ระบบจัดการการปรับตั้งค่าและกฎ
บนโปรแกรม Snort ผ่านเว็บบราวเซอร์
Snort Configuration Management System via Web Browser

โดย

จตุพล นิลพรัตน์

รหัสประจำตัว 46066228

อาจารย์ที่ปรึกษา

ผศ. ดร. จันทร์บูรณ์ สถิตวิริยวงศ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคฤดูร้อน ปีการศึกษา 2548
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ระบบจัดการการปรับตั้งค่าและกฎบนโปรแกรม Snort ผ่านเว็บบราวเซอร์
นักศึกษา	นายจตุพล นิลพรัตน์
อาจารย์ที่ปรึกษา	ผศ.ดร. จันทร์บูรณ์ สถิตวิริยวงศ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2548

บทคัดย่อ

ระบบตรวจจับการบุกเป็นองค์ประกอบสำคัญอย่างหนึ่ง ในการรักษาความปลอดภัยให้กับระบบเครือข่ายคอมพิวเตอร์ขององค์กร ผลิตภัณฑ์เกี่ยวกับระบบตรวจจับการบุกรุกมีทั้งแบบที่เป็นฮาร์ดแวร์และซอฟต์แวร์ แต่เนื่องจากว่าระบบตรวจจับการบุกรุกแบบฮาร์ดแวร์นั้นมีราคาสูง ทำให้องค์กรขนาดกลางและขนาดเล็ก เลือกใช้ระบบตรวจจับการบุกรุกแบบซอฟต์แวร์แทน ซอฟต์แวร์ที่ได้รับความนิยมและนำมาใช้งานกันอย่างกว้างขวางมีชื่อว่า Snort สาเหตุที่ Snort ถูกนำมาใช้งานกันอย่างกว้างขวางก็เพราะว่า เป็นโปรแกรมแบบเปิดเผยโค้ด ใช้งานได้โดยไม่เสียค่าลิขสิทธิ์ ต้องการทรัพยากรในการทำงานเพียงเล็กน้อย ตรวจจับการบุกรุกได้ครอบคลุม แต่ Snort ก็ยังมีข้อเสียอยู่คือการทำงานต้องทำผ่านคอมมานด์ไลน์ อีกทั้งค่าคอนฟิกูเรชันต่างๆยังอยู่ในรูปแบบของเท็กซ์ไฟล์ทำให้การใช้งานและการจัดการทำได้ไม่สะดวกนัก ดังนั้นโครงการพัฒนาระบบนี้จึงสร้างเครื่องมือสำหรับจัดการกับ คอนฟิกูเรชันไฟล์ กฎต่างๆของ Snort Sensors ด้วย Graphic User Interface (GUI) ผ่านทางเว็บบราวเซอร์

Title	Snort Configuration Management System via Web Browser
Student	Mr. Chatupon Nilparat
Advisor	Asst. Prof. Chanboon Sathitwiriyawong, Ph.D.
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2005

ABSTRACT

An intrusion detection system is an important factor to keep security for organization's network. The products about a verify intrusion detection system are hardware and software but hardware are expensive. That make medium size and small organization cannot effort and choose software in order to detect intrusion. The popular software that use in widespread is Snort because it is reveal source code. It is freeware and use less resource to work but Snort still has disadvantage. It must work through command line and file configuration is text files that cause difficult to use and inconvenient for management. So this project development create tool for manipulating with file configuration, rules of Snort Sensors with Graphic User Interface (GUI) pass by web browser.

กิตติกรรมประกาศ

โครงการพัฒนาระบบนี้จะสำเร็จไปไม่ได้เลยถ้าไม่ได้รับความช่วยเหลือและการสนับสนุนจากฝ่ายต่างๆดังต่อไปนี้

1. ผศ.ดร. จันทร์บูรณ์ สถิตวิริยวงศ์ อาจารย์ที่ปรึกษาโครงการและสัมมนา ผู้ซึ่งกรุณาให้คำปรึกษาในข้อปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างพัฒนาระบบงาน
2. คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เอื้อเฟื้อหนังสือในการค้นคว้า เพื่อพัฒนาระบบงาน
3. คุณอดิศัย ศิลารมย์ (ตูนชาย) ที่กรุณาให้คำปรึกษาและคำแนะนำในการพัฒนาระบบงาน
4. เพื่อน ๆ ทุกท่านที่คอยให้ความช่วยเหลือ และเป็นกำลังใจมาโดยตลอด

จึงใคร่ขอขอบคุณบุคคลดังกล่าวข้างต้นมา ณ โอกาสนี้

จตุพล นิลพรัตน์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงการพัฒนาระบบงาน

ระบบเครือข่ายคอมพิวเตอร์เป็นระบบที่ออกแบบมาโดยมีจุดประสงค์คือ ทำให้เครื่องคอมพิวเตอร์สามารถติดต่อสื่อสารกันได้ จากจุดประสงค์ดังกล่าวทำให้ละเลยประเด็นด้านความปลอดภัย ดังนั้นการจะใช้งานระบบเครือข่ายคอมพิวเตอร์ให้มีความปลอดภัยจะต้องติดตั้งระบบรักษาความปลอดภัยเพิ่มเติม ไฟร์วอลล์ (Firewall) เป็นอุปกรณ์รักษาความปลอดภัยที่พัฒนาขึ้นเพื่อควบคุมการเข้าถึงระบบเครือข่ายคอมพิวเตอร์ เปรียบเสมือนทหารที่คอยป้องกันไม่ให้บุคคลจากภายนอกบุกรุกเข้ามาภายในประเทศ แต่ไฟร์วอลล์ไม่สามารถตรวจจับพฤติกรรมของบุคคลหรือระบบที่อยู่ภายในเครือข่ายได้ การจะเติมเต็มในส่วนนี้ได้นั้นจะต้องติดตั้งระบบตรวจจับการบุกรุกลงไป ระบบตรวจจับการบุกรุกในปัจจุบันมีอยู่มากมายหลายชนิด ถ้าแบ่งตามเทคโนโลยีก็จะมีระบบตรวจจับที่เป็นฮาร์ดแวร์กับซอฟต์แวร์ กรณีที่แบ่งตามวิธีการทำงานก็จะมีระบบตรวจจับการบุกรุกแบบใช้สัญลักษณ์หรือกฎในการตรวจจับ กับแบบที่ใช้ค่าสถิติของการใช้งานทรัพยากรเป็นตัวตรวจจับ หรือถ้าแบ่งตามชนิดของข้อมูลที่นำมาตรวจสอบก็จะมี ระบบตรวจจับการบุกรุกบนเครือข่าย กับระบบตรวจจับการบุกรุกเฉพาะเครื่อง

Snort เป็นระบบตรวจจับการบุกรุกแบบซอฟต์แวร์ที่ได้รับความนิยมสูง สามารถดาวน์โหลดมาใช้งานได้ฟรี กฎมาตรฐานที่มีมานั้นสามารถตรวจจับการบุกรุกได้หลายรูปแบบ อีกทั้งยังอนุญาตให้ผู้ใช้ปรับปรุง เพิ่มเติมกฎได้ด้วย แต่เนื่องจากการใช้งานโปรแกรม Snort จะต้องกระทำผ่านคอมมานไลน์ อีกทั้งค่าคอนฟิกูเรชันต่างๆยังอยู่ในรูปแบบของเท็กซ์ไฟล์ทำให้การใช้งานและการจัดการทำได้ไม่สะดวกนัก ดังนั้นโครงการพัฒนาระบบนี้จึงสร้างเครื่องมือสำหรับจัดการกับคอนฟิกูเรชันไฟล์ กฎต่างๆของ Snort Sensors ด้วย Graphic User Interface (GUI) ผ่านทางเว็บบราวเซอร์

1.2 วัตถุประสงค์ของโครงงานพัฒนาระบบงาน

- เพื่อศึกษาการทำงานของโปรแกรม Snort
- เพื่อช่วยให้การจัดการกับกฎ และคอนฟิกูเรชันไฟล์ของ Snort Sensor เป็นไปอย่างสะดวกมากขึ้น
- เพื่อป้องกันความผิดพลาดที่จะเกิดกับ Snort Sensor อันเนื่องมาจากการสั่งงานที่ผิดพลาดของผู้ดูแลระบบ (ที่ไม่คุ้ยเคยกับการใช้งานโปรแกรม Snort)

1.3 ขอบเขตของการศึกษาและพัฒนาระบบงาน

- พัฒนาภายใต้สภาพแวดล้อมของระบบปฏิบัติการ Linux Redhat 9 ด้วย PHP
- สามารถสั่งการให้ Snort Sensor ทำงานหรือหยุดทำงานผ่านทางเว็บบราวเซอร์
- เพิ่ม แก้ไข ลบค่าคอนฟิกูเรชันและกฎต่างๆบน Snort Sensor ได้
- นำ Analysis Console for Intrusion Database (ACID) มาช่วยในการวิเคราะห์ข้อมูล
- ผู้ใช้งานระบบที่พัฒนาขึ้นนี้จะทำงานต่างๆผ่านทางเว็บบราวเซอร์

1.4 ขั้นตอนในการพัฒนาโครงการ

- ศึกษาการทำงานของโปรแกรม Snort
- ศึกษาการใช้เครื่องมือต่างๆที่ใช้ร่วมกับโปรแกรม Snort เมื่อทำงานในโหมด NIDS
- วิเคราะห์ออกแบบระบบ
- ศึกษาเครื่องมือที่ใช้ในการพัฒนาระบบงาน
- พัฒนาโปรแกรม
- ทดสอบการทำงานของโปรแกรม
- สรุปผลการทดสอบ
- จัดทำเอกสารประกอบการพัฒนาระบบ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- มีความรู้ความเข้าใจในการใช้โปรแกรม Snort มากยิ่งขึ้น
- ผู้ดูแลระบบเครือข่ายคอมพิวเตอร์สามารถจัดการการทำงานของ Snort ไม่ว่าจะเป็นการปิดเปิด Snort Sensor การแก้ไขคอนฟิกูเรชันไฟล์ การเพิ่ม ปรับปรุงหรือลบกฎผ่านเว็บบราวเซอร์
- สร้างความรู้พื้นฐานพร้อมทั้งชี้ให้เห็นถึงองค์ประกอบที่จำเป็นต่อการจัดการ Snort Sensor ให้แก่ผู้ดูแลระบบที่ไม่เคยใช้โปรแกรม Snort มาก่อน
- เป็นแนวทางในการพัฒนาโปรแกรมประยุกต์อื่นๆ ที่เกี่ยวข้องกับโปรