่ายว่าเป็นไปตามกฎที่ตั้งไว้หรือไม่ แต่ถ้าไม่ตรงกับกฎใด ๆ เลยก็จะทำการตรวจสอบโดยอาศัยระบบตรวจจับผู้บุกรุก เพื่อทำการจำแนกผู้บุกรุกออกจากผู้ใช้ปรกติ ถ้าเข้าข่ายผู้บุกรุกก็จะทำการเปลี่ยนให้ผู้บุกรุกไปใช้งานในส่วนของไฟร์แทรป (FireTrap) แทน

ในการจำแนกผู้บุกรุกนั้น แพ็คเก็ตที่ไม่ตรงกับกฎใด ๆ ที่รับมาจากไฟร์สเตชั่นเลยจะถูกส่งมาตรวจสอบว่ามีพฤติกรรมเข้าข่ายเป็นผู้บุกรุกหรือไม่ โดยใช้กฎของโปรแกรมตรวจจับผู้บุกรุก (Snort_inline) ซึ่งเป็นเครื่องมือประเภท IDS (Intrution Detection System) โดยหากว่าการเชื่อมต่อใดมีลักษณะตรงตามกฎของโปรแกรมตรวจจับผู้บุกรุก จะถือว่าเจ้าของการเชื่อมต่อนั้นมีพฤติกรรมเข้าข่ายเป็นผู้บุกรุก และระบบจะทำการเปลี่ยนทิศทางการเชื่อมต่อดังกล่าวไปยังส่วนไฟร์แทรป (FireTrap) เพื่อทำการบันทึกพฤติกรรมของผู้บุกรุกที่กระทำต่อระบบต่อไป

3.2.4 ไฟร์เบรก (FireBreak)

ออกแบบให้เป็นโปรแกรมซีเคียวรีเวิร์สเว็บพร็อกซี่สำหรับเพิ่มความปลอดภัยให้กับเว็บเซิร์ฟเวอร์ซึ่งอยู่ภายในโซน DMZ มีความสามารถตรวจจับผู้บุกรุกในระดับแอปพลิเคชั่น และเมื่อมีการบุกรุกจะส่งข้อมูลการบุกรุกกลับไปที่เครื่องแม่ข่าย ในการพัฒนาจะใช้โปรแกรม Squid-cache ในการใช้งานเป็นรีเวิร์สเว็บพร็อกซี่ทั่ว ๆ ไป

3.2.5 ไฟร์แทรป (FireTrap)

เป็นระบบจำลองซึ่งแพ็คเก็ตที่ถูกส่วนไฟร์สกรีนจำแนกว่าเข้าข่ายผู้รุกจะถูกส่งมาที่ส่วนนี้ และทำการหลอกล่อให้ผู้บุกรุกเข้าใจผิดว่าเป็นเครื่องที่ให้บริการจริง โดยจะจำลองการบริการต่าง ๆ ให้เหมือนจริงมากที่สุด พร้อมทั้งเปิดบริการหรือช่องโหว่ เพื่อให้ผู้บุกรุกหลงเชื่อและแสดงพฤติกรรมในการบุกรุกต่าง ๆ ออกมาซึ่งพฤติกรรมเหล่านั้นจะถูกบันทึกจะส่งไปเก็บไว้ยังระบบล็อกมอนิเตอร์เพื่อเก็บบันทึกและนำไปศึกษาต่อไป นอกจากนี้จะมีการตรวจสอบสภาพและความเสียหายของเครื่องไฟร์แทรปตลอดเวลาจากโปรแกรม S2I ซึ่งอยู่ในส่วนของไฟร์สกรีน

3.2.6 ล็อกมอนิเตอร์ (Log Monitor)

ออกแบบให้รับรายละเอียดการโจมตีจากเครื่องลูกข่ายในระบบและจากภายนอกระบบ โดยล็อกไฟล์จากส่วนต่างๆ จะถูกนำมาแสดงผลซึ่งจะแบ่งการแสดงผลออกเป็นสองส่วนคือ ส่วนที่เป็นล็อกไฟล์ที่อยู่ในระบบแอ็คทีฟไดเร็คทอรี่และส่วนที่เป็นล็อกไฟล์ที่เก็บอยู่ในส่วนของดาต้าเบส Mysql

3.2.7 Tartarus Management (TM)

โปรแกรม Tartarus Management (TM) เป็นโปรแกรม Front-end ที่สร้างขึ้นเพื่อแก้ปัญหาความยุ่งยากในการดูแลและจัดการระบบจำแนกผู้บุกรุกที่อยู่ในส่วนไฟร์สกรีน การจัดการเครื่องกับดักของระบบไฟร์แทรป รวมถึงการแสดงล็อกไฟล์จากระบบตรวจจับผู้บุกรุก และล็อกไฟล์จากการบันทึกพฤติกรรมของผู้บุกรุกในเครื่องกับดัก ซึ่งในการทำงานร่วมกับระบบอื่น ๆ นั้นทำงานผ่านทางช่องทาง IPsec เพื่อความปลอดภัย

บทที่ 4
การพัฒนาระบบไฟร์สเตชั่น

4.1 แนวคิด

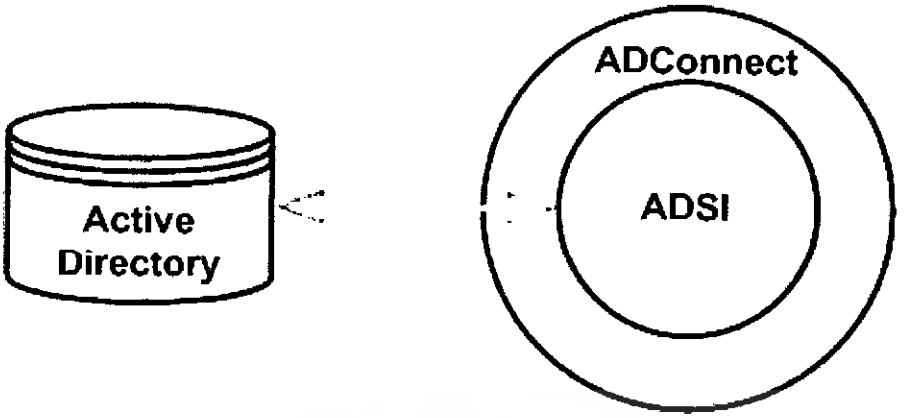
เป็นโปรแกรมที่ใช้ในการกำหนดกฎของเพอร์ซันนอลไฟร์วอล และเกตเวย์ไฟร์วอล โดยใช้เทคโนโลยีของแอ็คทีฟไดเร็คทอรีเซอร์วิส (Active Directory Service) ในการจัดเก็บกฎของเพอร์ซันนอลไฟร์วอลแต่ละเครื่องและกฎของเกตเวย์ไฟร์วอล เพื่อให้เครื่องลูกข่ายและเครื่องที่ทำหน้าที่เป็นเกตเวย์ไฟร์วอลที่มีสิทธิในการใช้งานสามารถเข้าถึงแอ็คทีฟไดเร็คทอรีเพื่อนำกฎไปใช้ในการป้องกันการบุกรุกได้ โดยข้อมูลกฎจะเก็บอยู่ในฐานข้อมูลของแอ็คทีฟไดเร็คทอรี ทำให้การวางตำแหน่งของเครื่องที่ควรจะติดตั้งโปรแกรมนี้คือ เครื่องที่ทำงานเป็นเครื่องแม่ข่ายหรือเครื่องเซิร์ฟเวอร์ที่เปิดใช้งานโดเมนคอนโทรลเลอร์

4.2 ขอบเขตความสามารถ

- สามารถพิสูจน์ผู้ใช้ที่มาขอใช้ทรัพยากรได้อย่างถูกต้องโดยใช้ความสามารถของแอ็คทีฟไดเร็คทอรี
- สามารถกระจายกฎและจัดเก็บได้อย่างมีระเบียบ

4.3 เทคโนโลยีที่ใช้ในการพัฒนาโปรแกรม

การติดต่อระหว่างไฟร์สเตชั่น ไฟร์อลาร์มและล็อกมอนิเตอร์ของไฟร์อลาร์มจะทำงานอยู่บนความสามารถของแอ็คทีฟไดเร็คทอรี (Active Directory) ของวินโดวส์ 2003 เซิร์ฟเวอร์โดยข้อมูลจะเก็บอยู่ในฐานข้อมูลของแอ็คทีฟไดเร็คทอรี (Active Directory Database) และเข้าถึงข้อมูลผ่าน ADSI (Active Directory Service Interfaces) ไปยังแอ็คทีฟไดเร็คทอรีเซอร์วิส (Active Directory Service) เพื่อเข้าถึงที่จัดเก็บ แสดงดังรูปที่ 4.1 ดังนั้นจึงต้องพัฒนาระบบไฟร์สเตชั่น ไฟร์อลาร์มและล็อกมอนิเตอร์ของไฟร์อลาร์มไปพร้อมๆกัน เพราะต้องมีการติดต่อกับแอ็คทีฟไดเร็คทอรีในกรณีเดียวกัน



รูปที่ 4.1 แสดงรูปแบบการเข้าถึง Active Directory Service

การพัฒนาโปรแกรมไฟร์สเตชั่น ไฟร์อลาร์ม และล็อกมอนิเตอร์ของไฟร์อลาร์มนี้จะมีการสร้างสกีมา(Schema) ซึ่งเป็นโครงสร้างฐานข้อมูลของแอ็คทีฟไดเร็คทอรีประกอบไปด้วย คอนเทนเนอร์คลาส และ แอตตริบิวต์ โดยการเข้าถึงข้อมูลเหล่านั้นจะมีการกำหนดหน้าที่การทำงานมาแล้ว ดังนั้นถ้าเราต้องการนำข้อมูลที่เราต้องการจัดเก็บหรือเข้าถึงรูปแบบใหม่เข้าไปจึงต้องมีการสร้างขึ้นมาใหม่ เพื่อให้เหมาะสมกับการทำงานของระบบ ไดเร็คทอรีเบสไฟร์วอล

ข้อมูลที่จัดเก็บประกอบด้วย ข้อมูลของกฎ ข้อมูลรายละเอียดการโจมตีและข้อมูลที่บอกการเปลี่ยนแปลงกฎโดยเพิ่มสกีมาที่เหมาะสมลงไปในฐานข้อมูลของแอ็คทีฟไดเร็คทอรีดังนี้

1. แอตทริบิวต์ รายละเอียดการเพิ่มจะเป็นดังตาราง ที่ 4.1

Common Name	OID	Syntax
firewallRule	1.2.840.113556.1.4.7000.142	Case Insensitive String
firewallLog	1.2.840.113556.1.4.7000.144	Case Insensitive String
update	1.2.840.113556.1.4.7000.146	Case Insensitive String

ตารางที่ 4.1 รายละเอียดของแอตทริบิวต์ที่เพิ่มในฐานข้อมูลของแอ็คทีฟไดเร็คทอรี

2. คลาสรายละเอียดการเพิ่มจะเป็นดังตารางที่ 4.2

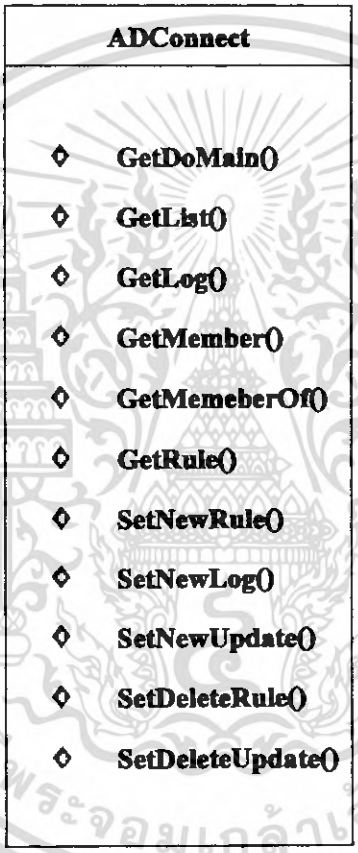
Common Name	OID	Syntax
ISAGFW	1.2.840.113556.1.4.7000.143	Auxiliary

ตารางที่ 4.2 รายละเอียดคลาสที่เพิ่มในฐานข้อมูลของแอ็คทีฟไดเร็คทอรี

โดยกำหนดให้แอตทริบิวต์ firewallRule เป็น Mandatory และ FirewallLog จะเป็น Auxiliary ใน คลาส ISAGFW

3. ผนวกสกีมาใหม่ เข้ากับ คลาส groups คลาส users และคลาส computers

ในการเข้าถึงข้อมูลที่อยู่ในยังแอ็คทีฟไดเร็คทอรีเซอร์วิส นั้นต้องใช้ ADSI ดังนั้นจึงมีการพัฒนาฟังก์ชั่นที่ใช้เข้าถึงและกำหนดการทำงานกับแอ็คทีฟไดเร็คทอรีได้อย่างง่ายขึ้น โดยการสร้างคลาส ADConnect ที่มีรายละเอียดฟังก์ชั่น ดังรูปที่ 4.2 โดยคลาส ADConnect จะใช้ในโปรแกรมไฟร์สเตชั่น ไฟร์อลาร์ม และล็อกมอนิเตอร์ของไฟร์อลาร์มที่พัฒนาขึ้นเพื่อติดต่อไปยังแอ็คทีฟไดเร็คทอรีเซอร์วิส



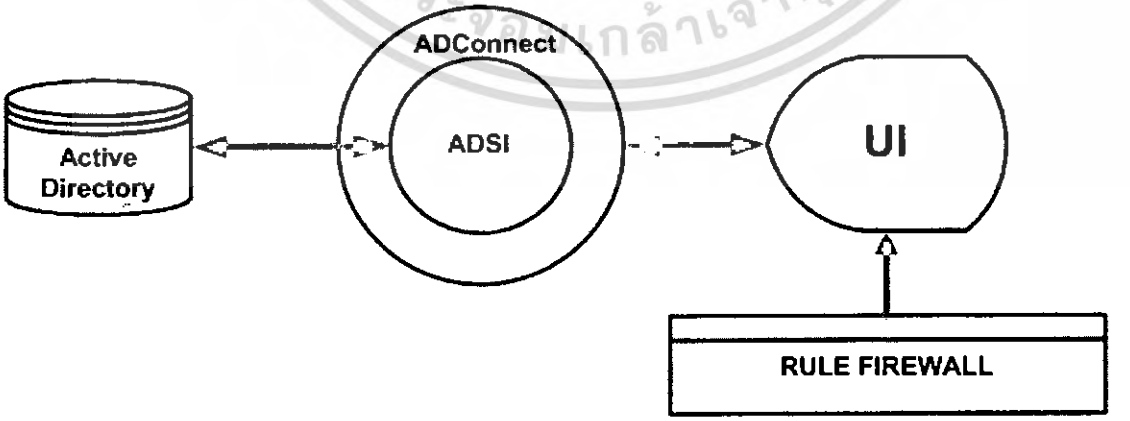
รูปที่ 4.2 รายละเอียดของคลาส ADConnect

Function	**การทำงาน**
GetDomain	หาชื่อเครื่องโดเมนที่ผู้ใช้สังกัดอยู่
GetList	อ่านรายละเอียดที่เป็นจุดอ้างอิง
GetLog	อ่านรายละเอียด Log
GetMember	อ่านรายชื่อ User จาก Group
GetMemberOf	อ่านรายชื่อ Group จาก User
GetRule	อ่านกฎที่มีอยู่จากแอ็คทีฟไดเร็คทอรี
SetNewRule	สร้างกฎ
SetNewLog	สร้าง Log
SetNewUpdate	บอกให้ไฟร์สกรีนทราบว่ากฎมีการเปลี่ยนแปลงโดยเซต flag เป็น YES
SetDeleteRule	ลบกฎที่ต้องการลบ
SetDeleteUpdate	ลบ flag ที่ใช้ในการบอกการเปลี่ยนแปลงของกฎ

ตารางที่ 4.3 รายละเอียดการทำงานของฟังก์ชั่นในคลาส ADConnect

4.4 การทำงานของโปรแกรม

ไฟร์สเตชั่น(FireStation) จะทำการอ่านข้อมูลผ่านคลาส ADConnect เพื่อเข้าถึงข้อมูลที่อยู่ในแอ็คทีฟไดเร็คทอรีเซอร์วิส (Active Directory Service) โดยจะเลือกแสดงตามกลุ่มผู้ใช้ที่ถูกเลือก และเมื่อมีการ เพิ่ม ลด และแก้ไขข้อมูลกฎที่อยู่ในฐานข้อมูล รูปที่ 4.3 จะแสดงการทำงานหลักของโปรแกรมไฟร์สเตชั่น (FireStation)



รูปที่ 4.3 แสดงโครงสร้างการทำงานของไฟร์สเตชั่น(FireStation)

บทที่ 5
การพัฒนาระบบไฟร์สกรีน

5.1 แนวคิด

ระบบไฟร์สกรีนในโครงงานนี้เป็นการรวมความสามารถของส่วนไฟร์สกรีนจากโครงงาน ISAG Firewall Suite 2548 และ และความสามารถในส่วนเกตเวย์ฮันนี่วอล จากโครงงาน ISAG Honeypot Suite 2548 เข้าไว้ด้วยกัน โดยจะพัฒนาและออกแบบให้เป็นเกตเวย์ไฟร์วอลทำหน้าที่ป้องกันการโจมตีได้ตามกฎที่รับมาจากเครื่องไฟร์สเตชั่นซึ่งเก็บไว้ในแอ็คทีฟไดเร็คทอรีดาต้าเบส โดยสามารถเข้าถึงผ่านทางโพรโตคอล LDAP โดยอาศัยไลบรารี่ OpenLDAP ซึ่งจะทำการตรวจสอบแพ็คเก็ตที่ผ่านเข้ามาในระบบเครือข่ายว่าเป็นไปตามกฎที่ตั้งไว้หรือไม่ แต่ถ้าไม่ตรงกับกฎใด ๆ เลย แพ็คเก็ตจะถูกตรวจสอบโดยอาศัยระบบตรวจจับผู้บุกรุกอีกทีหนึ่ง เพื่อทำการจำแนกผู้บุกรุกออกจากผู้ใช้ปรกติ ถ้าแพ็คเก็ตนั้นเข้าข่ายเป็นผู้บุกรุก ระบบจะทำการเปลี่ยนให้ผู้บุกรุกไปใช้งานในส่วนของไฟร์แทรป (FireTrap) แทน และส่งข้อมูลการบุกรุกเกี่ยวกับการบุกรุกและการกระทำของผู้บุกรุกนั้น ๆ ไปยังส่วนระบบล็อกมอนิเตอร์ต่อไป

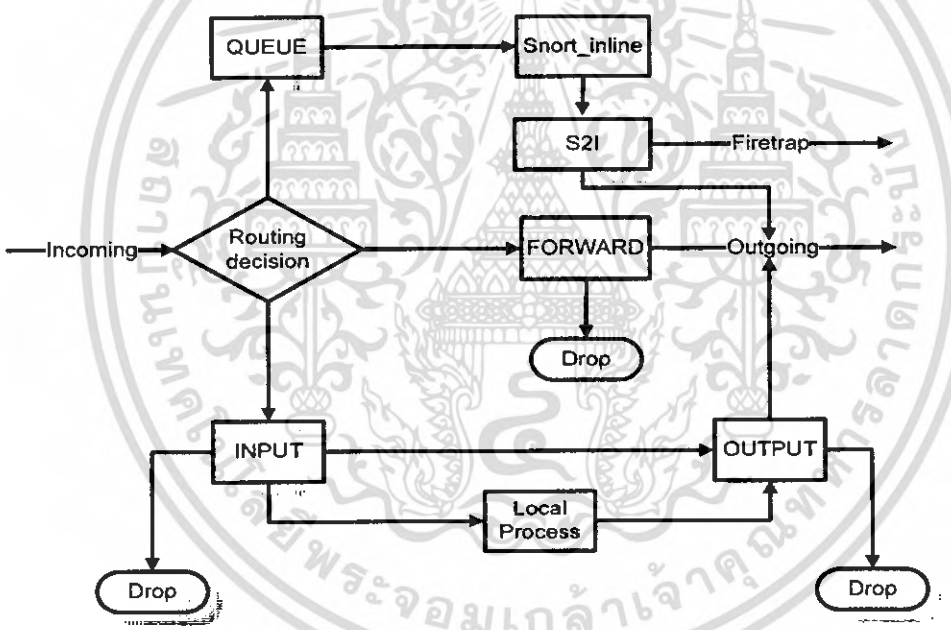
5.2 ขอบเขตความสามารถ

ไฟร์สกรีนจะทำหน้าที่เป็นเกตเวย์ไฟร์วอลของระบบทั้งหมด ซึ่งในการเชื่อมต่อทุก ๆ ส่วนจะต้องผ่านทางเกตเวย์นี้ ทำให้ไฟร์สกรีนสามารถควบคุมการเชื่อมต่อเกือบทั้งหมดของระบบไว้ได้ โดยในการดูแลการเข้าออกของแพ็คเก็ต จะทำการจำแนกและตรวจสอบแพ็คเก็ตหรือการเชื่อมต่อต่างๆ ที่เข้ามาในระบบตามกฎของไฟร์วอลที่กำหนดโดยผู้ดูแลระบบซึ่งรับมาจากไฟร์สเตชั่น และกระทำ Action ต่าง ๆ กับแพ็คเก็ตนั้นตามที่ได้กำหนดไว้ แต่สำหรับแพ็คเก็ตที่ไม่ตรงกับกฎใด ๆ ของไฟร์วอลจะถูกส่งมาตรวจสอบว่ามีพฤติกรรมเข้าข่ายเป็นผู้บุกรุกหรือไม่ โดยใช้ Signature ของโปรแกรมตรวจจับผู้บุกรุก เช่น สนอร์ตอินไลน์ (Snort_inline) ซึ่งเป็นเครื่องมือประเภท IDS (Intrusion Detection System) โดยหากว่าการเชื่อมต่อใดมีลักษณะตรงตาม Signature ของโปรแกรมตรวจจับผู้บุกรุกจะถือว่าเจ้าของการเชื่อมต่อนั้นมีพฤติกรรมเข้าข่ายเป็นผู้บุกรุก และจะต้องทำการเปลี่ยนทิศทางการเชื่อมต่อดังกล่าวไปยังระบบจำลองที่ชื่อ ไฟร์แทรป (FireTrap) เพื่อทำการบันทึกพฤติกรรมของผู้บุกรุกที่กระทำต่อระบบต่อไป

แต่ในการทำงานเพียงลำพังของระบบตรวจจับผู้บุกรุกนั้นไม่เพียงพอที่จะจัดการให้ผู้บุกรุกเข้าสู่ระบบจำลองไฟร์แทรปที่จัดเตรียมไว้ได้ เนื่องจากระบบตรวจจับผู้บุกรุกนั้นไม่มีความสามารถเพียงพอในการสร้างเส้นทางการเชื่อมต่อ ซึ่งความสามารถนี้เป็นความสามารถของไฟร์วอล ที่มีชื่อเรียกว่า ไอพีเทเบิล (Iptables) ซึ่งจะเป็นตัวจัดการการเชื่อมต่อสื่อสารเพื่อให้เป็นไปตามความ

ต้องการของผู้พัฒนา แต่ในการทำงานจริงนั้น ทั้งสองโปรแกรมนี้ไม่สามารถพูดคุยเพื่อควบคุมกันได้ เนื่องจากว่าไอพีเทเบิล เป็นโปรแกรมที่ทำงานในส่วนของเคอร์เนลโหมด (kernel mode) แต่ Sn สนอร์ตอินไลน์นั้นเป็นโปรแกรมที่ทำงานในส่วนยูสเซอร์โหมด (User mode) ดังนั้นจึงได้มีการพัฒนาตัวกลางที่ทำหน้าที่สื่อสารระหว่างระบบตรวจจับผู้บุกรุกกับไฟร์วอลไอพีเทเบิลขึ้นมาเพื่อเป็นตัวกลางที่จะอ่านค่าจากการแจ้งเตือนจากระบบตรวจจับผู้บุกรุก และนำค่าได้มานั้นแปลงเป็นคำสั่งของไอพีเทเบิลอย่างเหมาะสมเพื่อสั่งให้จัดส่งชิ้นข้อมูลไปยังปลายทางที่ถูกต้อง ซึ่งทางผู้พัฒนาในทำการพัฒนาโปรแกรม S2I (Snort command to Iptables) ขึ้นมาเพื่อทำหน้าที่ดังกล่าว

นอกจากนี้โปรแกรม S2I จะคอยตรวจสอบความอ่อนแอของเครื่องกับดักในส่วนของไฟร์แทรปอยู่ตลอดเวลาว่ามีความอ่อนแอหรือถูกเปลี่ยนแปลงโดยผู้บุกรุกหรือไม่ โดยถ้าถึงจุดที่เครื่องกับดักมีความเสียหายก็จะทำการหยุดให้บริการเครื่องนั้นโดยอัตโนมัติและจะทำการสั่งเครื่องกับดักเครื่องใหม่ขึ้นมาทำงานต่อ เพื่อป้องกันการใช้เครื่องกับดักเป็นฐานในการโจมตีระบบอื่น



รูปที่ 5.1 แผนผังการทำงานของระบบไฟร์สกรีน

5.3 เทคโนโลยีที่ใช้ในการพัฒนาโปรแกรม

5.3.1 ไอพีเทเบิล (Iptables)

ไอพีเทเบิลนั้นเป็นโปรแกรมไฟร์วอลแบบสเตทฟูลอินสเป็คชั่นที่ทำงานในเคอร์เนลโหมด โดยไอพีเทเบิลจะมีหน้าที่ในการจัดการเชื่อมต่อ และการอนุญาตหรือไม่อนุญาตให้ชิ้นข้อมูลต่างๆ สามารถผ่านเข้าออกระบบเครือข่ายได้ ในโครงงานนี้ได้นำไอพีเทเบิลมาใช้เพื่อการตรวจสอบชิ้นข้อมูลหรือการเชื่อมต่อต่าง ๆ ที่เข้ามาตามกฎที่ตั้งไว้จากไฟร์สเตชั่น โดยผู้ดูแลระบบ และทำการสร้างเส้นทางให้เหมาะสม นอกจากนี้ยังนำโมดูล ip_queue ของไอพีเทเบิลมาทำงานร่วมกับ

โปรแกรมสนอร์ตอินไลน์เพื่อตรวจสอบชิ้นข้อมูลที่ไม่ตรงตามกฎใด ๆ ของไฟร์วอลและทำการจำแนกชิ้นข้อมูลนั้น ๆ ว่ามีพฤติกรรมเข้าข่ายเป็นผู้บุกรุกหรือไม่

รูปแบบของคำสั่งในการใช้งานไอพีเทเบิล

iptables [table] <command> <match> <target/jump>

ไอพีเทเบิลมีรูปแบบการใช้งานดังข้างต้น โดยกฎที่เขียนขึ้นจะเป็นตัวบอกเคอร์เนลว่าให้กระทำอย่างไร (action) ในกรณีที่พบแพ็คเก็ตที่ตรงตามกฎที่ระบุไว้

- Command

  -A เพิ่มกฎใหม่ต่อท้าย chain (Append rule)

  -D ลบกฎ (Delete rule)

  -I เพิ่มกฎใหม่ใน chain (Insert rule)

  -R แทนที่กฎเดิมด้วยกฎใหม่ (Replace rule)

  -L แสดงกฎใน chain (ถ้าไม่ระบุ chain จะแสดงกฎใน filter table ทั้งสาม built-in chain)

  -F ลบกฎทั้งหมดใน chain ทิ้ง

  -Z ใช้ reset byte counter สำหรับทุกกฎใน chain ที่กำหนด

  -N ใช้สร้าง chain ใหม่

  -X ลบ chain ใช้กับ user-defined chain ที่ไม่มีกฎ แต่ไม่สามารถลบ built-in chain ได้

  -P เปลี่ยน default policy ของ chain ค่าที่ใช้กับcommand นี้คือ ACCCEPT, DROP

  -E ใช้เปลี่ยนชื่อ chain ใหม่

ซึ่งการใช้ command ข้างต้นนั้นสามารถใช้ร่วมกับออปชั่นบางอย่างได้ คือ

-v (--verbose) ใช้ร่วมกับ –L, -A, -I, -D, -R เพื่อให้แสดงจำนวน byte ที่ match กับ กฎออกมา (หน่วยที่แสดงออกมานั้นอาจจะเป็น K, M, G)

-x (--exact) ใช้ร่วมกับ –L และ –v เพื่อให้แสดงจำนวนแพ็คเก็ตและจำนวนของ byte ข้อมูลที่ match โดยไม่ให้แสดงผลในหน่วยของ K, M, G

-n (--numeric) ใช้ร่วมกับ –L เพื่อสั่งให้ไอพีเทเบิล แสดงข้อมูลหมายเลขไอพีแอดเดรส และ หมายเลขพอร์ตเป็นตัวเลขเท่านั้น

--line-numbers ใช้ร่วมกับ –L เพื่อแสดงเลขบรรทัดของกฎซึ่งตัวเลขที่แสดงนี้จะสามารถใช้ได้กับคำสั่งแทรกกฎ ที่ระบุเป็นลำดับที่ของกฎ

- Match

- การระบุหมายเลขไอพีแอดเดรสต้นทางและหมายเลขไอพีแอดเดรสปลายทาง

สามารถระบุไอพีแอดเดรสต้นทางของแพ็คเก็ตโดยใช้ -s หรือ –source หรือ --src และ ไอพีแอดเดรสปลายทางใช้ -d หรือ --destination หรือ --dst ในการระบุไอพีแอดเดรสนั้นสามารถทำได้ 3 แบบด้วยกันคือ

1. ใช้ชื่อเด็มแทน เช่น localhost หรือ www.ce.kmitl.ac.th
2. ระบุไอพีแอดเดรสโดยตรง เช่น 127.0.0.1 หรือ 161.246.4.7
3. ระบุเป็นกลุ่มของไอพีแอดเดรส เช่น 161.246.5.0/24 ซึ่งหมายถึงไอพีแอดเดรสตั้งแต่ 161.246.5.0 – 161.246.5.255 หรือ 161.246.5.0 /255.255.255.0 แทน 161.246.5.0 /24 ได้

- การทำ Inversion

ทำได้โดยใช้เครื่องหมาย ! นำหน้า argument ที่ต้องการ (! หมายถึง NOT) เช่น -p ! TCP

- การระบุโพรโตคอล

สามารถระบุโพรโตคอลที่ต้องการได้ดังนี้คือ TCP, UDP, ICMP และสามารถใช้ได้ทั้งตัวอักษรเล็กหรือใหญ่ (ใช้ได้ทั้ง tcp และ TCP) เช่น -p TCP หรือ -p ! tcp

- การระบุ interface

-i หรือ --in-interface ตามด้วยชื่ออินเทอร์เฟส ใช้เพื่อระบุอินเทอร์เฟสขาเข้าซึ่งหมายถึงว่าแพ็คเก็ตที่จะตรงกับกฎนี้ต้องเข้ามาจากอินเทอร์เฟสที่กำหนด เช่น -i eth0 หมายความว่า ทุกแพ็คเก็ตที่เข้ามาทาง eth0 จะตรงกับกฎ นี้

-o หรือ --out-interface ตามด้วยชื่อของอินเทอร์เฟส ใช้เพื่อระบุอินเทอร์เฟสขาออก ซึ่งหมายถึงว่าแพ็คเก็ตที่จะตรงกับกฎนี้ กำลังจะเดินทางผ่านอินเทอร์เฟสที่ระบุไว้ เช่น -o eth1 หรือ -o ! eth1 จะตรงกับกฎนี้

- fragment packet

ในการส่งข้อมูลผ่านเครือข่ายนั้นเป็นเรื่องปรกติที่จะเกิดการ แฟร็กเมนต์ (fragment) ของแพ็คเก็ตเนื่องจากขนาดของแพ็คเก็ตมีขนาดใหญ่เกินไปที่จะส่งไปในครั้งเดียว จำเป็นต้องมีการแบ่งแพ็คเก็ตออกเป็นหลายๆชิ้นทยอยส่งไป ซึ่งเรียกกันว่าการทำ fragmentation โดยเครื่องปลายทางจะทำหน้าที่ประกอบแพ็คเก็ตที่ผ่านการทำแฟร็กเมนต์ ให้มารวมกันเป็นแพ็คเก็ตที่สมบูรณ์ดังเดิม ข้อมูลของแฟร็กเมนต์แพ็คเก็ตจะมีเฮดเดอร์ที่สมบูรณ์แค่แพ็คเก็ตแรกเท่านั้น แพ็คเก็ตที่ตามมาจะมีเฮดเดอร์แค่บางส่วนคือ ไอพีแอดเดรส ไม่มีข้อมูลของโพรโตคอลแนบมาด้วย ดังนั้นการตรวจสอบข้อมูลเฮดเดอร์ของ TCP, UDP, ICMP จึงไม่สามารถทำได้ในแพ็คเก็ตที่สองเป็นต้นมา

โดยปรกติแล้วมักจะปล่อยให้แพ็คเก็ตที่ทำการแฟร็กเมนต์มาแล้วผ่านไป เนื่องจากถ้าสามารถ DROP แพ็คเก็ตตัวแรกได้แล้ว มันก็ไม่สามารถถูกประกอบที่เครื่องปลายทางได้ แต่ทั้งนี้แฟร็กเมนต์แพ็คเก็ต ที่ถูกปล่อยไปดังกล่าวอาจจะทำให้เครื่องที่ได้รับไม่สามารถทำงานต่อได้

- TCP extension

ถ้ามีการเรียกใช้ -p tcp ตัว TCP extension จะถูกโหลดมาใช้งานโดยอัตโนมัติ โดยมีออปชั่นให้เลือกใช้งานดังนี้

--tcp-flags mask flags : mask นั้นหมายถึง flag ที่ต้องการตรวจสอบ และ flag เป็นตัวที่บ่งชี้ว่า flag ใดต้องถูก set บ้าง

--source-port หรือ --sport สามารถใช้ได้ทั้งตัวเลขและตัวอักษร ระบุเป็นพอร์ตเดี่ยว หรือช่วงของพอร์ตได้

--destination-port หรือ --dport มีรูปแบบการใช้งานเช่นเดียวกันกับ –sport

- UDP extension

เช่นเดียวกับ TCP ตัว UDP extension มีออปชั่นให้เลือกใช้เพียงแค่ 2 อย่างเท่านั้นคือ --source-port (--sport) และ --destination-port (--dport) โดยต้องระบุ -p udp ด้วย

- ICMP extension

โดยการระบุ -p icmp ก็สามารถใช้งาน ICMP extension ได้ โดยมีออปชั่นให้เลือกคือ --icmp-type เช่น --icmp-type host-unreachable (หรือใช้เลข 3 แทนได้) นอกจากนี้ยังสามารถระบุ type/code ได้ เช่น 3/3 ซึ่งหมายถึง port unreachable

- Match Extension

เป็น netfilter package อื่นๆ รูปแบบการใช้งานให้ใช้ -m แล้วตามด้วย match ที่ต้องการ มีออปชั่นให้เลือกใช้งานดังต่อไปนี้

mac รูปแบบการใช้งาน: -m mac ใช้ตรวจสอบ source MAC address ว่าตรงกับค่าที่ระบุไว้หรือไม่ มีประโยชน์สำหรับ *PREROUTING และ INPUT chain*

limit รูปแบบการใช้งาน: -m limit ใช้เพื่อจำกัดจำนวนของการ match ที่อาจจะมากเกินไป เป็นประโยชน์สำหรับกฎที่วางไว้ตอนท้ายสุดของ chain (ใช้ร่วมกันกับ DROP policy) ซึ่งส่วนใหญ่เป็นกฎที่ใช้เก็บข้อมูลลงล็อกไฟล์ ซึ่งถ้าผู้บุกรุกส่งแพ็คเก็ตที่ไม่เข้าข่ายกฎใดๆ ใน chain จนกระทั่งมาถึงกฎที่ทำหน้าที่เก็บล็อกนี้ ถ้าแพ็คเก็ตที่เข้ามามีจำนวนมาก อาจจะทำให้ฮาร์ดดิสก์เต็มได้ ดังนั้นจึงต้องใช้จำกัดจำนวนในการเก็บข้อมูลลงล็อก ซึ่งมีออปชั่นให้ใช้งานดังนี้คือ

State Match รูปแบบการใช้งานคือ -m state หรือ --match state เป็นโมดูลที่ใช้ประโยชน์ได้เป็นอย่างดี มีออปชั่นให้ใช้งานดังนี้

NEW – แพ็คเก็ตที่เป็นตัวสร้างการเชื่อมต่อใหม่

ESTABLISHED – แพ็คเก็ตที่เกี่ยวข้องกันกับการเชื่อมต่อที่สร้างไว้แล้ว

RELATED – แพ็คเก็ตที่เกี่ยวข้องกับการเชื่อมต่อที่สร้างไว้แล้ว แต่ไม่ใช่ส่วนหนึ่งส่วนใดของการเชื่อมต่อนั้น

INVALID – เป็นแพ็คเก็ตที่ไม่เกี่ยวข้องกับส่วนอื่นเลย

- Target

เป็นส่วนที่ระบุ action ที่กระทำกับแพ็คเก็ตต่างๆ ที่ตรงกับกฎที่ได้ตั้งขึ้นซึ่งมี target ในแบบต่างๆดังนี้

- ACCEPT อนุญาตให้แพ็คเก็ตที่ตรงกับกฎนี้สามารถผ่านไปได้
- DROP แพ็คเก็ตที่ตรงกับกฎนี้ให้ทำการ Drop ทิ้งหรือไม่ให้ผ่าน
- LOG เป็นโมดูลที่มีความสามารถในการเก็บข้อมูลลงล็อก
- REJECT คล้ายกับ DROP แต่จะมีการส่ง ICMP port unreachable กลับไปยังผู้ส่ง
- RETURN ออกจากเชน (chain) ปัจจุบันกลับไปยังเชนที่ทำการเรียกมา
- QUEUE เป็นเชนพิเศษใช้สำหรับส่งต่อแพ็คเก็ตไปยังแอปพลิเคชั่นที่เขียนขึ้นมารองรับโดยเฉพาะ โดยจะต้องมี queue handler และแอปพลิเคชั่นเป็นส่วนประกอบที่จะทำงานร่วมกัน
- User-defined chain ผู้ใช้จะสามารถสร้างเชนใหม่ได้โดยต้องใช้ตัวอักษรตัวเล็กทั้งหมด สำหรับเชนที่สร้างขึ้นใหม่ เมื่อแพ็คเก็ตตรงกับกฎที่เป็น user-defined chain แพ็คเก็ตจะถูกนำไปตรวจสอบใหม่โดย user-defined chain นั้นๆและถ้าในเชนไม่มีการตัดสิน action ใดๆ ตัวแพ็คเก็ตก็สามารถย้อนกลับมายังกฎถัดไปในเชนเริ่มต้นได้

5.3.2 สนอร์ตอินไลน์ (Snort_inline)

สนอร์ตอินไลน์เป็นโปรแกรมไอดีเอสที่ทำงานในยูสเซอร์โหมด ซึ่งจะทำหน้าที่เป็นระบบตรวจจับผู้บุกรุก ในโครงงานนี้ได้นำสนอร์ตอินไลน์ มาใช้ในการจำแนกว่าผู้ใดเป็นผู้ใช้งานปรกติและผู้ใดเป็นผู้บุกรุกระบบ ซึ่งจะจัดจำแนกตามข้อมูลหรือกฎที่ผู้ดูแลระบบใช้ ซึ่งในที่นี้โปรแกรมจะทำการอ่านค่าชิ้นข้อมูลที่ส่งมาทางมอดูล ip_queue ของไอพีเทเบิลแล้วนำมาเปรียบเทียบกับกฎที่ได้ตั้งไว้ ว่าเข้าข่ายเป็นการโจมตีหรือไม่ และทำการแจ้งเตือนในกรณีที่การตรวจสอบมีผลออกมาว่าชิ้นข้อมูลที่ส่งมามีลักษณะของการโจมตีระบบ

สนอร์ตอินไลน์เป็นโปรแกรมที่พัฒนามาจากโปรแกรมสนอร์ต (Snort) ซึ่งโปรแกรมสนอร์ตอินไลน์จะรับแพ็คเก็ตจากโปรแกรมไอพีเทเบิล ผ่านทาง libipq แทนที่ของเดิมที่จะรับจาก libpcap และกฎที่ใช้ในสนอร์ตอินไลน์จะเป็นรูปแบบของกฎใหม่ โดยมีการทำงาน drop, sdrop, reject โดยสั่งงานไอพีเทเบิลได้ 3 คำสั่งดังกล่าว ทำให้สนอร์ตอินไลน์จึงเหมือนทำหน้าที่เป็น IPS (Intrution Prevention System)

การกระทำอันเนื่องจากกฎที่ใช้

Drop - The drop rule type

จะไปสั่งให้ไอพีเทเบิลทำการ drop แพ็คเก็ต และทำการเก็บล็อกไว้

Reject - The reject rule type

จะไปสั่งให้ไอพีเทเบิลทำการ drop แพ็คเก็ต และทำการเก็บล็อกไว้ พร้อมทั้งส่ง TCP reset หรือ UDP reset กลับไป แล้วแต่กรณี

Sdrop - The sdrop rule type

จะไปสั่งให้ไอพีเทเบิลทำการ drop แพ็คเก็ต โดยไม่ทำการบันทึกใดๆ

ลำดับในการพิจารณากฎของSnort inline

->activation->dynamic->drop->sdrop->reject->alert->pass->log

สามารถใช้ทางเลือก –o เพื่อเปลี่ยนลำดับเป็น

->activation->dynamic->pass->drop->sdrop->reject->alert->log

Stream4 option การทำงานใหม่ใน Snort inline

ซึ่งสามารถเปิดการใช้งานในทางเลือก ได้อีก 2 ความสามารถดังนี้

inline_state (no arguments)

เมื่อเปิดใช้งาน สนอร์ตอินไลน์จะทำการdrop แพ็คเก็ตที่ไม่เกี่ยวข้องหรือไม่สอดคล้องกับ TCP sessionที่มีอยู่ รวมถึงแพ็คเก็ตที่มี TCP initiator ผิดรูปแบบ

midstream_drop_alerts (no arguments)

โดยค่ากำหนดเริ่มต้นสำหรับสนอร์ตอินไลน์จะทำการdrop โดยไม่แสดงการเตือนใดๆ ถ้าหากเราต้องการให้มีการแจ้งเตือนจะต้องเปิดโหมดการทำงานอันนี้ด้วย

ความสามารถในการแก้ไขข้อมูลในแพ็คเก็ต

นั่นคือเราสามารถสั่งให้มีการตรวจสอบ ข้อมูลที่ผ่านเข้าออกที่ผ่านตัวสนอร์ตอินไลน์ว่ามีค่าที่ตรงกับที่เราตั้งไว้หรือไม่ และถ้ามีจะทำการเปลี่ยนให้เป็นไปตามรูปแบบที่เราต้องการ เช่น

```
alert tcp any any <> any 80 (msg: "tcp replace"; content:"GET"; replace:"BET";)
alert udp any any <> any 53 (msg: "udp replace"; content: "yahoo"; replace: "xxxxx";)
```

ดังตัวอย่าง สนอร์ตอินไลน์จะดูข้อมูลที่เป็นTCP ที่ใช้งาน port 80 โดยถ้าพบว่ามีส่วนที่ตรงกับคำว่า GET โปรแกรมจะให้เปลี่ยนเป็น BET และถ้าพบข้อมูลที่เป็น UDP ที่ใช้งาน port 53 ที่มีข้อมูลตรงกับคำว่า yahoo ก็ให้เปลี่ยนเป็น xxxxx

5.3.3 S2I (snort command to iptables)

เป็นโปรแกรมที่สร้างมาจากภาษาเพิร์ลที่พัฒนาขึ้น เพื่อให้สามารถแปลงการแจ้งเตือนจากโปรแกรมสนอร์ตอินไลน์ (Snort_inline) ไปเป็นคำสั่งของโปรแกรมไฟร์วอลไอพีเทเบิล (Iptables) เพื่อโปรแกรมไอพีเทเบิลจะได้จัดเส้นทางให้กับชิ้นข้อมูลที่มาจากผู้ใช้งานปรกติผ่านไปยังส่วนที่เป็นเซิร์ฟเวอร์จริง และในส่วนของผู้บุกรุกเองนั้นก็ให้ส่งไปยังเครื่องกับดักที่ได้เตรียมไว้ ซึ่งสาเหตุที่ต้องมีเอสทูไอนั้นเนื่องมาจากโปรแกรมไอดีเอสและโปรแกรมไอพีเทเบิลที่ใช้ในโครงงานนี้จะทำงานในโหมดที่ต่างกันจึงติดต่อกันเองโดยตรงไม่ได้ ดังนั้นจึงต้องมีโปรแกรมสื่อกลางช่วยในการติดต่อระหว่างกัน

นอกจากเป็นโปรแกรมในการสร้างเส้นทางใหม่ให้ผู้บุกรุกแล้ว ยังทำหน้าที่เป็นเซอร์วิสที่คอยตรวจสอบสภาวะของเครื่องกับดักในระบบไฟร์แทรป (FireTrap) และจะทำการหยุดการทำงานของเครื่องกับดักหากเครื่องกับดักมีความอ่อนแอเกินกว่าที่จะสามารถควบคุมพฤติกรรมของผู้บุกรุกได้ โดยในการทำงานโปรแกรมจะไปอ่านข้อกำหนดสำหรับเครื่องกับดักที่ผู้ดูแลระบบกำหนดไว้ในฐานข้อมูลแล้วทำการตรวจสอบเครื่องกับดักว่ามีสภาพตรงตามเงื่อนไขที่กำหนดหรือยัง เช่นมีการเปลี่ยนแปลง Password ในเครื่องไฟร์แทรปหรือไม่ หากเครื่องกับดักมีสภาพถึงตามที่กำหนดไว้ โปรแกรมจะทำการหยุดการทำงานของเครื่องกับดักในระบบไฟร์แทรปโดยอัตโนมัติ และทำการสั่งเครื่องกับดักตัวใหม่ขึ้นมาทำงาน เพื่อป้องกันไม่ให้ผู้บุกรุกใช้เครื่องกับดักเป็นฐานที่มั่นในการโจมตีระบบอื่น ๆ ซึ่งจะก่อให้เกิดความเสียหายต่อระบบจริงขององค์กรต่อไป

5.3.4 Oinkmaster

เป็นโปรแกรมที่ช่วยในการอัพเดต Signature ของโปรแกรมสนอร์ตอินไลน์โดยอัตโนมัติจากเว็บไซต์ต่างๆ ที่มี Signature ของสนอร์ตอินไลน์อยู่ ซึ่งผู้ดูแลระบบสามารถทำการกำหนดเว็บไซต์ที่ต้องการดาวน์โหลดได้แต่จะต้องแน่ใจว่าเว็บไซต์ต่าง ๆ นั้นน่าเชื่อถือ ซึ่งทางผู้พัฒนาได้ใช้โปรแกรมนี้เพื่อช่วยในการอัพเดตและแจกจ่ายกฎของสนอร์ตอินไลน์ไปยังส่วนต่างๆ ของระบบที่ต้องมีการจำแนกแพ็คเก็ตผู้บุกรุก

ในการใช้งาน Oinkmaster ทำได้โดยติดตั้งแพ็คเกจ Oinkmaster เพื่อช่วยในการดาวน์โหลดกฎจากเว็บไซต์ต่าง ๆ แบบอัตโนมัติ โดยขั้นตอนการติดตั้งแพ็คเกจ Oinkmaster สามารถทำได้ดังนี้

- แตกไฟล์แพ็คเกจ Oinkmaster ที่ดาวน์โหลดมา แล้วทำการย้ายไฟล์ oinkmaster.pl และ oinkmaster.conf ไปไว้ที่ไดเรกทอรี่ใดก็ได้

- แก้ไขไฟล์ oinkmaster.conf ที่บรรทัด 61 โดยแทนที่ <oinkcode> ด้วย key ที่ได้จากการลงทะเบียนทางเว็บไซต์ โดยแก้ไขไฟล์ดังนี้

url = http://www.snort.org/pub-bin/oinkmaster.cgi/<oinkcode>/snortrules-snapshot-2.3.tar.gz

เป็น

url = http://www.snort.org/pub-bin/oinkmaster.cgi/587cba039g9f03300f9e98b/snortrules-snapshot-2.3.tar.gz

หรือหากไม่มี key ที่ลงทะเบียนจากเว็บไซต์ www.snort.org ก็สามารถแก้ไขโดยเพิ่ม URL ที่เป็น URL สำหรับดาวน์โหลด Signature แบบ Community-rules ได้

- ทดสอบการดาวน์โหลดกฎได้โดยสั่ง /etc/oinkmaster.pl –o /ect/snort_inline/rules/

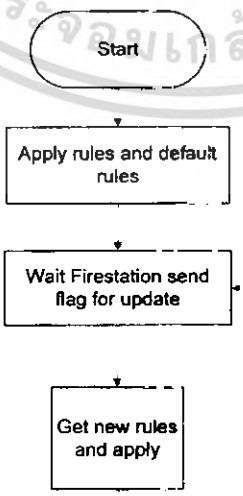
5.4 การทำงานของโปรแกรม

5.4.1 การทำงานร่วมกับไฟร์สเตชั่น

ไฟร์สกรีนมีระบบปฏิบัติการเป็นลินุกซ์ ซึ่งในที่นี้ใช้ดิสโทรเดเบียน (debian) โดยทำงานร่วมกับแอ็คทีฟไดเร็คทอรีของไมโครซอฟท์บนไฟร์สเตชั่นนั้นจะใช้แพ็กเกจของ OpenLDAP โดยตัวโปรแกรมไฟร์สกรีนจะต้องใช้ไลบรารีของ OpenLDAP ในการเขียนโปรแกรมทำงานร่วมกับแอ็คทีฟไดเร็คทอรี

5.4.1.1 การดึงกฎจากไฟร์สเตชั่น

สำหรับกฎของไฟร์วอลที่เก็บไว้ในแอ็คทีฟไดเร็คทอรีบนไฟร์สเตชั่นนั้นถูกเก็บไว้ใต้ dn คือ "cn=firescreen, cn=computers, dc=hephaestus, dc=com" ภายในแอตทริบิวต์ชื่อ firewallRule ดังนั้นจึงสามารถเข้าถึงได้โดยใช้ฟังก์ชั่น ldap_search_s() กำหนดขอบเขตของการค้นหาเป็น "cn=firescreen, cn=computers, dc=hephaestus, dc=com" จากนั้นจึงดึงค่าออกจากแอตทริบิวต์ firewallRule จนครบทุกค่าและเก็บค่าไว้รอใช้งานในอาเรย์



รูปที่ 5.2 การรับกฎจากไฟร์สเตชั่น

จากรูปแบบการทำงานของไฟร์สกรีนที่กล่าวถึงในหัวข้อก่อนหน้า เมื่อกฎมีการถูกแก้ไข ไฟร์สเตชั่นจะส่งสัญญาณมายังไฟร์สกรีนเพื่อติดต่อไปยังแอ็คทีฟไดเร็คทอรีเพื่อทำการดึงกฎมาจากแอตทริบิวต์ Firewallrule

5.4.1.2 การนำมาใช้งานกับไอพีเทเบิล

กฎที่ดึงมาจากทางไฟร์สเตชั่นนั้นจะอยู่ในรูปแบบดังต่อไปนี้

SRC:MASK:DST:MASK:PROTOCOL:SPORT:DPORT:ICMP-TYPE:ZONE:TARGET

SRC	ไอพีแอดเดรสต้นทาง
MASK	เน็ตมาสก์ของไอพีแอดเดรสต้นทาง
DST	ไอพีแอดเดรสปลายทาง
MASK	เน็ตมาสก์ของไอพีแอดเดรสปลายทาง
PROTOCOL	โพรโตคอล (TCP, UDP, ICMP หรือ any)
SPORT	พอร์ตต้นทาง (เมื่อระบุโพรโตคอลเป็น TCP หรือ UDP เท่านั้น)
DPORT	พอร์ตปลายทาง (เมื่อระบุโพรโตคอลเป็น TCP หรือ UDP เท่านั้น)
ICMP-TYPE	หมายเลข ICMP-TYPE (เมื่อระบุโพรโตคอลเป็น ICMP เท่านั้น)
ZONE	โซนที่กฎนี้จะมีผลในการใช้งาน
TARGET	ปลายทางของแพ็กเก็ตที่แมชท์กับกฎนี้

ตารางที่ 5.1 แสดงความหมายของแต่ละค่า

โดยก่อนที่จะสามารถนำมาใช้งานกับไฟร์สกรีนได้จะต้องแปลงกฎที่รับมาจากไฟร์สเตชั่นที่มีรูปแบบดังกล่าวให้อยู่ในรูปแบบของไอพีเทเบิลเสียก่อน โดยผ่านฟังก์ชั่นแปลง (parser) ซึ่งจะแยกแต่ละส่วนออกจากกันโดยมีเครื่องหมายโคลอน (:) เป็นตัวแบ่ง โดยนำเอาแต่ละส่วนมาต่อเข้าด้วยกันตามรูปแบบคำสั่งของไอพีเทเบิล ส่วนกฎจะถูกเพิ่มเข้าเชน (CHAIN) ใดของไอพีเทเบิลจะพิจารณาจาก ZONE และ TARGET ที่ระบุเป็น DROP ทั้งหมดจะถูกระบุ TARGET ในคำสั่งไอพีเทเบิลเป็น -j LOG-CHAIN ซึ่งจะส่งแพ็คเก็ตที่ตรงกับกฎเข้าไปยังเชนชื่อ LOG-CHAIN เพื่อที่หากแพ็คเก็ตที่ถูกละทิ้ง (DROP) จะมีการเก็บข้อมูลลงล็อกไฟล์

```
Chain LOG-CHAIN (4 references)
num  target     prot opt source               destination
1    LOG        all  --  anywhere             anywhere            limit: avg 1/sec
 burst 5 LOG level info prefix `FIRESCREEN '
2    DROP       all  --  anywhere             anywhere
```

รูปที่ 5.3 กฎในเชน LOG-CHAIN

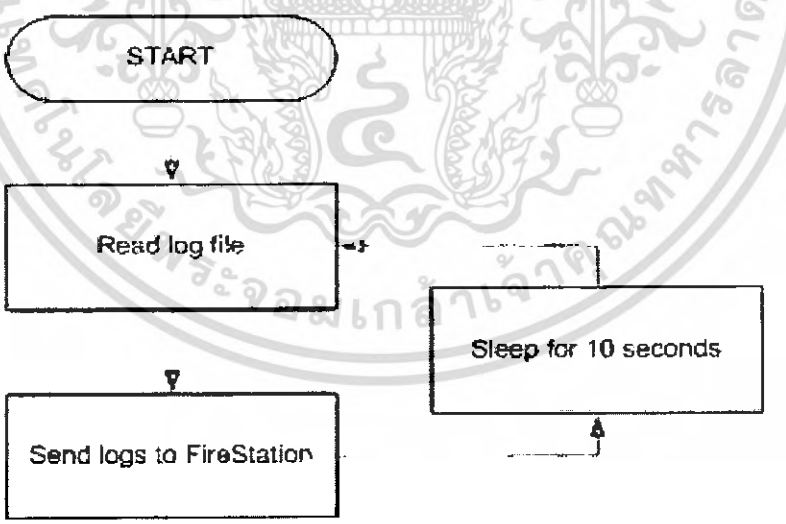
กฎแรกจะทำการเก็บข้อมูลทุกแพ็คเก็ตที่ผ่านเข้ามาในเชนนี้ลงล็อกไฟล์ในอัตรา 1 แพ็คเก็ตต่อวินาที (limit: avg 1/sec burst 5) โดยทุกๆค่าที่ถูกเก็บลงล็อกไฟล์จะมีคำว่า "FIRESCREEN " (prefix 'FIRESCREEN') ขึ้นต้น เพื่อประโยชน์ในการคัดแยกล็อกในภายหลัง เนื่องจากกระบวนการเก็บเหตุการณ์ลงล็อกไฟล์นั้นถูกจัดการโดย syslogd ซึ่งไม่ได้ถูกจัดการโดยไอพีเทเบิลเองทำให้ในล็อกไฟล์จะเก็บเหตุการณ์ที่มีระดับของการล็อกเหมือนกันเอาไว้ โดยในเชน LOG-CHAIN นี้ระบุระดับของการล็อกเป็น info (LOG level info)

กฎที่สองจะทำการละทิ้งหมดทุกๆแพ็คเก็ต เนื่องด้วยแพ็คเก็ตที่ถูกส่งต่อมายังเชนนี้เพื่อเก็บข้อมูลลงล็อกไฟล์ ซึ่งคือแพ็คเก็ตที่ถูกกำหนด TARGET จากไฟร์สเตชั่นไว้เป็น DROP นั่นเอง

สุดท้ายเมื่อได้กฎจากไฟร์สเตชั่นอยู่ในรูปแบบคำสั่งของไอพีเทเบิลทั้งหมดแล้วก็จะทำการเขียนทั้งหมดลงบนไฟล์เชลล์สคริปต์ เพื่อทำการรันเพื่อใช้งานตามคำสั่งไอพีเทเบิลดังกล่าว รวมทั้งเก็บไฟล์นี้ไว้ใช้ครั้งต่อไปหากโปรแกรมถูกปิดหรือเครื่องไฟร์สกรีนเริ่มทำงานใหม่เพื่อที่จะสามารถคงกฎเดิมไว้ได้

5.4.1.3 ส่วนที่ทำหน้าที่ส่งล็อกไปยังไฟร์สเตชั่น

โปรแกรมในส่วนนี้จะทำการอ่านล็อกไฟล์จากเครื่องเกตเวย์และส่งไปเก็บยังแอ็คทีฟไดเร็คทอรีบนไฟร์สเตชั่น สามารถแสดงลักษณะการทำงานเป็นแผนภาพดังนี้



รูปที่ 5.4 แสดงการส่งล็อกของไฟร์สกรีน

เมื่อเริ่มการทำงานขึ้นมาโปรแกรมจะคอยตรวจสอบล็อกไฟล์เป็นระยะ และทำการส่งล็อกที่เกี่ยวข้องในส่วนของไอพีเทเบิลกลับไปยังไฟร์สเตชั่น

5.4.2 การทำงานร่วมกับไฟร์แทรป

หน้าที่ในส่วนของการทำงานร่วมกับไฟร์แทรปนั้น ไฟร์สกรีนจะรับแพ็คเก็ตที่ไม่ตรงตามกฎใด ๆ ที่ได้รับจากไฟร์สเตชั่น แล้วนำมาจำแนกว่ามีพฤติกรรมเข้าข่ายเป็นผู้บุกรุกหรือไม่ โดยอาศัยระบบตรวจจับผู้บุกรุก ที่ชื่อสนอร์ตอินไลน์ (Snort_inline) ซึ่งเป็นเครื่องมือประเภท IDS ที่ทำงานอยู่ในระบบไฟร์สกรีนเพื่อให้มีความสามารถดังที่ได้กล่าวมา ในการทำงานของไฟร์สกรีนนั้นจะทำงานในรูปแบบของ Bridge Mode ทำให้ผู้ใช้งานที่ต้องการใช้งานเครื่องเซิร์ฟเวอร์จริง ไม่ทราบว่ามีการส่งข้อมูลผ่านตัวไฟร์สกรีน

และเมื่อระบบตรวจจับผู้บุกรุกในไฟร์สกรีนทำการตรวจจับแพ็คเก็ตที่เข้าข่ายเป็นผู้บุกรุกได้แล้ว โปรแกรม S2I ที่อยู่ในไฟร์สกรีนจะทำการเปลี่ยนเส้นทางให้บุกรุกเข้าไปใช้งานเครื่องกับดักในระบบไฟร์แทรปแทน ซึ่งเป็นเครื่องจำลองที่เตรียมไว้และมีสภาพแวดล้อมและค่าต่างๆ เหมือนเครื่องให้บริการจริง พร้อมทั้งส่งข้อมูลเกี่ยวกับการบุกรุกที่ตรวจจับได้กลับไปยังเครื่องแม่ข่ายฐานข้อมูลกลางของระบบ (LogServer) นอกจากนี้โปรแกรม S2I ยังทำหน้าที่ในการตรวจสอบความอ่อนแอของเครื่องกับดักในระบบไฟร์แทรป เช่น เครื่องกับดักถูกผู้บุกรุกทำการเปลี่ยนพาสเวิร์ด หรือเพิ่มผู้ใช้ใหม่หรือไม่ ซึ่งถ้าเครื่องกับดักถูกบุกรุกด้วยวิธีการดังกล่าว โปรแกรม S2I จะทำการหยุดการทำงานของเครื่องกับดักตัวนั้นพร้อมทั้งสั่งให้เครื่องกับดักตัวใหม่ขึ้นมาทำงานแทน

ในส่วนการทำงานของไฟร์สกรีนที่เป็นส่วนจำแนกผู้บุกรุกและทำงานร่วมกับระบบไฟร์แทรปตามที่ได้กล่าวมานี้ จะมีโปรแกรมควบคุมการทำงานจากศูนย์กลาง คือ โปรแกรม Tartarus Management ซึ่งเป็นโปรแกรมที่ควบคุมการทำงานของสนอร์ตอินไลน์ , ควบคุมการทำงานของเครื่องกับดักในไฟร์แทรป รวมถึงการอัพเดต , สร้าง และเลือกใช้กฎต่างๆ ของสนอร์ตอินไลน์โดย ในการควบคุมระบบผ่านโปรแกรม Tartarus Management นี้ผู้ดูแลระบบจะกระทำผ่านช่องทางที่ปลอดภัยโดยใช้ IPsec (IPsecurity) ซึ่งจะมีการพิสูจน์ตัวตน (Authentication) และการเข้ารหัสข้อมูล (Encryption) ทำให้สามารถจัดการระบบผ่านเครือข่ายได้อย่างปลอดภัยมากยิ่งขึ้น

5.4.3 หลักการจำแนกว่าเป็นผู้บุกรุก

ในการจำแนกผู้บุกรุกนั้นจะพิจารณาว่าผู้ใดเป็นผู้บุกรุกหรือเข้าข่ายว่าเป็นผู้บุกรุกโดยใช้กฎของโปรแกรมตรวจจับผู้บุกรุก (Snort_inline) ซึ่งเป็นเครื่องมือประเภท IDS (Intrusion Detection System) โดยหากว่าการเชื่อมต่อใดมีลักษณะตรงตามกฎของโปรแกรมตรวจจับผู้บุกรุกจะถือว่าเจ้าของการเชื่อมต่อนั้นเข้าข่ายเป็นผู้บุกรุกจะต้องทำการเปลี่ยนทิศทางการเชื่อมต่อดังกล่าวไปยังเครื่องกับดักเพื่อดำเนินการบันทึกพฤติกรรมของผู้บุกรุกนั้นและเพื่อให้การจำแนกผู้บุกรุกของโปรแกรมตรวจจับผู้บุกรุกเป็นไปตามความต้องการของผู้พัฒนามากขึ้น ทางผู้พัฒนาจึงได้ทำการแก้ไขกฎการจำแนกบางส่วน เช่น

Mysql cazz exploit

กฎของการตรวจสอบการบุกรุกของเครื่องมือที่ใช้เข้าโจมตีโปรแกรมฐานข้อมูล ที่ชื่อว่า มายเอสคิวแอล (mysql) โดยชื่อโปรแกรมที่เรียกว่า cazz เพื่อเข้าโจมตีแบบยึดครองระบบ เพื่อตั้งให้กฎเองมีความยืดหยุ่นที่ว่าหากมีการเชื่อมต่อเข้ามายังช่องทางหมายเลข 3306 ลักษณะการขอการสถาปนา ก็ให้ถือว่าเป็นการโจมตีจากผู้บุกรุก

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306

(msg:"MYSQL root login attempt";

flow:to_server,established;)

5.4.4 หลักการจำแนกว่าเป็นผู้ใช้ปรกติ

ในการจำแนกผู้ใช้ปรกตินั้นจะพิจารณาโดยใช้กฎของโปรแกรมตรวจจับผู้บุกรุกเช่นเดียวกับการจำแนกผู้บุกรุก คือการเชื่อมต่อใดที่ไม่ตรงกับกฎให้ถือว่าเข้าของการเชื่อมต่อนั้นๆ เป็นผู้ใช้ปรกติสามารถให้ผ่านไปใช้บริการของเครื่องแม่ข่ายจริงได้ตามปรกติ และกรณีที่ไม่ต้องการให้มีการตรวจสอบเครื่องบางเครื่องที่เชื่อมต่อเข้ามา สามารถทำได้โดยการกำหนดกฎที่ ไฟร์วอลให้การเชื่อมต่อนั้นสามารถผ่านเข้าไปได้โดยไม่ต้องมีการตรวจสอบเนื่องจากว่าผู้ที่ควบคุมการเชื่อมต่อจริงแล้วคือไฟร์วอล ในการตรวจสอบของโปรแกรมตรวจจับผู้บุกรุกอาจมีความผิดพลาดได้จึงจำเป็นต้องตั้งกฎของระบบตรวจจับผู้บุกรุกให้มีความรัดกุมและบ่งบอกชัดเจนว่าผู้ใดเป็นผู้บุกรุก เพื่อให้กระบวนการตรวจสอบนั้นไม่เกิดผลการเข้าใจผิดว่า ผู้ใช้งานปรกตินั้นเป็นผู้บุกรุก

5.4.5 หลักการจำแนกว่าเป็นโค้ดหนอน

ในการระบุได้ว่าเป็นโค้ดหนอน จะใช้การพิจารณาแบบ Rules-base เช่นเดียวกับการตรวจจับผู้บุกรุก โดยจะใช้ฐานข้อมูลที่เก็บ Signature ของโค้ดหนอนชนิดต่างๆ เอาไว้ มาเปรียบเทียบรูปแบบที่พบ ซึ่งจำเป็นจะต้องมีการปรับปรุงฐานข้อมูล Signature นี้ให้ทันสมัยอยู่ตลอดเวลา ซึ่งทางผู้พัฒนาได้เพิ่มเข้าไปในส่วนของสนอร์ตอินไลน์เพื่อให้สามารถตรวจสอบโค้ดหนอนไปพร้อมกับตรวจจับผู้บุกรุกเลย

บทที่ 6
การพัฒนาระบบไฟร์อลาร์ม

6.1 แนวคิด

ระบบไฟร์อลาร์มเป็นระบบสำหรับเป็นไฟร์วอลส่วนบุคคล ที่มีระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ที่พัฒนาขึ้นเองโดยภาษา C เพื่อใช้งานร่วมกับไฟร์สเตชั่นในการรับกฎของเพอร์ซันนอลไฟร์วอลสำหรับแต่ละเครื่องลูกข่าย โดยเป็นการใช้งานไฟร์วอลแบบอิงผู้ใช้คือเมื่อผู้ใช้มีการล็อกอินใช้งานเครื่องที่อยู่ในระบบเครือข่ายภายในเครื่องใดก็ตาม ก็จะมีกฎสำหรับผู้ใช้คนนั้นในการใช้งานเพอร์ซันนอลไฟร์วอลเสมอ

6.2 ขอบเขตและความสามารถ

ในระบบจะมีเครื่องลูกข่ายหลายเครื่องโดยทุกเครื่องจะมีการติดตั้งไฟร์วอลส่วนบุคคล ซึ่งเป็นเอเจนต์ที่ฝังตัวทำงานอยู่แบบอัตโนมัติ โดยจะมีการควบคุมดูแลจากส่วนกลาง ทำให้ผู้ใช้ไม่รับรู้ถึงการทำงานของเอเจนต์ รวมไปถึงผู้ดูแลระบบสามารถกำหนดกฎการป้องการบุกรุกให้ผู้ใช้ในแต่ละกลุ่มตามความเหมาะสม

อีกทั้งยังมีส่วนของการทำงานที่เป็นระบบตรวจจับผู้บุกรุกที่คอยทำหน้าที่ตรวจสอบว่ามีการบุกรุกเข้ามาที่เครื่องลูกข่ายนั้นๆหรือไม่ ถ้ามีจะมีการแจ้งเตือนไปยังเครื่องเซิร์ฟเวอร์กลางเพื่อนำผลที่ได้ไปวิเคราะห์หาแนวทางป้องกันต่อไป และมีส่วนตรวจการเชื่อมต่อเครือข่ายของโพรเซสต่างๆ เพื่อให้ผู้ใช้งานได้รับรู้ว่ามีโพรเซสอะไรที่เชื่อมต่อกับเครือข่ายอยู่บ้าง และสามารถกำจัดโพรเซสที่น่าสงสัยได้ด้วยตนเอง ทำให้ผู้ใช้สามารถป้องกันตนเองได้ในระดับหนึ่ง

6.3 เทคโนโลยีที่เกี่ยวข้องและการพัฒนาโปรแกรม

การพัฒนาโปรแกรมไฟร์อลาร์มจะใช้ Firewall-Hook Driver ซึ่งเป็น DDK (Driver Development Kit) ของทางบริษัทไมโครซอฟต์ ซึ่งไดร์เวอร์ตัวนี้ใช้ในการฟิลเตอร์แพ็คเก็ต การเขียนโปรแกรมนั้นจะใช้เฮดเดอร์ (Header) ที่ชื่อว่า IpDrvFw.h โดยภายในเฮดเดอร์ (Header) นี้จะมีโครงสร้างต่างๆที่ใช้ในการฟิลเตอร์ และใช้ WinPCap ช่วยในการดักจับแพ็คเก็ตเพื่อมาวิเคราะห์หาว่ามีการบุกรุกระบบเกิดขึ้นหรือไม่ โดยกฎที่ใช้ในการฟิลเตอร์นั้นจะรับมาจากฐานข้อมูลของแอ็คทีฟไดเร็คทอรี (Active Directory Database)

เทคโนโลยีอื่นๆ ที่ใช้ในการพัฒนาโปรแกรมมีดังนี้

6.3.1 Windows Service

ไมโครซอฟต์วินโดวส์เซอร์วิส หรือก่อนหน้านี้ถูกเรียกว่า NT service ซึ่งวินโดวส์เซอร์วิสนี้จะช่วยในการสร้างโปรแกรมที่รันในส่วนของวินโดวส์เซ็สชั่น (Windows session) ซึ่งโปรแกรมที่เป็นเซอร์วิสจะมีการทำงานตลอดตั้งแต่วินโดวส์เริ่มบูตและทำงานเป็นแบ็คกราวไปเรื่อยๆ จนกว่าจะทำการปิดวินโดวส์ ซึ่งเซอร์วิสนี้จะสามารถควบคุมได้เช่น หยุดการทำงานชั่วคราว (pause) หรือสั่งให้เริ่มการทำงานใหม่ (restart) โดยการควบคุมจะควบคุมผ่าน Services Control Manager ซึ่งเป็นตัวควบคุมเซอร์วิสทั้งหมด โดยทั่วไปโปรแกรมที่เป็นเซอร์วิสจะไม่มีส่วนติดต่อกับผู้ใช้ (User Interface)

6.3.1.1 การสร้างโปรแกรมเซอร์วิส (Service program)

ในการสร้างในที่นี้จะใช้ภาษา C++ ซึ่งจะต้องลง platform sdk ก่อน และทำการ #include <windows.h> ด้วย โดยมีขั้นตอนหลักๆดังนี้

ติดต่อกับ SCM

ในการทำงานที่เกี่ยวกับเซอร์วิสทุกๆอย่าง จะต้องทำการติดต่อกับ SCM (Service Control Manager) เพื่อทำการเปิด database ของ SCM ทุกครั้ง และนำ handle ที่ได้ไปใช้ในการสร้าง, ควบคุมเซอร์วิส โดยในการการติดต่อกับ SCM ทำได้ดังนี้

```
SC_HANDLE schSCManager;     // Open a handle to the SC Manager database.
SchSCManager = OpenSCManager (
  NULL,                     // local machine
  NULL,                     // ServicesActive database
  SC_MANAGER_ALL_ACCESS);  // full access rights
```

สร้างเซอร์วิส

ทำการสร้างเซอร์วิส โดยนำ handle ของ SCM database มาใช้กับฟังก์ชัน CreateService และทำการเซ็ตค่าพารามิเตอร์ต่างๆ ดังนี้

```
SC_HANDLE schService = CreateService (
    schSCManager,                   // SCManager database
    TEXT("Sample_Srv"),             // name of service
    lpszDisplayName,                // service name to display
    SERVICE_ALL_ACCESS,             // desired access
    SERVICE_WIN32_OWN_PROCESS,  // service type
```

```
SERVICE_DEMAND_START,        // start type
SERVICE_ERROR_NORMAL,        // error control type
szPath,                      // path to service's binary
NULL,                        // no load ordering group
NULL,                        // no tag identifier
NULL,                        // no dependencies
NULL,                        // LocalSystem account
NULL);                       // no password
```

จากตัวอย่างมีพารามิเตอร์ที่สำคัญอยู่ดังนี้

- พารามิเตอร์ตัวแรกที่ชื่อ schSCManager พารามิเตอร์ตัวนี้ได้มาจากขั้นตอนแรก
- พารามิเตอร์ตัวที่สองที่เป็นฟังก์ชัน TEXT("Sample_Srv") พารามิเตอร์นี้จะนำมาใช้เป็นชื่อของเซอร์วิส
- พารามิเตอร์ตัวที่สามที่ชื่อ lpszDisplayName พารามิเตอร์นี้จะนำมาใช้แสดงเป็นชื่อเซอร์วิสในโปรแกรม Control panel > Administrative tools > Services
- พารามิเตอร์ตัวที่แปด ที่ชื่อ szPath พารามิเตอร์ตัวนี้เป็นที่อยู่ของโปรแกรมที่ต้องการทำให้มีการทำงานเป็นเซอร์วิส

เมื่อเสร็จสิ้นขั้นตอนนี้เราก็จะได้เซอร์วิสโปรแกรมแล้วแต่ยังไม่ครบถ้วน นั่นคือต้องมีส่วนที่ควบคุมการทำงานของเซอร์วิสและส่วนการนำเซอร์วิสออกอีก

ควบคุมการทำงานของเซอร์วิส (Control Service)

การเริ่มเซอร์วิส (Start Service) ก่อนที่จะสั่งให้เซอร์วิสทำการเริ่มต้นการทำงานนั้นจะต้องทำการเปิดเซอร์วิสที่ต้องการจะสั่งก่อน โดยใช้ฟังก์ชัน OpenService ดังนี้

```
SC_HANDLE schService;
    schService = OpenService(
    schSCManager,            // SCM database
    "Sample_Srv",            // service name
    SERVICE_ALL_ACCESS);
```

เมื่อทำการเปิดเซอร์วิสที่ต้องการแล้วเราจะได้ handle ของเซอร์วิสนั้น จากนั้นเราก็นำ handle ที่ได้มาทำการเริ่มการทำงานโดยเรียกฟังก์ชัน StartService ดังนี้

```
StartService(
    schService,      // handle to service
    0,               // number of arguments
    NULL)
```

การหยุดเซอร์วิส (Stop Service) จะต้องทำการเปิดเซอร์วิสก่อนเหมือนดังที่ได้กล่าวมาแล้ว จากนั้นทำการหยุดเซอร์วิสโดยเรียกฟังก์ชัน ControlService และทำการส่ง พารามิเตอร์ SERVICE_CONTROL_STOP กับ service handle ไป ดังนี้

```
ControlService(schService,SERVICE_CONTROL_STOP,&status)
```

หลังจากทำงานเสร็จให้ปิด handle ของเซอร์วิสนั้นดังนี้

```
CloseServieeHandle(sehService);
```

การนำเซอร์วิสออก (Uninstall Service)

ทำการเปิดเซอร์วิสที่ต้องการเอาออกโดยใช้ OpenService ดังที่ได้กล่าวไปแล้ว หลังจากได้ handle ของเซอร์วิสแล้ว เราจะทำการลบเซอร์วิสได้โดยการเรียกฟังก์ชัน DeleteService ดังนี้

```
DeleteService(schService);
```

6.3.2 IPC (Inter Process Communication)

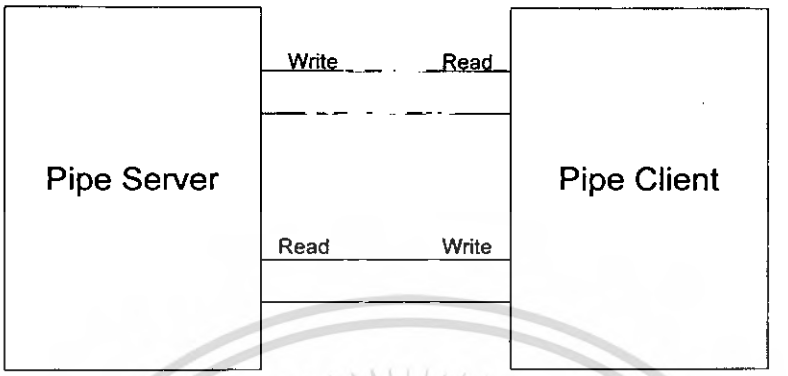
IPC คือกลไกในการติดต่อสื่อสารกันระหว่างโพรเซสหรือระหว่างโปรแกรม โดยปรกติโปรแกรมที่ใช้กลไก IPC จะใช้ในลักษณะการทำงานของไคลเอนต์เซิร์ฟเวอร์ โดยโปรแกรมที่เป็นไคลเอนต์จะทำการร้องขอไปยังโปรแกรมหรือโพรเซสอื่นๆ และในส่วนโปรแกรมที่เป็นเซิร์ฟเวอร์จะทำการตอบสนองต่อการร้องขอของโปรแกรมไคลเอนต์ และในหลาย ๆ โปรแกรมอาจจะเป็นทั้งไคลเอนต์และเซิร์ฟเวอร์ในตัวเลยก็ได้ ในการทำ IPC จะมีการทำอยู่หลายวิธีขึ้นอยู่กับลักษณะโปรแกรมที่ใช้ เช่น Clipboard, COM, Data Copy, DDE, File Mapping, Pipes, Windows Socket ... ในที่นี้เราทำ IPC โดยวิธีการทำ Pipes

6.3.2.1 การทำ IPC โดยใช้เทคนิค Pipes

Pipes สำหรับการทำการติดต่อสื่อสารมี 2 ชนิด คือ anonymous pipe และ named pipe ในที่นี้ใช้ named pipe โดยข้อแตกต่างระหว่าง pipes ทั้งสองนี้คือ

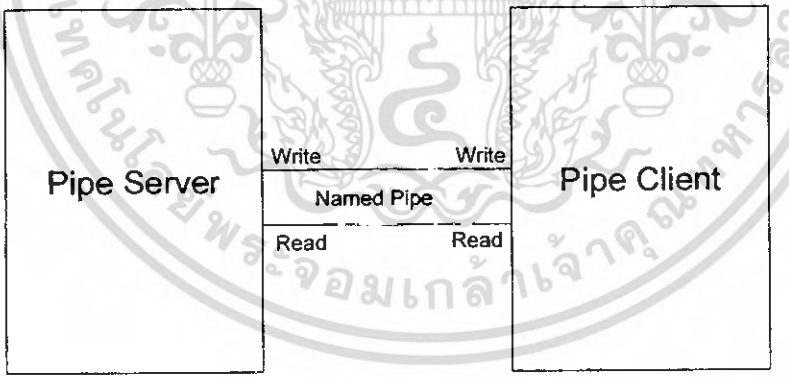
- Anonymous pipe - จะใช้งานในลักษณะที่สองโปรแกรมหรือโพรเซสที่จะทำการติดต่อสื่อสารกันต้องเกี่ยวเนื่องกันเช่นจะต้องเป็น parent process กับ child process โดยการติดต่อทาง pipe server จะทำการสร้าง pipe สำหรับการอ่านและการเขียนขึ้น และทาง

ฝั่งไคลเอ็นต์จะทำการติดต่อกับเซิร์ฟเวอร์ทางช่องทางนี้โดยไคลเอนต์จะทำการเขียนที่ช่องทางที่ฝั่งเซิร์ฟเวอร์อ่าน และอ่านทางช่องทางที่เซิร์ฟเวอร์เขียน ดังนี้



รูปที่ 6.1 รูปแสดงการทำงานแบบ Anonymous pipe

- Named pipe - จะใช้งานในลักษณะที่สองโปรแกรมหรือโพรเซสไม่จำเป็นต้องเกี่ยวข้องกันโดยการทำงานของ Named pipes จะเริ่มโดย named-pipe server โพรเซสทำการสร้าง named pipe ซึ่งเป็นชื่อของ pipe ที่จะทำการติดต่อกับไคลเอ็นต์ ซึ่ง named-pipe client จะรู้ชื่อของ pipe ที่ named pipe server สร้างขึ้น แล้วทำการเปิด pipe เพื่อทำการเชื่อมต่อและหลังจากเชื่อมต่อกันแล้ว named pipe client/server จะสามารถส่งข้อมูลหากันได้ ดังรูป



รูปที่ 6.2 รูปแสดงการทำงานแบบ Named pipe

การสร้าง Named pipes ที่ Named pipe server มีขั้นตอนดังนี้

- ที่ฝั่ง server ทำการสร้าง pipe โดยใช้ฟังก์ชัน CreateNamedPipe ดังนี้

```
#define g_szPipeName "\\\\.\\Pipe\\MyNamedPipe"
HANDLE hPipe;
hPipe = CreateNamedPipe(
 g_szPipeName,                    // pipe name
```

```
PIPE_ACCESS_DUPLEX,              // read/write access
PIPE_TYPE_MESSAGE |              // message type pipe
PIPE_READMODE_MESSAGE |          // message-read mode
PIPE_WAIT,                       // blocking mode
PIPE_UNLIMITED_INSTANCES,        // max. instances
BUFFER_SIZE,                     // output buffer size
BUFFER_SIZE,                     // input buffer size
NMPWAIT_USE_DEFAULT_WAIT,        // client time-out
NULL);                           // default security attribute
```

พารามิเตอร์ตัวแรกคือชื่อ named pipe ที่ทางฝั่งเซิร์ฟเวอร์สร้างขึ้นมาจะมีรูปแบบดังนี้ \\.\pipe\pipename ในที่นี้ named pipe มีชื่อเป็น "\\\\.\\Pipe\\MyNamedPipe" ซึ่งในส่วนของ pipename อาจจะเป็นชื่อเครื่องก็ได้ในกรณีที่ต้องการติดต่อออกสู่เน็ตเวิร์ก และเมื่อทำการสร้าง named pipe ขึ้นมาเราจะได้ handle ของ pipe มาใช้ในการติดต่อที่นี้ คือ hPipe

- รอการร้องขอจาก Client โดยใช้ฟังก์ชัน ConnectNamedPipe ดังนี้

```
//Wait for the client to connect
BOOL bClientConnected = ConnectNamedPipe(hPipe, NULL);
```

hPipe ที่เป็นพารามิเตอร์คือ handle ของ named pipe ที่ได้จากขั้นตอนแรก

- เมื่อเกิดการร้องขอจากไคลเอ็นต์ ทางฝั่งเซิร์ฟเวอร์ต้องทำการอ่านข้อความที่ไคลเอ็นต์ส่งมาใน pipe โดยใช้ฟังก์ชัน ReadFile ดังนี้

```
//Read client message
BOOL bResult = ReadFile(
hPipe,                  // handle to pipe
szBuffer,               // buffer to receive data
sizeof(szBuffer),       // size of buffer
&cbBytes,               // number of bytes read
NULL);                  // not overlapped I/O
```

โดยค่าที่อ่านได้จาก pipe จะอยู่ในตัวแปร szBuffer

- เซิร์ฟเวอร์สามารถตอบกลับทางฝั่งไคลเอ็นต์ได้โดยการเขียน Pipe ดังนี้

```
//Reply to client
bResult = WriteFile(
hPipe,                  // handle to pipe
szBuffer,               // buffer to write from
strlen(szBuffer)+1,     // number of bytes to write
&cbBytes,               // number of bytes written
NULL);                  // not overlapped I/O
```

- เมื่อเสร็จสิ้นการส่งข้อมูลให้ทำการปิด pipe ที่ได้ใช้งานไป ดังนี้

```
CloseHandle(hPipe);
```

<u>การติดต่อ Named pipes ทางฝั่ง Named pipe client มีขั้นตอนดังนี้</u>

ทำการติดต่อกับ server pipe โดยใช้ฟังก์ชัน CreateFile โดยใส่ชื่อ pipe ที่ทางฝั่งเซิร์ฟเวอร์สร้างขึ้นมาเข้าไปที่พารามิเตอร์ตัวแรกในที่นี้คือ g_szPipeName หลังจากนั้นจะได้ handle มาใช้งาน

```
HANDLE hPipe;
hPipe = CreateFile(
g_szPipeName,           // pipe name
GENERIC_READ |          // read and write access
GENERIC_WRITE,
0,                      // no sharing
NULL,                   // default security attributes
OPEN_EXISTING,          // opens existing pipe
0,                      // default attributes
NULL);                  // no template file
```

พารามิเตอร์ตัวที่สอง คือการกำหนดสิทธิในการเขียนหรืออ่าน pipe ในโครงงานนี้ส่วนของไฟร์อลาร์มเอเจ็นต์จะต้องกำหนดให้มีสิทธิให้อ่านได้เพียงอย่างเดียว เพราะว่าโปรแกรมไฟร์อลาร์มเอเจ็นต์จะต้องไม่สามารถที่จะเขียนทับ pipe ที่สร้างมาจากวินโดวส์เซอร์วิสได้

- ในส่วนของการอ่านหรือเขียนลง pipe จะทำเหมือนกับฝั่งเซิร์ฟเวอร์
- ปิด handle pipe ทุกตัวเหมือนกับทางฝั่งเซิร์ฟเวอร์

6.4 การทำงานของโปรแกรม

โปรแกรมไฟร์อลาร์มมีลักษณะการทำงานเป็นแบบวินโดวส์เซอร์วิสซึ่งการทำงานลักษณะนี้ระบบจะเป็นตัวรันโปรแกรม ซึ่งจะมีข้อดีดังนี้

- ทำให้ผู้ใช้ไม่จำเป็นต้องมีสิทธิเป็นผู้ดูแลระบบในการเข้าใช้งาน ดังนั้นผู้ใช้จึงไม่สามารถทำการปิดโปรแกรมได้ นอกจากผู้ดูแลระบบเท่านั้น
- ทำให้สามารถป้องกันเครื่องคอมพิวเตอร์ได้แม้ผู้ใช้ยังไม่ได้ทำการล็อกอิน

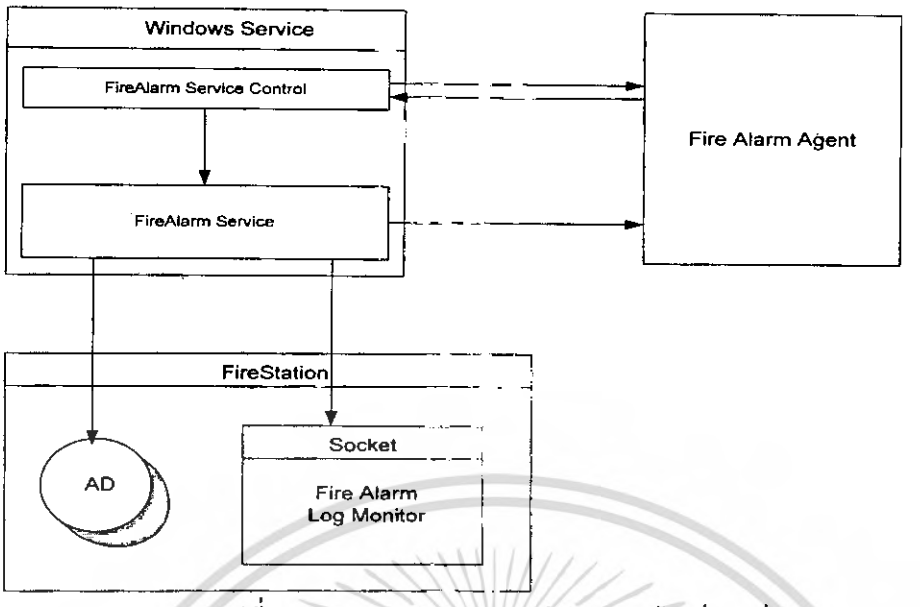
โปรแกรมไฟร์อลาร์มมีการทำงานหลักๆ แบ่งออกเป็น 3 ส่วนด้วยกันได้แก่

- ไฟร์อลาร์มเซอร์วิส (Firealarm Service) ทำหน้าที่ในการดึงกฎมาจาก AD แล้วทำการฟิลเตอร์กฎนั้นๆ นอกจากนี้ยังมีส่วนตรวจจับผู้บุกรุกแล้วทำการส่งรายงานไปยังไฟร์สเตชัน

- ไฟร์อลาร์มเซอร์วิสคอนโทล (Firealarm Service Control) ทำหน้าที่ในการควบคุมการทำงานของไฟร์อลาร์มเซอร์วิส เช่น ขั้นตอนของการล็อกอินผู้ใช้

- ไฟร์อลาร์มเอเจนต์โปรแกรม (Firealarm Agent Program) ทำหน้าที่แสดงกฎของผู้ใช้และแสดงรายงานเมื่อมีผู้บุกรุกเข้ามา และมีระบบตรวจการใช้งานเครือข่ายของโพรเซส

โดยการทำงานของโปรแกรมนี้จะทำงานตั้งแต่เริ่มระบบปฏิบัติการวินโดวส์ โดยเมื่อผู้ใช้ทำการล็อกอินเข้าใช้ระบบ โปรแกรมไฟร์อลาร์มเซอร์วิสคอนโทล จะทำการตรวจสอบเหตุการณ์การล็อกอินของผู้ใช้ แล้วทำการสั่งให้โปรแกรมไฟร์อลาร์มเซอร์วิสทำการเริ่มการทำงานใหม่เพื่อทำการดึงกฎที่ถูกต้องของผู้ใช้ เมื่อโปรแกรมไฟร์อลาร์มเซอร์วิสพร้อมที่จะใช้งานแล้ว โปรแกรมไฟร์อลาร์มเซอร์วิสคอนโทลจะสั่งให้โปรแกรมไฟร์อลาร์มเอเจนต์ทำงาน โดยโปรแกรมไฟร์อลาร์มเอเจนต์ทำหน้าที่ในการติดต่อกับไฟร์อลาร์มเซอร์วิส เพื่อดึงข้อมูลของผู้ใช้มาแสดง เมื่อเกิดเหตุการณ์มีผู้บุกรุกเข้ามา โปรแกรมไฟร์อลาร์มเซอร์วิสจะทำหน้าที่ตรวจสอบและทำการรายงานการบุกรุกผ่านทาง Socket ไปยังเครื่องไฟร์สเตชันโดยมีโปรแกรมไฟร์อลาร์มล็อกมอนิเตอร์เป็นตัวรับรายงานและเก็บลงฐานข้อมูลต่อไป



รูปที่ 6.3 รูปแสดงการทำงานโปรแกรมไฟร์อลาร์ม

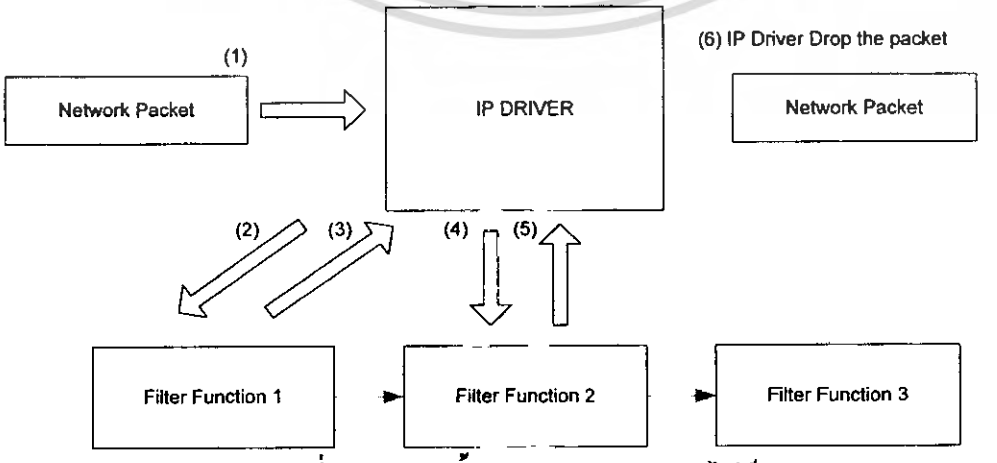
6.4.1 ไฟร์อลาร์มเซอร์วิส (Firealarm Service)

โปรแกรมไฟร์อลาร์มเซอร์วิส มีการทำงานหลักๆ อยู่ 2 ส่วนคือ

- ทำหน้าที่เป็นไฟร์วอลส่วนบุคคล
- ทำหน้าที่เป็นระบบตรวจจับผู้บุกรุก

6.4.1.1 <u>แพ็คเก็ตฟิลเตอริงไฟร์วอลบนวินโดวส์ XP</u>

เพอร์ซันนอลไฟร์วอลที่พัฒนามีลักษณะการทำงานเป็นแบบแพ็คเก็ตฟิลเตอริง (Packet Filtering) โดยจะมีการตรวจสอบแพ็คเก็ตด้วยฟังก์ชั่นฟิลเตอร์ว่าจะอนุญาตหรือไม่อนุญาตให้แพ็คเก็ตนั้นผ่านไปได้ โดยที่ฟังก์ชั่นฟิลเตอร์จะมีการกำหนดลำดับไว้แล้วให้ระบบเรียกฟังก์ชั่นฟิลเตอร์เข้าไปทำงานทีละฟังก์ชั่นตามลำดับ จนกว่าจะมีฟังก์ชั่นใดฟังก์ชั่นหนึ่งส่งค่ากลับเป็น "DROP PACKET" ถ้าฟังก์ชั่นทั้งหมดส่งค่ากลับเป็น "ALLOW PACKET" แพ็คเก็ตเหล่านั้นจะสามารถผ่านไปได้



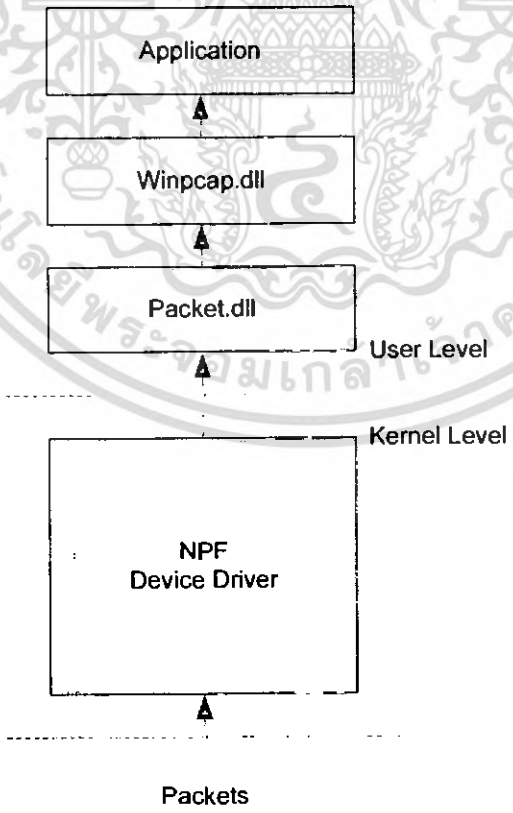
รูปที่ 6.4 แสดงขั้นตอนการทำงานของไฟร์วอล

ขั้นตอนการทำงานของไฟร์วอลตามรูปที่ 6.4

1) เครื่องได้รับแพ็กเก็ตเข้ามา โดยที่ IP Driver มีฟังก์ชั่นฟิลเตอร์อยู่ตามที่กำหนดไว้
2) IP Driver จะส่งแพ็คเก็ตนั้นเข้าไปโดยผ่านเข้าไปยังฟังก์ชั่นฟิลเตอร์ตามลำดับไปเรื่อยๆ แล้วรอค่าที่ส่งกลับออกมา
3) สมมุติฟังก์ชั่นแรกส่งค่ากลับเป็น "ALLOW PACKET" เมื่อ IP DRIVER ได้รับค่าส่งกลับจากฟังก์ชั่นแรกเป็น "ALLOW PACKET" ดังนั้น IP DRIVER จะส่งแพ็คเก็ตนั้นไปที่ฟังก์ชั่นที่สองต่อไปในกรณีนี้
4) สมมุติให้ฟังก์ชั่นที่สองนี้ส่งค่ากลับเป็น "DROP PACKET" เมื่อ IP DRIVER ได้รับค่าส่งกลับจากฟังก์ชั่นที่สองเป็น "DROP PACKET" ดังนั้น IP DRIVER จะไม่ส่งแพ็กเก็ตนี้ต่อไปยังระบบ และจะไม่ส่งไปยังฟังก์ชั่นต่อไปอีก

6.4.1.2 <u>ระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์</u>

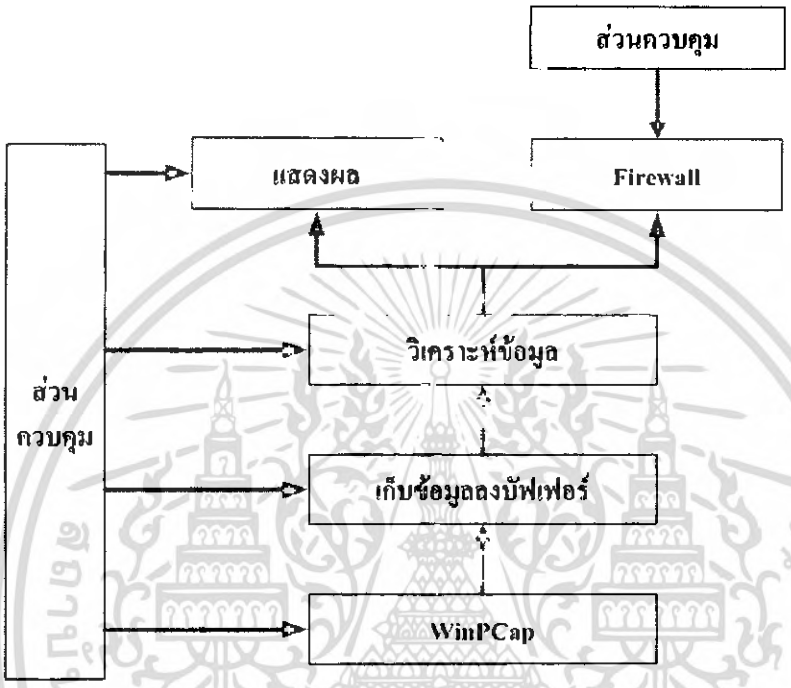
ระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์จะอาศัยคุณสมบัติ และ ความสามารถของ WinPCap ซึ่งเป็นไลบรารี่ที่ทำการติดต่อกับการ์ดแลน เพื่อทำการควบคุมการทำงานของการ์ดแลน และตรวจจับแพ็คเก็ต เพื่อนำมาวิเคราะห์ผลว่ามีการโจมตีเกิดขึ้นหรือไม่



รูปที่ 6.5 แสดงระดับชั้นการทำงานของ WinPCap

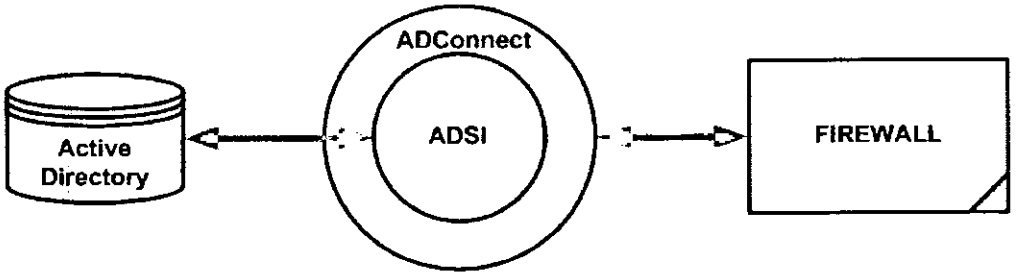
การทำงานของระบบตรวจจับผู้บุกรุกนั้นจะแบ่งการทำงานออกเป็น 2 ส่วนหลักๆ ด้วยกัน คือ ส่วนการดักจับแพ็คเก็ตเก็บลงบัฟเฟอร์ และส่วนวิเคราะห์ผลแพ็คเก็ตที่ได้ดักจับมา เมื่อวิเคราะห์ผลเสร็จจะแสดงผล พร้อมกับส่งแพ็คเก็ตไปฟิลเตอร์ตามกฎที่กำหนดเอาไว้ดังรูปที่ 6.6

โดยการทำงานทั้ง 2 ส่วนนี้จะมีการแบ่งเธรด (Thread) การทำงานเพื่อที่จะสามารถดักจับแพ็คเก็ตและวิเคราะห์ผลไปได้พร้อมๆกัน



รูปที่ 6.6 โครงสร้างของระบบตรวจจับผู้บุกรุกทำงานร่วมกับไฟร์วอล

เนื่องจากเพอร์ซันนอลไฟร์วอล (Personal Firewall) จะมีความสามารถในการที่จะรับกฎเพื่อเริ่มการทำงานของไฟร์วอลจากเครื่องโดเมนคอนโทลเลอร์ที่ได้ติดตั้งแอ็คทีฟไดเร็คทอรีไว้และจะมีการส่งล็อกไฟล์ไปเก็บไว้ยังแอ็คทีฟไดเร็คทอรีเมื่อมีการตรวจพบการโจมตี หรือ มีสิ่งผิดปรกติเกิดขึ้น ซึ่งโครงสร้างการทำงานของส่วนที่ใช้ในการติดต่อกับเครื่องเซิร์ฟเวอร์แสดงดังรูปที่ 6.7 โดยเพอร์ซันนอลไฟร์วอลจะมีการติดต่อไปยังเซิร์ฟเวอร์โดยผ่านคลาส ADConnect ที่สร้างขึ้น



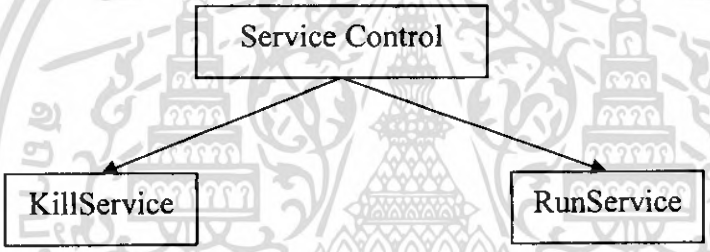
รูปที่ 6.7 แสดงโครงสร้างส่วนติดต่อกับเซิร์ฟเวอร์

เพอร์ซันนอลไฟร์วอลจะติดตั้งอยู่ที่เครื่องลูกข่ายทุกเครื่องในระบบโดยมีหน้าที่การทำงานดังนี้

- เป็นไฟร์วอลชนิดแพ็คเก็ตฟิลเตอริ่ง (Packet Filtering Firewall)
- ตรวจจับผู้บุกรุกทางเครือข่ายที่เข้ามาที่เครื่องลูกข่าย รวมทั้งสิ่งผิดปรกติที่ไม่เป็นไปตามมาตรฐานของโพรโตคอลทีซีพี (TCP) , ยูดีพี (UDP), ไอพี (IP), ไอซีเอ็มพี (ICMP)
- ตรวจการเชื่อมต่อเครือข่ายของโพรเซสต่างๆ โดยผู้ใช้สามารถจบโพรเซสการทำงานและปิดการเชื่อมต่อเครือข่ายของโพรเซสใดๆได้
- ส่งการแจ้งเตือนกลับไปยังล็อกมอนิเตอร์เมื่อตรวจจับได้ว่ามีการบุกรุกเกิดขึ้น

6.4.2 โปรแกรมไฟร์อลาร์มเซอร์วิสคอนโทรล (FireAlarm Service Control)

ในส่วนของเซอร์วิสคอนโทรลทำหน้าที่ควบคุมการทำงานของเซอร์วิส ประกอบด้วย KillService ซึ่งทำหน้าที่ยกเลิกการทำงานของเซอร์วิส และในส่วน RunService ทำหน้าที่เริ่มต้นการทำงานของเซอร์วิส



รูปที่ 6.8 รูปแสดงส่วนประกอบของ Service Control

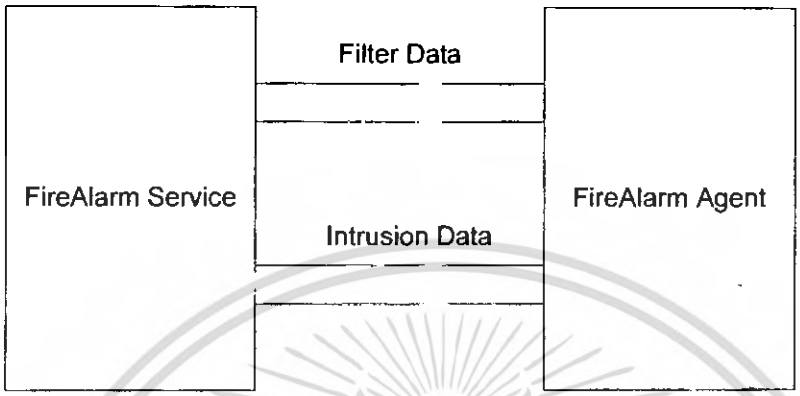
และในการทำงานเมื่อโปรแกรมไฟร์อลาร์มต้องการที่จะทำการอัพเดตข้อมูลใหม่ๆ โปรแกรมไฟร์อลาร์มเซอร์วิสคอนโทรลจะทำการ Refresh เซอร์วิสทำให้ตัวเซอร์วิสมีการดึงกฎใหม่ๆเข้ามาฟิลเตอร์ โดยการ Refresh จะทำการ KillService ที่ชื่อ “firealarmservice” จากนั้นก็ทำการ RunService อีกครั้ง โดยในการทำ Refresh โปรแกรมไฟร์อลาร์มเซอร์วิสคอนโทรลจะทำการตรวจสอบการล็อกอินเข้าใช้ระบบ และทำการ Refresh เซอร์วิส

6.4.3 โปรแกรมไฟร์อลาร์มเอเจนต์ (Firealarm Agent)

6.4.3.1 <u>ส่วนการรับและแสดงผลข้อมูลที่ได้จากไฟร์อลาร์มเซอร์วิส</u>

ในการรับส่งข้อมูลระหว่างโปรแกรมไฟร์อลาร์มเซอร์วิสกับไฟร์อลาร์มเอเจ็นต์จะใช้ IPC (Inter Process Communication) ทำการส่งข้อมูล โดยจะต้องทำการสร้าง named pipe server ที่โปรแกรมไฟร์อลาร์มเซอร์วิสและสร้าง named pipe client ที่โปรแกรมไฟร์อลาร์มเอเจ็นต์ เพื่อทำการรับส่งข้อมูล

โดยในโปรแกรมไฟร์อลาร์มเอเจนต์จะรับข้อมูล 2 อย่างจากไฟร์อลาร์มเซอร์วิส คือข้อมูลกฎการฟิลเตอร์และข้อมูลการบุกรุก ในการติดต่อจะทำการสร้าง pipe ขึ้นมา 2 อัน เพื่อทำการส่งข้อมูลกฎการฟิลเตอร์และข้อมูลการบุกรุก



รูปที่ 6.9 รูปแสดงส่วนการรับและแสดงผลข้อมูลที่ได้จากไฟร์อลาร์มเซอร์วิส

6.4.3.2 <u>ระบบตรวจการใช้งานเครือข่ายของโพรเซส</u>

ระบบตรวจการใช้งานเครือข่ายของโพรเซสจะทำงานคล้ายกับคำสั่ง netstat –bn โดยจะแสดงโพรเซสต่างๆ ที่กำลังใช้งานเครือข่ายอยู่ในขณะนั้น พร้อมทั้งบอกโพรเซสไอดี (PID) โพรโตคอล (Protocol) โลคอลแอดเดรส (Local Address) รีโมตแอดเดรส (Remote Address) และสถานะของการเชื่อมต่อ (State) โดยที่จะมีเวลาการอัพเดตข้อมูลตามเวลาที่ผู้ใช้ได้ตั้งเอาไว้

ระบบตรวจการใช้งานเครือข่ายของโพรเซสนี้มีความสามารถที่จะจบการทำงานของโพรเซส (Kill Connection) และ ปิดการเชื่อมต่อเครือข่ายของโพรเซสนั้นได้ (Close Connection) ถ้าหากพบว่ามีโพรเซสใดที่มีการใช้งานเครือข่ายที่น่าสงสัย ผู้ใช้สามารถจบการทำงานของโพรเซสหรือปิดการเชื่อมต่อได้ทันที เพื่อทำให้ผู้ใช้สามารถป้องกันตนเองได้อีกทางหนึ่ง

บทที่ 7
การพัฒนาระบบไฟร์แทรป

7.1 แนวคิด

ไฟร์แทรป (FireTrap) เป็นระบบจำลองทำหน้าที่เป็นเครื่องจำลองที่จำลองตัวเองให้เหมือนเครื่องให้บริการจริงมากที่สุด โดยอาจมีการเปิดบริการหรือช่องโหว่ต่าง ๆ เพื่อให้ผู้บุกรุกหลงเชื่อและแสดงพฤติกรรมในการบุกรุกต่าง ๆ ซึ่งพฤติกรรมเหล่านั้นจะถูกบันทึกไว้ และส่งกลับไปให้ยังส่วนเก็บข้อมูลและส่วนแสดงผลกลาง นอกจากนี้จะมีการตรวจสอบสภาพและความเสียหายของเครื่องกับดักอยู่ตลอดเวลา ถ้าถึงจุดที่เครื่องกับดักอ่อนแอ ผู้ดูแลสามารถที่จะระงับการใช้เครื่องกับดักนั้นโดยอัตโนมัติ

7.2 ขอบเขตความสามารถ

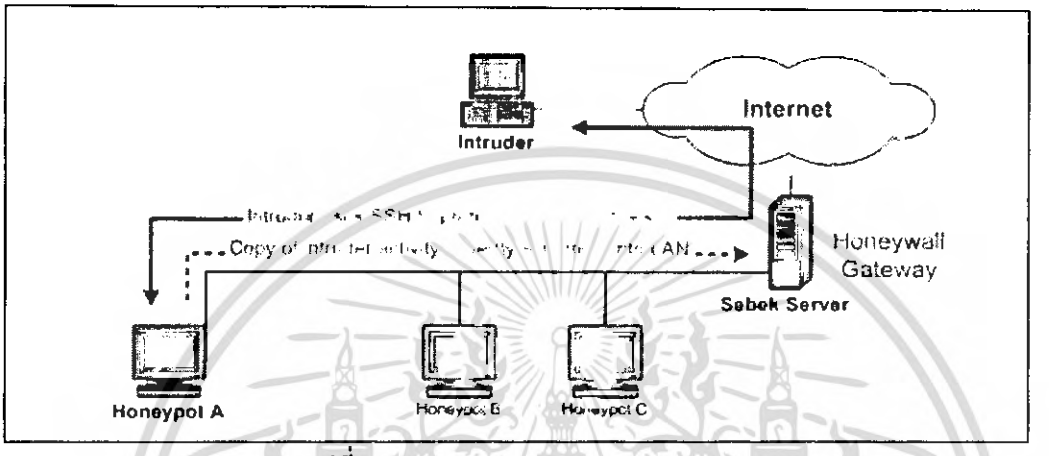
ไฟร์แทรป (FireTrap) เป็นระบบจำลองซึ่งแพ็คเก็ตที่ถูกระบบไฟร์สกรีนจำแนกว่าเข้าข่ายผู้บุกรุกจะถูกส่งมาที่ส่วนนี้ และทำการหลอกล่อให้ผู้บุกรุกเข้าใจผิดว่าเป็นเครื่องที่ให้บริการจริง โดยจะจำลองการบริการต่าง ๆ ให้เหมือนจริงมากที่สุด พร้อมทั้งเปิดบริการหรือช่องโหว่ เพื่อให้ผู้บุกรุกหลงเชื่อและแสดงพฤติกรรมในการบุกรุกต่าง ๆ ซึ่งพฤติกรรมเหล่านั้นจะถูกบันทึกจะส่งไปเก็บไว้ยังส่วนเครื่องแม่ข่ายฐานข้อมูล และจะมีการตรวจสอบสภาพและความเสียหายของเครื่องกับดักในไฟร์แทรปตลอดเวลาจากโปรแกรม S2I ซึ่งอยู่ในส่วนของไฟร์สกรีน

เครื่องกับดักในระบบไฟร์แทรปนี้ ทางผู้พัฒนาได้พัฒนาให้มีความปลอดภัยต่อองค์กร นั่นคือผู้บุกรุกจะไม่สามารถใช้เครื่องกับดักไปเป็นฐานโจมตีเครื่องอื่นๆ และพัฒนาให้มีความแนบเนียนเสมือนว่าเป็นเครื่องให้บริการจริงที่เป็นเป้าหมายของผู้บุกรุก โดยปกปิดความผิดปรกติของค่าต่างๆ ที่ระบุว่าเป็นเครื่องกับดัก เช่น ไอพีแอดเดรส แมคแอดเดรส หรือไม่พบเซอร์วิสที่ทำให้ทราบว่าได้มีการเฝ้าดูการกระทำต่างๆ โดย ทุก ๆ keystroke ที่ผู้บุกรุกสั่งงานเข้ามา แม้ผู้บุกรุกจะใช้คำสั่งผ่านการเข้ารหัสก็ตาม หรือแม้แต่การนำเข้าไฟล์หรือส่งออก ตัวเครื่องไฟร์แทรปจะส่งข้อมูลทุกอย่างไปเก็บไว้ในเครื่องแม่ข่ายฐานข้อมูล ซึ่งการส่งข้อมูลออกไปจะส่งไปแบบเข้ารหัสและไร้ตัวตน โดยที่ผู้บุกรุกไม่ทราบว่าได้มีการส่งข้อมูลออกไป ด้วยเทคนิคของ Sebek ซึ่งทำงานในแบบ kernel-base นั่นคือแม้ว่าผู้บุกรุกจะนำโปรแกรมดักจับแพ็คเก็ตมาติดตั้งก็จะไม่พบว่าเครื่องไฟร์แทรปได้ส่งข้อมูลออกไป

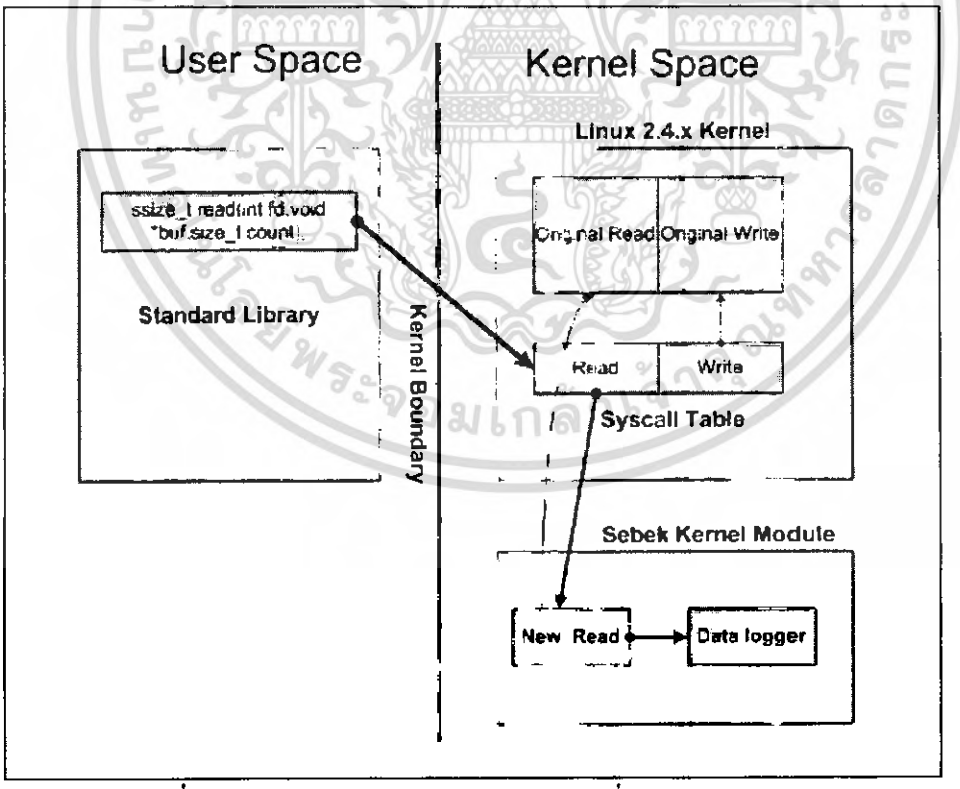
7.3 เทคโนโลยีที่ใช้ในการพัฒนาโปรแกรม

7.3.1 เซเบค (Sebek)

เซเบคไคลเอนต์จะคอยซ่อนตัวอยู่ในกับดักเพื่อเฝ้าดูพฤติกรรมและการกระทำของผู้บุกรุก และคอยส่งข้อมูลนั้นไปยังเครื่องที่ใช้เก็บพฤติกรรมลงฐานข้อมูล โดยส่งผ่านช่องทางที่มีความปลอดภัย และมีการเข้ารหัสข้อมูลในรูปแบบของเซเบคเอง

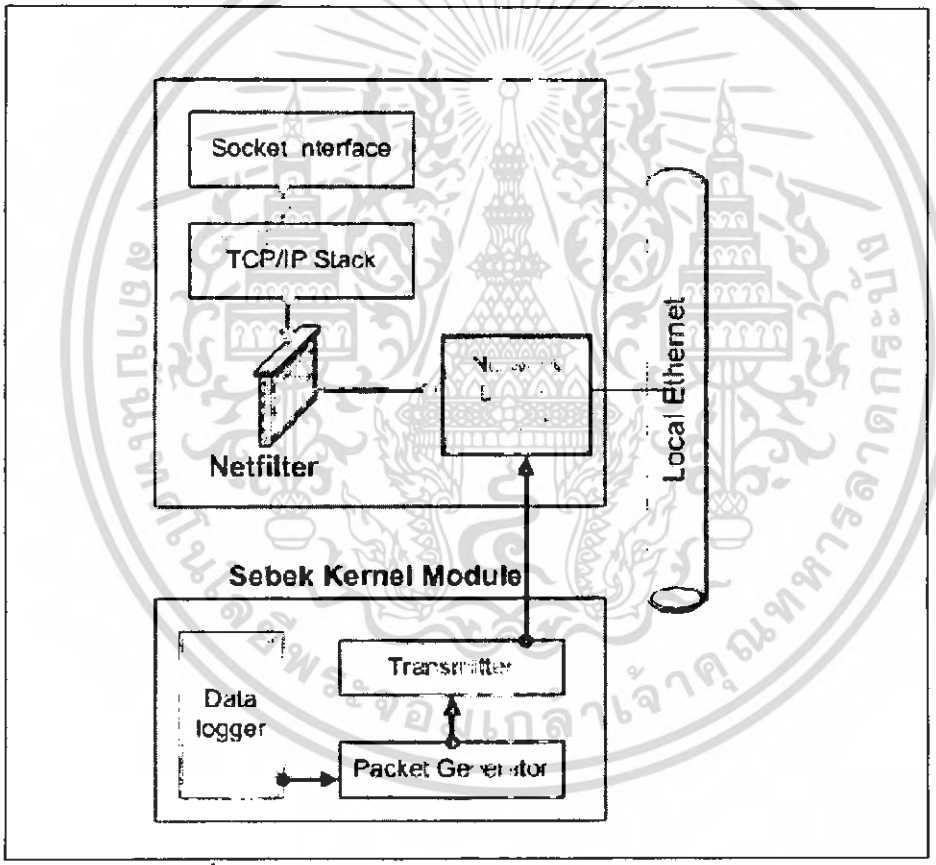


รูปที่ 7.1 แสดงการทำงานของ Sebek



รูปที่ 7.2 รูปภาพแสดงการcapture ข้อมูลที่ทำภายใน kernel

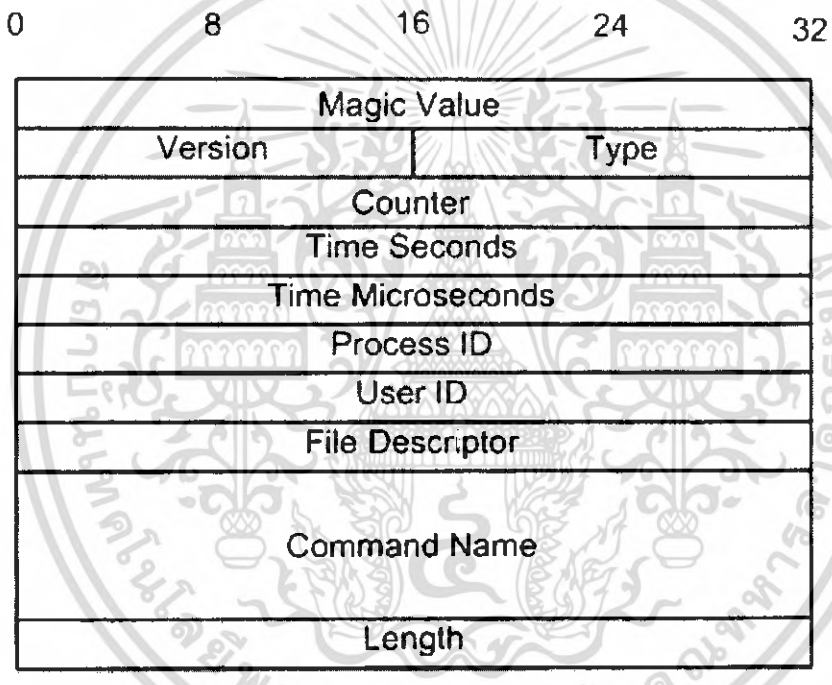
เซเบคเป็น Data Capture Tool แบบ Kernel-based โดยในการนำมาใช้จะประยุกต์ใช้เป็นแบบ LKM (Loaded Kernel Module) มีความโดดเด่นในความสามารถในการดักจับข้อมูลได้ทุกการกระทำของผู้บุกรุกที่กระทำผ่าน read() system call ไม่ว่าจะเป็นในส่วนของ keystroke, files transfer, burneye passwords หรือหากมีการใช้ IRC client หรือ e-mail client ตัวเซเบคก็สามารถดักจับได้ ดังเช่นเมื่อผู้บุกรุกมีการนำไฟล์ส่งเข้ามา ตัวเซเบคจะสามารถอ่านและบันทึกไฟล์นั้นไว้ได้ ทั้งยังสามารถใช้เป็น Glass-box เพื่อเปรียบเทียบกับการทำงานของ Black-box ที่เรารู้จัก นั้นคือเราสามารถใช้ติดตามการทำงานของโปรแกรมที่ผู้บุกรุกเอามารันได้ด้วย แม้ว่าผู้บุกรุกจะออกจากระบบไปแล้วก็ตาม นอกจากนั้นยังสามารถบันทึกพฤติกรรมของผู้บุกรุกที่กระทำผ่านทางช่องทางที่มีการเข้ารหัส เช่น SSH ได้ด้วย



รูปที่ 7.3 รูปภาพแสดงการby-passไม่ผ่าน TCP stack

เซเบคมีรูปแบบวิธีการปิดซ่อนตัวเองได้อย่างมีประสิทธิภาพ และค่อนข้างยากที่จะตรวจสอบพบทั้งจากภายในและภายนอก โดยจะทำการปิดซ่อนโมดูลที่ใช้ และลบ linked list ที่จะแสดงการมีตัวตนออก โดยใช้โมดูลที่ชื่อ the cleaner ทำให้ผู้บุกรุกไม่สามารถตรวจสอบพบว่าเซเบคกำลังทำงานอยู่ภายในเครื่อง ตัวเซเบคจะมีฟังก์ชั่นในการสร้างและส่งแพ็คเก็ตของเซเบคเองโดยไม่ใช้ TCP/IP stack ดังนั้นจึงไม่สามารถถูกมองเห็นและไม่สามารถทำการ block traffic ของ

เซเบคได้ หลังจากเซเบคสร้างแพ็คเก็ตเสร็จ แพ็คเก็ตจะถูกส่งไปยัง device driver โดยตรง ไม่ผ่าน raw socket socket interface ซึ่งทำให้พวกโปรแกรม sniffer ที่ใช้ libpcap เป็นฐานไม่สามารถเห็นได้ เพราะมันไปดักจับข้อมูลที่ raw socket interface และเพื่อป้องกันการตรวจจับในเครือข่าย เซเบคจะไม่ใช้ ARP แต่จะกำหนด MAC Address ของเซิร์ฟเวอร์สำหรับเก็บพฤติกรรมผู้บุกรุกไว้เลย เพื่อจะได้ไม่ต้องทำการ ARP request เพื่อป้องกันการทำ ARP spoofing ซึ่งจะลดการดักจับข้อมูลในเครือข่ายได้ ทั้งนี้เซเบคยังมีรูปแบบการส่งข้อมูลด้วยโพรโตคอลของตัวเองซึ่งใช้การส่งแบบโพรโตคอล UDP ในการสื่อสารระหว่างเซเบคไคลเอนต์และเซเบคเซิร์ฟเวอร์ โดยจะเป็นการสื่อสารทางด้านเดียว คือ ส่งข้อมูลจากเซเบคไคลเอนต์ไปยังเซเบคเซิร์ฟเวอร์เท่านั้น โพรโตคอลของเซเบคมีรูปแบบดังต่อไปนี้



รูปที่ 7.4 รูปแสดงโพรโตคอลของเซเบค

ค่า Process ID , User ID และ File Descriptor จะเป็นค่าของโพรเซสที่ทำการส่งสัญญาณ read () system call นั้น ๆ จากเคอร์เนลมาให้ ฟิลด์ Length จะแปรเปลี่ยนไปตามความยาวข้อมูลที่ส่งมาโดยถ้าเป็นข้อมูลที่ส่งมาจากฟังก์ชั่น read () จะยาวกว่าข้อมูลทั่ว ๆ ไป จากนั้นเซเบคจะตัดเอาข้อมูลเฉพาะข้อมูลที่เกี่ยวกับเรคคอร์ดของเซเบคมาประมวลผลเท่านั้น

ภายในตัวเซเบคไคลเอนต์จะมีการตัดสินใจว่าจะซ่อนแพ็คเก็ตเพื่อป้องกันการดักจับจาก user space หรือไม่ โดยพิจารณาจากว่าแพ็คเก็ตนั้นๆ ต้องเป็นแพ็คเก็ต UDP มีหมายเลขพอร์ตปลายทางและหมายเลขในฟิลด์ Magic value ที่ตรงกับค่าที่กำหนดไว้ในโปรแกรมเซเบค ซึ่งค่า Magic value นี้จะช่วยป้องกันการ brute force ในการดักจับแพ็คเก็ตได้

7.3.2 เซมเฮน (Samhain)

เซมเฮน เป็นระบบที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลและใช้ในการแจ้งเตือนการบุกรุกระบบซึ่งสามารถใช้บนเครื่องโฮสต์เครื่องเดียว หรือใช้ในระบบเครือข่ายขนาดใหญ่ได้ ซึ่งในโครงงานนี้ไม่ได้ใช้กับเครื่องโฮสต์เพียงเครื่องเดียว หากแต่จะใช้กับเครื่องหลายๆ เครื่องที่อยู่ในระบบเครือข่ายโดยจะมีเครื่องแม่ข่ายที่จะทำหน้าที่เป็นศูนย์กลางคอยรับข้อมูลจากเครื่องลูกข่ายทั้งหลายในระบบเครือข่าย ซึ่งจะได้อธิบายรายละเอียดของโครงสร้างและการทำงานในส่วนถัดไป

7.3.2.1 การใช้งานในรูปแบบระบบเครือข่าย (network)

เซมเฮน เป็นระบบที่ถูกออกแบบมาให้ง่ายต่อการมอนิเตอร์โฮสต์หลายๆ เครื่องในเครือข่าย ซึ่งจะประกอบไปด้วยโพรเซสที่ทำงานในลักษณะที่เป็นเดมอนในโฮสต์แต่ละเครื่อง และจะมีเครื่องแม่ข่ายกลางที่จะคอยเก็บบันทึกหรือรายงานต่างๆ จากเดมอนเหล่านั้นผ่านทางการเชื่อมต่อแบบทีซีพี/ไอพี ซึ่งในการส่งข้อมูลจากเครื่องลูกข่ายต่างๆ ไปให้เครื่องแม่ข่ายกลางนั้นจะต้องมีการยืนยันตนก่อนเพื่อป้องกันการปลอมแปลงข้อความที่ส่งไปให้เครื่องแม่ข่าย โดยในขั้นแรกจะต้องตกลงกันในเรื่องของโพรโตคอลที่จะใช้ในการยืนยันตนก่อน จากนั้นจะมีการแลกเปลี่ยน session key กัน การเชื่อมต่อที่มาจากเครื่องโฮสต์ที่ไม่ได้ทำการลงทะเบียนไว้จะถูกดรอปทิ้งทันที และในอีกกรณีหนึ่งที่จะถูกดรอปคือการเชื่อมต่อจากเครื่องโฮสต์ที่ได้มีการลงทะเบียนไปแล้ว แต่เครื่องลูกข่ายไม่สามารถทำการยืนยันตนได้สำเร็จ เมื่อ Session key ได้ถูกสร้างขึ้นมาเครื่องลูกข่ายจะใช้ session key นั้นในการเซ็นลงไปในข้อความของตนเพื่อใช้ในการยืนยันตนและเมื่อข้อความถูกส่งไปถึงเครื่องแม่ข่าย เครื่องแม่ข่ายจะทำการตรวจสอบลายเซ็นของเครื่องลูกข่าย จากนั้นเมื่อตรวจสอบพบว่าถูกต้องแล้วเครื่องแม่ข่ายจะเอาลายเซ็นนั้นๆ ออกและนำเอาลายเซ็นของตนเซ็นลงไปเมื่อจะเก็บข้อความนั้นลงในล็อกไฟล์ ทั้งคอนฟิกูเรชันไฟล์และฐานข้อมูลสามารถเก็บไว้ที่ส่วนกลางบนฝั่งของเครื่องแม่ข่ายได้และสามารถดาวน์โหลดไปใช้โดยเครื่องลูกข่ายในขณะที่ทำการ startup เครื่องได้อีกด้วย

7.3.2.2 การไม่แสดงตัวตนว่ามีการใช้งานอยู่ (stealth)

เซมเฮน มีความสามารถในการอำพรางตัวไม่ให้ผู้บุกรุกสามารถรู้ได้ว่ามีโพรเซสของเซมเฮนกำลังทำงานอยู่ในระบบซึ่งการทำงานในลักษณะนี้เรียกว่าทำงานแบบ stealth ซึ่งสามารถทำได้โดยการ คอมไพล์เซมเฮนให้สนับสนุนโหมด stealth แต่การทำงานในโหมด stealth นั้นก็ยังอาจสามารถถูกตรวจสอบได้จากการค้นหาคอนฟิกูเรชั่นไฟล์หรือค้นหาจากสตริง ดังนั้นเซมเฮนจึงมีออปชั่นที่เพิ่มเติมเข้ามาคือ

- สตริงที่สามารถอ่านได้ซึ่งอยู่ใน executable file, ล็อกไฟล์ และในฐานข้อมูลสามารถทำให้ดูไม่รู้เรื่องได้ตัวอย่างเช่น อาจทำให้มองดูแล้วเหมือนเป็นข้อมูลไบนารี่
- สามารถยกเลิกการใช้ command line parsing ได้

- สามารถทำข้อมูลคอนฟิคกูเรชั่นต่างๆ ให้เป็นรูปภาพโดยใช้ steganography
- Executable สามารถบีบอัดและเข้ารหัสลับได้
- ฐานข้อมูลและล็อกไฟล์อาจถูกนำไปซ่อนโดยการนำไปต่อท้ายภาพที่มีอยู่แล้วไต้โดยภาพนั้นยังคงดูเหมือนปรกติ

7.3.2.3 คุณสมบัติในการทำงานของ Samhain

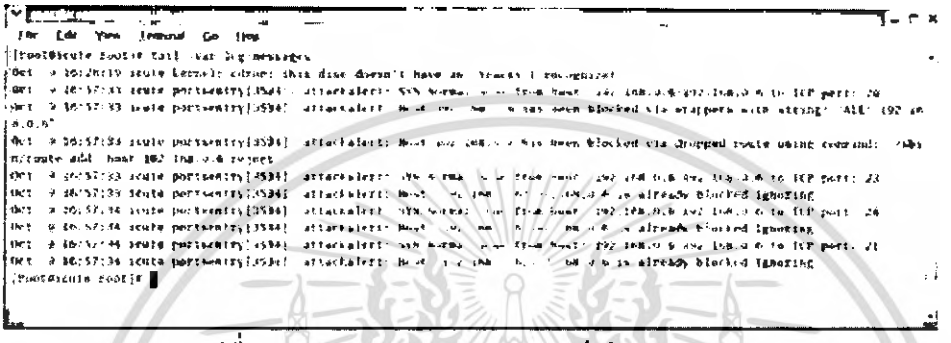
- สามารถ checksum ได้หลากหลายรูปแบบ เช่น TIGER192, SHA-I ,MD5 เป็นต้น โดยจะตรวจสอบได้จากหลาย ส่วน เช่น size, mode/permission, owner, group, creation/ modification/access time, inode, number of hardlinks, linked path(symbolic links) major/minor device number
- สามารถใช้ shell wildcard pattern ในการระบุไฟล์หรือไดเรกทอรี่ที่จะตรวจสอบได้
- สามารถตั้งและปรับแต่ง policy ของระบบได้ 8 รูปแบบที่แตกต่างกัน
- การตรวจสอบในเชิงลึก สามารถตั้งค่าให้ตรวจสอบทั้งหมดหรือเฉพาะส่วนได้
- สามารถตรวจสอบความเปลี่ยนแปลงของระบบได้ ว่ามีไฟล์ใดถูกแก้ไข หรือถูกลบไป
- สามารถตรวจสอบความถูกต้องของเคอร์เนลที่กำลังใช้งานอยู่ได้ เพื่อตรวจสอบรูทคิตได้
- สามารถตรวจสอบการเปลี่ยนแปลงหรือสร้าง SUID/SGID files โดยสามารถสั่งให้ตรวจสอบอย่างสม่ำเสมอตามเวลาที่กำหนดได้
- สามารถตรวจสอบการ mount ต่างๆ รวมทั้งการ mount files system
- สามารถตรวจสอบการ login/logoff ของผู้ใช้งานได้ โดยใช้ system file ที่ชื่อว่า utmp
- สามารถกำหนดให้มีการจัดเก็บและแสดงการเตือนได้
- เครื่องแม่ข่ายฐานข้อมูลจะได้รับข้อมูลผ่านทาง TCP connection ที่มีการเข้ารหัสลับไว้ และมีการพิสูจน์ตัวตนระหว่างกัน เพื่อให้แน่ใจได้ว่า เป็นตัวจริงทั้งคู่
- สามารถใช้งาน Syslog ได้
- สามารถใช้งานผ่านทางคอนโซลได้ถ้ามีการติดตั้งตัวเดมอนไว้
- ล็อกไฟล์ที่จัดเก็บจะมีการ Sign ไว้เพื่อป้องกันการแก้ไขของผู้ไม่มีสิทธิ์
- อีเมลล์ที่ใช้จะมีการ Sign ไว้เช่นเดียวกัน
- รองรับตัว RDBMS ที่หลากหลาย เช่น MySQL, PostgreSQL, และ Oracle

7.3.3 PortSentry

โปรแกรม PortSentry เป็นโปรแกรมสำหรับลีนุกซ์และโอเอสตระกูลยูนิกซ์ที่ออกแบบขึ้นเพื่อตรวจสอบการสแกนพอร์ตจากภายนอก และสามารถตอบสนองต่อการกระทำนั้น ๆ ได้หลายรูปแบบ พัฒนาขึ้นโดยเป็นส่วนหนึ่งของ Abacus Project ซึ่งริเริ่มโดย Craig H. Rowland โดย

ลักษณะแล้วถือว่าเป็นซอฟต์แวร์ประเภท Network based Intrusion Detection (NIDS) ซึ่งนอกจากจะมีโหมดการตรวจสอบการสแกนพอร์ตให้เลือกใช้งานถึง 6 โหมดแล้ว ยังสามารถโต้ตอบการสแกนพอร์ตนั้นได้หลายแบบอีกด้วย เช่น

- บันทึกข้อมูลของการสแกนพอร์ตที่เกิดขึ้น และรายงานผ่านบริการ syslogd ซึ่งโดยปรกติจะสามารถย้อนตรวจดูเหตุการณ์ย้อนหลังได้ที่ไฟล์ /var/log/messages หรืออาจจะเปลี่ยนไปบันทึกที่ไฟล์อื่นแยกออกไปก็ได้ตามต้องการ

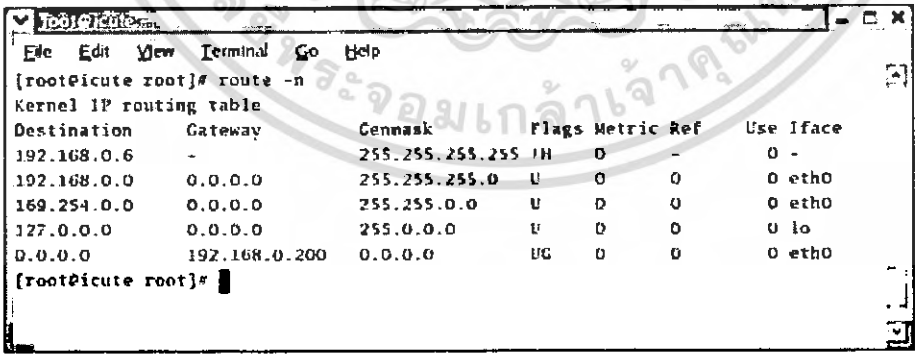


รูปที่ 7.5 รายงานการถูกสแกนพอร์ตโดย Syslogd

- ปิดกั้นโฮสต์ฝ่ายตรงข้ามโดยอัตโนมัติ หรือ Automatic Real time blocking ด้วย TCP Wrappers โดยจะอัพเดตที่ไฟล์ /etc/hosts.deny ให้เองทันที เช่น เมื่อถูกเครื่อง 203.182.16.1 สแกนเข้ามา ไฟล์ /etc/hosts.deny จะมีข้อความเกิดขึ้นเองดังนี้

ALL: 203.182.16.1

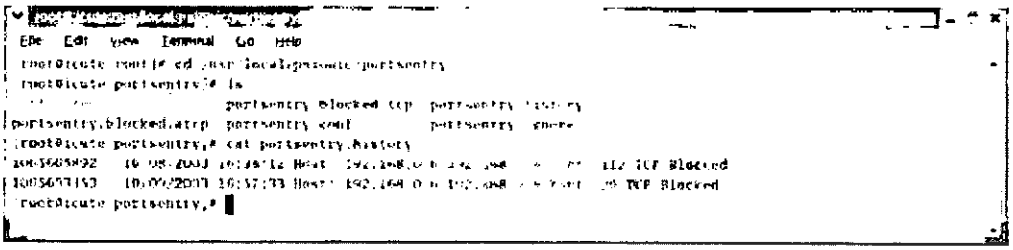
- คอนฟิก kernel route table โดยอัตโนมัติ เพื่อตัดการติดต่อกับโฮสต์ฝ่ายตรงกันข้าม



รูปที่ 7.6 Kernel Routing Table ภายหลังจากถูกเครื่อง 192.168.0.6 สแกนเข้ามา

- ระบบ History เป็นการบันทึกผลการตัดการติดต่อที่ได้กระทำไปแล้ว โดยมีข้อมูลเกี่ยวกับวันเวลาที่มีการสแกนพอร์ตเกิดขึ้น ชื่อและไอพีแอดเดรสของโฮสต์ฝ่ายตรง

ข้าม (สามารถกำหนดให้ใช้ระบบ DNS Lookup หรือไม่ใช้ได้ตามต้องการ) โดยจะเก็บไว้ที่ไฟล์ portsentry.history



รูปที่ 7.7 ไฟล์ History ของ PortSentry

- Prioritize Response เป็นความสามารถใหม่มีตั้งแต่เวอร์ชั่น 1.1 ขึ้นไป คือ สามารถจัดระดับความสำคัญของการจัดการต่อการสแกนพอร์ตได้ และยังสามารถรันคำสั่งภายนอกได้ตามต้องการ จึงมีประโยชน์มากต่อการสร้างระบบแจ้งเตือน (Alert and Alarm) ในรูปแบบที่ผู้ดูแลระบบสามารถกำหนดขึ้นได้เอง

ในการนำ portsentry มาใช้ในโครงงานนี้ จะต้องทำการแก้ไขค่าคอนฟิกใด ๆ ทั้งโปรแกรมไฟล์คอนฟิก และ ไฟล์ History โดยจะอยู่รวมกันที่ /usr/local/psionic/portsentry เมื่อต้องการเรียกใช้งานจะใช้คำสั่งดังนี้

 # ./portsentry –atcp

ซึ่งอาร์กิวเมนต์ –atcp จะเป็นการกำหนดให้ PortSentry ทำงานในโหมด Advanced Stealth Detection ซึ่งจะซ่อนการทำงานไว้ได้ แม้แต่ Localhost เองก็ไม่รู้ว่ามีการเปิด Socket จาก Portsentry ไว้ (โหมดนี้ทำงานได้เฉพาะ Linux Kernel เท่านั้น หากนำไปรันบนระบบอื่น ๆ จะไม่สามารถทำงานได้)

7.4 การทำงานของโปรแกรม

ไฟร์แทรปจะรอรับการติดต่อจากเซอร์วิสของโปรแกรม S2I ที่รันอยู่บนไฟร์สกรีน ผ่านช่องทาง SSH เพื่อให้ไฟร์สกรีนตรวจสอบสภาพความอ่อนแอของตัวเครื่องกับดักในไฟร์แทรป และยังรอรับเส้นทางที่จะถูกสร้างขึ้นจาก S2I เพื่อให้ผู้บุกรุกเข้ามา เมื่อผู้บุกรุกถูกส่งเข้ายังเครื่องไฟร์แทรปแล้ว โปรแกรมเซเบคไคลเอ็นต์ที่อยู่ในไฟร์แทรป จะทำการบันทึกพฤติกรรมต่าง ๆ ที่ผู้บุกรุกกระทำต่อไฟร์แทรป เช่น keystroke , ไฟล์ต่าง ๆ ที่ผู้บุกรุกนำเข้ามา หรือนำออกไป

นอกจากนี้เพื่อความแนบเนียนในการซ่อนตัวจากผู้บุกรุกของไฟร์แทรป จึงมีการสร้างสคริปท์ในการปลอมไอพีแอดเดรสของเครื่องกับดัก ให้เป็นไอพีแอดเดรสของเครื่องเว็บเซิร์ฟเวอร์จริง เช่น สคริปหลอกเมื่อใช้คำสั่ง ifconfig , traceroute หรือ ping เพื่อให้ผู้บุกรุกไม่สงสัยว่าตนเองอยู่ในเครื่องกับดัก

บทที่ 8
การพัฒนาระบบไฟร์เบรก

8.1 แนวคิด

ไฟร์เบรก (FireBreak) ทำหน้าที่เป็นซีเคียวรีเวิร์สเว็บพร็อกซี่ให้กับระบบเพื่อเพิ่มความปลอดภัยให้กับเว็บเซิร์ฟเวอร์ที่อยู่ภายในโซน DMZ และเป็นปราการด่านสุดท้ายที่ผู้บุกรุกจะต้องเจอก่อนที่จะเข้าถึงเว็บเซิร์ฟเวอร์ของเรา โดยในโครงงานได้ใช้ความสามารถของโปรแกรม Squid-cache ในการทำงานเป็นรีเวิร์สเว็บพร็อกซี่ ทำให้เมื่อผู้บุกรุกสร้างการเชื่อมต่อไปยังเว็บเซิร์ฟเวอร์จะต้องผ่านไฟร์เบรกก่อนทุกครั้ง ซึ่งไฟร์เบรกจะมีระบบตรวจจับการบุกรุกอยู่ภายในเพื่อทำการตรวจสอบการเชื่อมต่อที่เข้ามายังไฟร์เบรกว่าเข้าข่ายเป็นผู้บุกรุกหรือไม่ โดยในโครงงานนี้ได้ใช้โปรแกรมสนอร์ตอินไลน์ เป็นเครื่องมือในการตรวจสอบ

8.2 ขอบเขตความสามารถ

ไฟร์เบรกสามารถตรวจสอบแพ็คเก็ตที่จะเข้าสู่เว็บเซิร์ฟเวอร์ที่อยู่ข้างในได้ โดยผู้ที่ต้องการติดต่อกับเว็บเซิร์ฟเวอร์จะไม่สามารถติดต่อกับเว็บเซิร์ฟเวอร์ได้โดยตรงแต่ต้องติดต่อผ่านไฟร์เบรกก่อน ดังนั้นจึงไม่สามารถโจมตีเว็บเซิร์ฟเวอร์โดยตรงได้ แต่การโจมตีจะเกิดขึ้นกับตัวเครื่องไฟร์เบรกนี้แทน ซึ่งไฟร์เบรกนี้จะอาศัยโปรแกรม Squid-cache ในการทำหน้าที่เป็นซีเคียวรีเวิร์สเว็บพร็อกซี่ให้กับเว็บเซิร์ฟเวอร์ และเนื่องจากทุก ๆ การเชื่อมต่อที่จะติดต่อไปยังเว็บเซิร์ฟเวอร์จะต้องติดต่อผ่านทางไฟร์เบรกนี้ทุกครั้ง ซึ่งภายในไฟร์เบรกจะมีระบบตรวจจับผู้บุกรุกทำงานอยู่ทำให้ทุก ๆ แพ็คเก็ตที่ร้องขอเข้ามาจะต้องผ่านการตรวจสอบโดยโปรแกรมสนอร์ตอินไลน์ เพื่อแยกแยะผู้บุกรุกออกจากผู้ใช้งานจริง และหากการเชื่อมต่อหรือแพ็คเก็ตใด ๆ เข้าข่ายเป็นผู้บุกรุก ระบบไฟร์เบรกจะทำการตัดการเชื่อมต่อนั้นทันทีเพื่อป้องกันการโจมตีหรือการบุกรุกต่างๆ ที่อาจจะเกิดขึ้นได้

เนื่องจากในการบุกรุกทางเว็บแอพพิลเคชั่นนั้น มีรูปแบบและวิธีการใหม่ ๆ อยู่เสมอทำให้ต้องมีการอัพเดต Signature ของสนอร์ตอินไลน์ อยู่เสมอเพื่อให้สามารถตรวจจับการบุกรุกรูปแบบใหม่ ๆ ได้ ทางผู้พัฒนาจึงได้เพิ่มความสามารถในการอัพเดต Signature ของสนอร์ตอินไลน์ที่ทำงานอยู่ในไฟร์เบรกด้วย

8.3 เทคโนโลยีที่ใช้ในการพัฒนาโปรแกรม

8.3.1 โปรแกรม Squid

สำหรับความสามารถในการทำหน้าที่เป็นรีเวิร์สเว็บพร็อกซี่ของไฟร์เบรกนั้น จะอาศัยโปรแกรม Squid-cache ซึ่งเป็นโปรแกรมแบบเปิดเผยซอร์สโค้ด สำหรับใช้งานเป็นเว็บพร็อกซี่ ซึ่งสามารถนำมาประยุกต์ใช้เป็นรีเวิร์สเว็บพร็อกซี่สำหรับเว็บเซิร์ฟเวอร์ในโครงงานนี้ได้ โดยสามารถดาวน์โหลดซอร์สโค้ดของ Squid-cache ได้ที่เว็บ http://www.squid-cache.org ในที่นี้ใช้เวอร์ชั่น 2.5

Squid-cache มีลักษณะการตั้งค่าการใช้งานโดยทำการแก้ไขได้ที่ไฟล์ squid.conf ซึ่งโดยปรกติจะอยู่ที่ /etc/squid/squid.conf ซึ่งเป็นไฟล์สำหรับการตั้งค่าต่าง ๆ ของโปรแกรม squid-cache ซึ่งมีการอธิบายการใช้งานของแต่ละคำสั่งอยู่แล้ว ในโครงงานนี้เพื่อการใช้งานเป็นรีเวิร์สเว็บพร็อกซี่ จึงต้องทำการตั้งค่าในไฟล์ squid.conf ดังต่อไปนี้

```
http_port 80
visible_hostname 172.16.144.131
httpd_accel_host 172.16.144.131
httpd_accel_port 80
http_access allow all
```

คำอธิบายในการใช้งานส่วนต่าง ๆ ของไฟล์ squid.conf

- http_port - ใช้กำหนดพอร์ตที่ใช้ในการรับการติดต่อที่เข้ามายังไฟร์เบรก ซึ่งทำหน้าที่เป็นพร็อกซี่ โดยค่าเริ่มต้นจะมีค่าเป็น 3128 ดังนั้นเพื่อทำหน้าที่เป็นรีเวิร์สเว็บพร็อกซี่จึงต้องกำหนดค่าเป็น 80 ซึ่งเป็นพอร์ตทั่วไปในการใช้งาน HTTP
- visible_hostname - ใช้สำหรับระบุโฮสต์ที่เป็นเว็บเซิร์ฟเวอร์เมื่อเกิด error_message ในการทำงานของ Squid กรณีที่ไม่สามารถระบุได้ว่าโฮสต์ที่เป็นเว็บเซิร์ฟเวอร์คือเครื่องใด
- httpd_accel_host - ใช้สำหรับระบุโฮสต์ที่ทำหน้าที่เป็นเว็บเซิร์ฟเวอร์จริงที่อยู่ข้างหลังไฟร์เบรกเพื่อทำให้มีคุณสมบัติเป็น Transparent Proxy ซึ่งในที่นี้คือโฮสต์ที่มีไอพีแอดเดรสเป็น 172.16.144.131
- httpd_accel_port - ใช้ระบุพอร์ตที่ใช้งานของเว็บเซิร์ฟเวอร์จริง ในการทำเป็น Transparent Proxy
- http_access - ใช้ระบุสิทธิ์ผู้ที่สามารถใช้งานรีเวิร์สเว็บพร็อกซี่ไฟร์เบรกนี้ได้ โดยค่าเริ่มต้นจะเป็น deny all แต่เนื่องจากการใช้งานเป็นรีเวิร์สเว็บพร็อกซี่นี้จะรองรับบริการทั้งจากเครือข่ายภายในและจากอินเตอร์เน็ตซึ่งจะต้องเข้าใช้งานได้หมด จึงต้องเปลี่ยนค่าเป็น allow all

8.3.2 โปรแกรมสนอร์ตอินไลน์

สำหรับความสามารถตรวจจับผู้บุกรุกที่เข้ามายังไฟร์เบรกจะโปรแกรมสนอร์ตอินไลน์ โดยใช้ความสามารถในการเป็น Intrusion Prevention System (IPS) ในการตรวจจับแพ็คเก็ตที่เป็นอันตรายต่อเว็บเซิร์ฟเวอร์ของระบบ โดยถ้าไฟร์เบรกพบการเชื่อมต่อหรือแพ็คเก็ตใด ๆ ที่น่าสงสัยและมีพฤติกรรมเข้าข่ายผู้บุกรุก ตรงกับ Signature ของสนอร์ตอินไลน์ ไฟร์เบรกจะทำการดร็อปแพ็คเก็ตนั้นทิ้งทันที แล้วทำการส่งล็อกไฟล์ของการบุกรุกนั้น ๆ ไปแสดงผลที่โปรแกรมควบคุม Tartarus Management ต่อไป

ในโครงงานนี้ได้ใช้ซอร์สโค้ดของสนอร์ตอินไลน์ เวอร์ชั่น 2.3.0-RC1 ซึ่งสามารถดาวน์โหลดได้ที่http://prdownloads.sourceforge.net/snort-inline/snort_inline-2.3.0-RC1.tar.gz?download โดยหลังจากการติดตั้งแล้วให้ทำการแก้ไขไฟล์ snort_inline.conf ซึ่งโดยปรกติจะอยู่ที่ /etc/snort_inline/snort_inline.conf โดยแก้ไขข้อความดังต่อไปนี้

```
var RULE_PATH /etc/snort_inline/drop_rules
output alert_full : snort_inline-full
output alert_fast : snort_inline-fast
```

ให้เป็นข้อความดังต่อไปนี้

```
var RULE_PATH /etc/snort_inline/rules
output alert_full : snort_inline-full
output alert_fast : snort_inline-fast
```

8.3.3 โปรแกรม Oinkmaster

สำหรับการอัพเดต Signature หรือกฎของสนอร์ตอินไลน์ ในไฟร์เบรกจะใช้ความสามารถของโปรแกรม Oinkmaster ในการอัพเดต Signature ให้ทันสมัย สำหรับการติดตั้งและการใช้งานโปรแกรม Oinkmaster สามารถกระทำได้เหมือนดังที่กล่าวมาในส่วนการพัฒนาระบบไฟร์สกรีน

สำหรับกฎต่าง ๆ ที่สำคัญในการป้องกันเว็บเซิร์ฟเวอร์ได้แก่ bleeding-web.rules, bleeding-expliot.rules, bleeding-scan.rules, bleeding-attack response.rules ซึ่งสามารถดาวน์โหลดได้ที่ http://www.bleedingsnort.com/ หลังจากนั้นทำการปรับแต่งค่าต่าง ๆ ในไฟล์ต่อไปนี้

- ปรับแต่งค่าในไฟล์ /etc/oinkmaster.conf ดังนี้

```
url = http://www.bleedingthreats.net/bleeding.rules.tar.gz
modifysid * "^alert" | "drop"
```

url - เป็นส่วนในการกำหนด url ที่ใช้สำหรับดาวน์โหลด Signature

modifysid - เป็นส่วนสำหรับเปลี่ยนแปลงกฎทั้งหมดที่เป็น alert อยู่ ให้กลายเป็น drop

- ปรับแต่งค่าในไฟล์ /etc/snort_inline/snort_inline.conf ให้รองรับการทำงานของกฎใหม่ๆ เหล่านี้โดยเพิ่มการตั้งค่าในส่วนของ drop rules ในไฟล์ดังกล่าว ดังนี้

include $RULE_PATH/bleeding-web.rules

include $RULE_PATH/bleeding-exploit.rules

include $RULE_PATH/bleeding-scan.rules

include $RULE_PATH/bleeding-attack response.rules

8.4 การทำงานของโปรแกรม

สำหรับการทำงานของไฟร์เบรกเป็นดังต่อไปนี้ คือเมื่อทำการตั้งค่าและเปิดบริการของ Squid และสนอร์ตอินไลน์แล้ว เครื่องไฟร์เบรกจะอยู่ในสถานะ Listening รอรับการเชื่อมต่อจากไคลเอ็นต์ที่ต้องการใช้บริการเว็บเซิร์ฟเวอร์ที่อยู่หลังไฟร์เบรก โดยไฟร์เบรกจะรอรับแพ็คเก็ตที่มาจากไคลเอ็นต์แล้วทำการตรวจสอบแพ็คเก็ตนั้นกับกฎที่มีอยู่ว่าเข้าข่ายพฤติกรรมเป็นผู้บุกรุกหรือไม่ ถ้าแพ็คเก็ตนั้นตรงตามกฎของสนอร์ตอินไลน์แสดงว่าเข้าข่ายเป็นผู้บุกรุก ไฟร์เบรกจะทำการดร็อปแพ็คเก็ตนั้นทิ้งและทำการส่งล็อกไฟล์ที่เกี่ยวกับการบุกรุกนั้น ๆ ไปยังโปรแกรมควบคุม Tartarus Management แต่ถ้าแพ็คเก็ตนั้นเป็นแพ็คเก็ตปรกติ ไฟร์เบรกจะอนุญาตให้ไคลเอนต์สามารถทำการติดต่อกับเว็บเซิร์ฟเวอร์เพื่อรับบริการจากเว็บเซิร์ฟเวอร์ได้ตามปรกติ

บทที่ 9
การพัฒนาระบบล็อกมอนิเตอร์

9.1 แนวคิด

ระบบล็อกมอนิเตอร์ในโครงงานนี้ประกอบด้วยล็อกไฟล์จากส่วนต่าง ๆ ในระบบ ได้แก่ โดยข้อมูลส่วนใหญ่จะเป็นข้อมูลเกี่ยวกับการบุกรุก , ข้อมูลเกี่ยวกับการกระทำของผู้บุกรุกต่อระบบในส่วนต่าง ๆ , ข้อมูลเกี่ยวกับการกระทำของผู้บุกรุกในส่วนไฟร์แทรป โดยข้อมูลทั้งหมดจะถูกเก็บลงในดาต้าเบสและนำออกมาแสดงผลให้ผู้ดูแลระบบสามารถตรวจสอบการบุกรุกที่เกิดขึ้นกับระบบได้

9.2 ขอบเขตและความสามารถ

ระบบล็อกมอนิเตอร์ในโครงงานนี้แบ่งออกเป็น 4 ส่วนย่อย อันได้แก่ ล็อกจากไฟร์สกรีน , ล็อกจากไฟร์อลาร์ม , ล็อกจากไฟร์เบรก และ ล็อกจากไฟร์แทรป โดยแต่ละส่วนพัฒนาด้วยเทคโนโลยีที่แตกต่างกันตามความเหมาะสม ดังนี้

9.2.1 ล็อกมอนิเตอร์ของไฟร์กรีน

โปรแกรมล็อกมอนิเตอร์ของไฟร์สกรีนนั้นแบ่งได้เป็น 2 ส่วนคือ ส่วนล็อกไฟล์ที่เกิดจากการจำแนกแพ็คเก็ตที่เข้ามาในระบบตามกฎของไฟร์วอล และส่วนล็อกไฟล์ที่เกิดจากระบบตรวจจับผู้บุกรุก (Snort_inline) โดยในแต่ละส่วนจะมีขอบเขตความสามารถแตกต่างกันดังนี้

- ล็อกไฟล์จากการจำแนกแพ็คเก็ตตามกฎของไฟร์วอล – เป็นโปรแกรมล็อกมอนิเตอร์ที่ถูกติดตั้งอยู่ในเครื่องแม่ข่ายที่เป็นโดเมนคอนโทรลเลอร์ (FireStation) และเก็บล็อกลงในแอ็คทีฟไดเร็คทอรี จึงพัฒนาโปรแกรมล็อกมอนิเตอร์ของไฟร์สกรีนในส่วนนี้ด้วยภาษาซีที่มีการใช้คลาส ADCONNECT ในการพัฒนาโปรแกรม

- ล็อกไฟล์จากระบบตรวจจับผู้บุกรุก - เป็นโปรแกรมล็อกมอนิเตอร์ที่เก็บล็อกของการจำแนกผู้บุกรุกออกจากผู้ใช้งานจริงโดยอาศัยโปรแกรมสนอร์ดอินไลน์ โดยจะเป็นข้อมูลเกี่ยวกับรูปแบบการบุกรุกที่ตรวจจับได้ วัน เวลา และไอพีแอดเดรสของผู้บุกรุก ซึ่งล็อกไฟล์ในส่วนนี้จะถูกเก็บลงฐานข้อมูลที่อยู่ในเครื่องดาต้าเบสเซิร์ฟเวอร์ของระบบ ซึ่งข้อมูลต่าง ๆ ในเครื่อง LogServer จะถูกส่งออกมาแสดงทางโปรแกรม Tartarus Management

9.2.2 โปรแกรมล็อกมอนิเตอร์ของไฟร์อลาร์ม

ใช้การเขียนโปรแกรมผ่านทางซ็อกเก็ตเพื่อรับข้อมูลจากไฟร์อลาร์มและจัดเก็บลงดาต้าเบส จึงพัฒนาโปรแกรมล็อกมอนิเตอร์ด้วยภาษา C ที่มีคลาส CAsyncSocket ในการติดต่อ

9.2.3 โปรแกรมล็อกมอนิเตอร์ของไฟร์แทรป

เป็นล็อกไฟล์เกี่ยวกับการกระทำของผู้บุกรุกที่กระทำต่อเครื่องกับดักในระบบไฟร์แทรป โดยในการบันทึกพฤติกรรมของผู้บุกรุกจะใช้โปรแกรมเซเบคไคลเอ็นต์ที่อยู่ในเครื่องกับดักเป็นตัวดักจับพฤติกรรมต่าง ๆ แล้วส่งข้อมูลต่าง ๆ ไปให้โปรแกรมเซเบคเซิร์ฟเวอร์ที่อยู่ในเครื่องดาต้าเบสเซิร์ฟเวอร์ของระบบ แล้วนำออกมาแสดงผลให้ผู้ดูแลระบบตรวจสอบผ่านทางโปรแกรม Tartarus Management โดยข้อมูลต่าง ๆที่ได้จากไฟร์แทรป เช่น

- ข้อมูลเบื้องต้นของผู้บุกรุก เช่น หมายเลขไอพีแอดเดรส
- ข้อมูลการป้อนคำสั่ง หรือ keystroke ของผู้บุกรุก
- ข้อมูลที่นำเข้าหรือออก อาจเป็นไฟล์หรือเครื่องมือที่ผู้บุกรุกจะนำมาใช้
- ข้อมูลที่ส่งออกจากเครื่องกับดัก อาจเป็นข้อมูลหรือเป็นการกระทำเพื่อโจมตีระบบ

9.2.4 โปรแกรมล็อกมอนิเตอร์ของไฟร์เบรก

เป็นล็อกไฟล์เกี่ยวกับการบุกรุกของผู้บุกรุกที่เข้ามาในระบบไฟร์เบรก โดยหลังจากที่โปรแกรมสนอร์ตอินไลน์ในไฟร์เบรกตรวจพบรูปแบบการบุกรุกที่เกิดขึ้นและทำการตัดการเชื่อมต่อนั้น ๆ โปรแกรมสนอร์ตอินไลน์ในไฟร์เบรกจะทำการส่งล็อกไฟล์เหตุการณ์ที่เกิดขึ้นไปแสดงผลที่โปรแกรม Tatarus Management

9.3 เทคโนโลยีที่ใช้และการพัฒนาโปรแกรม

9.3.1 ล็อกมอนิเตอร์ของไฟร์สกรีน

- <u>ล็อกไฟล์จากการจำแนกแพ็คเก็ตตามกฎของไฟร์วอล</u>

ล็อกไฟล์ในส่วนนี้จะทำการพัฒนาโดยภาษา C โดยการใช้คลาส ADCONNECT ที่ใช้ในการติดต่อกับแอ็คทีฟไดเร็คทอรี เพื่อเข้าถึงค่าของล็อกไฟล์ที่ถูกส่งมาเก็บยังแอ็คทีฟไดเร็คทอรี จากนั้นจะนำค่าที่ได้นี้มาลงไฟล์ดาต้าเบสที่สร้างจาก Microsoft Access ที่มีชื่อว่า LogfireScreen โดยมีโครงสร้างตารางที่ใช้ในการเก็บล็อกของเครื่องลูกข่ายดังตารางที่ 9.1 และแสดงผลผ่านโปรแกรม

ชื่อเขตข้อมูล	ชนิดข้อมูล	คำอธิบาย
LogNo (คีย์หลัก)	NUMBER	หมายเลขลำดับที่ของล็อก
Date	TEXT	วันที่
Time	TEXT	เวลา
In	TEXT	อินเตอร์เฟสขาเข้า
Out	TEXT	อินเตอร์เฟสขาออก
Mac	TEXT	หมายเลขแม็คแอดเดรส
Source	TEXT	หมายเลขไอพีต้นทาง
Destination	TEXT	หมายเลขไอพีปลายทาง
Protocol	TEXT	โปรโตคอล
Source Port	TEXT	หมายเลขพอร์ตต้นทาง
Destination Port	TEXT	หมายเลขพอร์ตปลายทาง
Type	TEXT	รูปแบบของไอซีเอ็มพี

ตารางที่ 9.1 แสดงโครงสร้างตารางที่ใช้ในการเก็บล็อกของเครื่องเกต์เวย์

- ล็อกไฟล์จากระบบตรวจจับผู้บุกรุก

ล็อกไฟล์ในส่วนนี้เป็นล็อกที่ได้จากโปรแกรมตรวจจับสนอร์ตอินไลน์ที่อยู่ในไฟร์สกรีน โดยข้อมูลจะถูกส่งมาเก็บไว้ในเครื่องเซิร์ฟเวอร์ฐานข้อมูล (LogServer) ซึ่งใช้ในการจัดเก็บข้อมูลลงฐานข้อมูล MySQL เพื่อความเป็นระบบ และสามารถเข้าใช้งานได้อย่างเป็นระบบ เพื่อเชื่อมต่อข้อมูลให้โปรแกรมควบคุม Tartarus Management สามารถมาดึงไปใช้งานได้ และรองรับการเชื่อมต่อข้อมูลในการพัฒนาในอนาคต

9.3.2 ล็อกมอนิเตอร์ของไฟร์อลาร์ม

โปรแกรมนี้จะทำการพัฒนาโดยภาษา C โดยการใช้คลาส CAsyncSocket ที่ใช้ในการสร้าง Socket และทำเป็นเครื่องแม่ข่าย (Server) รอรับการติดต่อมาจากเครื่องลูกข่าย (Client) เมื่อเกิดเหตุการณ์โจมตีขึ้น จากนั้นจะนำค่าที่ได้นี้มาลงไฟล์ดาต้าเบสที่สร้างจาก Microsoft Access ที่มีชื่อว่า LogfireAlarm โดยมีโครงสร้างตารางที่ใช้ในการเก็บล็อกของเครื่องลูกข่ายดังตารางที่ 9.2 และแสดงผลผ่านทางโปรแกรม

ชื่อเขตข้อมูล	ชนิดข้อมูล	คำอธิบาย
LogNo (คีย์หลัก)	NUMBER	หมายเลขลำดับที่ของล็อก
AttackDst	TEXT	หมายเลขไอพีปลายทาง
AttackSrc	TEXT	หมายเลขไอพีต้นทาง
AttackType	TEXT	ชนิดของการโจมตี
AttackDate	TEXT	วันที่
AttackTime	TEXT	เวลา
User	TEXT	ชื่อผู้ใช้
Group	TEXT	กลุ่มผู้ใช้

ตารางที่ 9.2 แสดงโครงสร้างตารางที่ใช้ในการเก็บล็อกของเครื่องลูกข่าย

9.3.3 ล็อกมอนิเตอร์ของไฟร์แทรป

เป็นล็อกไฟล์ที่แสดงข้อมูลเกี่ยวกับการกระทำของผู้บุกรุกที่กระทำเมื่ออยู่ในเครื่องกับดักไฟร์แทรป เช่น Keystroke ของผู้บุกรุก ไอพีแอดเดรสของผู้บุกรุก การนำไฟล์เข้าออกจากระบบของผู้บุกรุก รวมถึงการเปลี่ยนแปลงของไฟล์ภายในเครื่องกับดัก ซึ่งข้อมูลต่าง ๆ จะถูกส่งมาจากโปรแกรมเซเบคไคลเอ็นต์ และโปรแกรมเซมเฮนที่ทำงานอยู่ในเครื่องกับดัก มาเก็บไว้ในเครื่องเซิร์ฟเวอร์ฐานข้อมูล (LogServer) โดยในเครื่องฐานข้อมูลนี้จะมีโปรแกรมเซเบคเซิร์ฟเวอร์และเซมเฮน (Yule) เป็นตัวรับข้อมูลมาเก็บในฐานข้อมูล Mysql อีกทีหนึ่ง

Sebek_server

เซเบคเซิร์ฟเวอร์ เป็นส่วนที่ทำหน้าที่รับค่าข้อมูลที่เซเบคไคล์เอนต์ดักจับได้จากเครื่องกับดัก และระบบการทำงานจะส่งค่ามาเก็บยังฐานข้อมูลที่ได้จัดเตรียมไว้ ซึ่งจะส่งผ่านมาด้วยโพรโตคอลยูดีพี พร้อมทั้งเข้ารหัสลับไว้ โดยไม่จำเป็นจะต้องระบุไอพีปลายทาง เพื่อความปลอดภัยจะระบุเป็น MAC Address ของเครื่องเซิร์ฟเวอร์ ซึ่งเครื่องไคลเอ็นต์ไม่จำเป็นต้องมีการARP Request เพื่อหาปลายทางซึ่งจะช่วยลดความผิดสังเกตจากผู้บุกรุกได้ และการทำงานของเซเบคไคลเอนต์ จะทำการ by-pass ผ่าน TCP stack คือมันจะสร้างแพ็คเก็ตเองเลย ทำให้โปรแกรมจำพวก sniffer ที่ผู้บุกรุกอาจนำเข้ามาติดตั้งในเครื่องกับดัก ไม่สามารถตรวจพบ

Samhain (Yule)

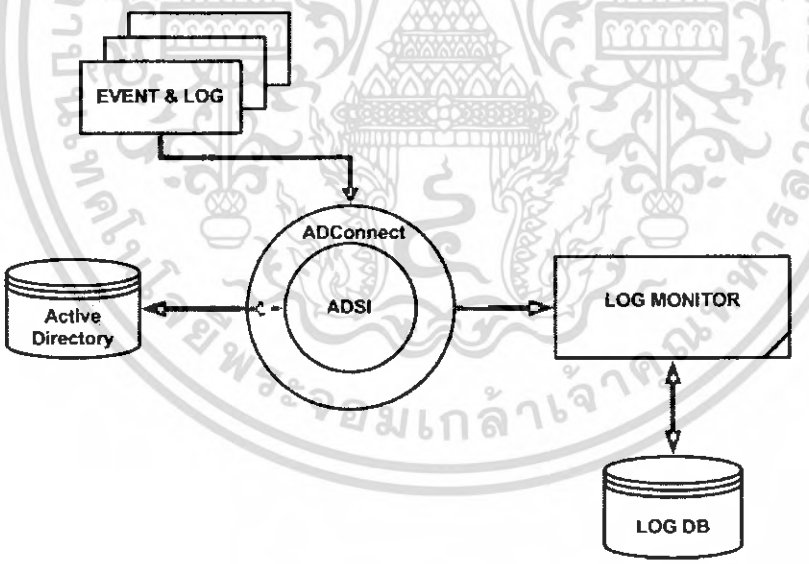
เซมเฮนในส่วนนี้จะเป็นเซอร์วิสที่ทำหน้าที่ในการรับข้อมูลจากเซมเฮนที่อยู่บนเครื่องกับดักมาบันทึกเก็บไว้ในฐานข้อมูลที่ได้จัดเตรียมไว้

9.3.4 ล็อกมอนิเตอร์ของไฟร์เบรก

เป็นล็อกไฟล์เกี่ยวกับการบุกรุกของผู้บุกรุกที่เข้ามาในระบบไฟร์เบรก โดยหลังจากที่โปรแกรมสนอร์ตอินไลน์ในไฟร์เบรกตรวจพบรูปแบบการบุกรุกที่เกิดขึ้นและทำการปิดการเชื่อมต่อนั้น ๆ โปรแกรมสนอร์ตอินไลน์ ในไฟร์เบรกจะบันทึกล็อกไฟล์ลงใน /var/log/snort_inline/snort_inline-fast และ /var/log/snort_inline/snort_inline-full จากนั้นไฟร์เบรกการส่งล็อกไฟล์ไปแสดงผลที่โปรแกรม Tatarus Management

9.4 การทำงานของโปรแกรม

ในการทำงานของระบบล็อกมอนิเตอร์ที่ทำงานร่วมกับความสามารถของบริการแอ็คทีฟไดเร็คทอรี (Active directory) ของ Windows server 2003 จะมีการทำงานโดยที่ข้อมูลจะเก็บอยู่ในฐานข้อมูลของแอ็คทีฟไดเร็คทอรี (Active Directory Database) และเข้าถึงข้อมูลผ่าน ADSI ไปยังทีฟไดเร็กทอรีเซอร์วิส (Active Directory Services) เพื่อเข้าถึงข้อมูลที่จัดเก็บ ซึ่งการทำงานของล็อกมอนิเตอร์ที่ทำงานร่วมกันกับแอ็คทีฟไดเร็คทอรี (Active Directory) แสดงโครงสร้างการทำงาน ดังรูปที่ 8.1



รูปที่ 9.1 แสดงโครงสร้างการทำงานระหว่างล็อกมอนิเตอร์กับแอ็คทีฟไดเร็คทอรี

ส่วนโปรแกรมล็อกมอนิเตอร์ของไฟร์เบรกนั้นจะมีการทำงานโดยการติดต่อกับดาต้าเบส MySql โดยตรงด้วยการส่งคำสั่ง SQL ผ่านทางโปรแกรมเพื่อทำการ Select ROW ที่ต้องการมาแสดงผลคือ Row ที่อยู่ในตาราง Event, Signature, Sensor โดยคำสั่ง SQL ที่ส่งเข้าไปนั้นจะเป็นการ Select ดังต่อไปนี้

1. ค่า NO. คือค่าของ Cid ในตาราง Event
2. ค่า Source IP คือค่าของ Hostname ในตาราง Sensor
3. ค่า Acctack Type คือค่าของ Sig_name ในตาราง Signature
4. ค่า Acctack Time คือค่าของ TimeStamp ในตาราง Event

บรรณานุกรม

เอกสารอ้างอิงที่เป็นหนังสือ

[1] เรืองไกร รังสิพล 2544. เจาะระบบ TCP/IP จุดอ่อนโปรโตคอลและวิธีป้องกัน พิมพ์ครั้งที่ 1 กรุงเทพฯ : บริษัท โปรวิชั่น จำกัด

[2] เรืองไกร รังสิพล 2545. เปิดโลก Firewall และ Internet Security พิมพ์ครั้งที่ 1 กรุงเทพฯ : บริษัท โปรวิชั่น จำกัด

[3] สุรวัฒน์ ปุณณชัยยะ 2543. เปิดโลกของ TCP/IP และโปรโตคอลของอินเตอร์เน็ต พิมพ์ครั้งที่ 1 กรุงเทพฯ : บริษัท โปรวิชั่น จำกัด

[4] สุรศักดิ์ สงวนพงษ์ 2545. สถาปัตยกรรมและโปรโตคอลทีซีพี/ไอพี พิมพ์ครั้งที่ 2 กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชั่น จำกัด

[5] อ. บัณฑิต จามรภูติ 2548. คู่มือ Windows 2003 Server ภาคปฏิบัติ เล่ม 1. พิมพ์ครั้งที่ 2 กรุงเทพฯ : บริษัท เอช.เอ็น กรุ๊ป จำกัด

[6] Anthonh Jones. 2002. Network Programming for Microsoft Windows Second Edition. Canada : Microsoft Press.

[7] Charlie Kaufman. 2002. Network Security Private Communication in a PUBLIC World. New Jersey : Prentice Hall.

[8] Joel Scambray, Sturat McClure. 2001. Hacking Exposed: Network Security Secrets & Solution Second Edition. New York : McGraw-Hill Companies.

[9] Douglas E. Comer. 2000. Internetworking With TCP/IP Principles, Protocols, and Architecture Forth Edition. New Jersey : Prentice Hall.

[10] หนังสือ Honeypots Tracking Hackers ผู้แต่ง Lance Spitzner แปลโดย Marcus J. Ranum ISBN: 0-321-10895-7

ตัวอย่างเอกสารอ้างอิงที่เป็นวิทยานิพนธ์

[1] ระบบตรวจจับผู้บุกรุกทางคอมพิวเตอร์ (Intrusion Detection System)

[2] ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก (Honeypot Program Suite)

[3] ชุดโปรแกรมเน็ตเวิร์คไฟร์วอล (Network Firewall Program Suite)

เอกสารอ้างอิงที่เป็น Web-Site

[1] http://www.codeproject.com/system/windowsservices.asp

[2] http://www.codeproject.com/system/serviceskeleton.asp

บรรณานุกรม (ต่อ)

[3] http://www.codeproject.com/system/xservice.asp

[4] http://www.microsoft.com/windowsserver2003/technologies/pki/default.mspx

[5] http://www.microsoft.com/thailand/windowsserver2003/prodinfo/ActiveDirectory.asp

[6] http://msdn.microsoft.com/msdnmag/issues/01/12/NETServ/

[7] http://www.codeguru.com/

[8] http://www.honeynet.org

[9] http://www.snort.org

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

[11] http://www.securityfocus.com

[12] http://www.bleedingsnort.com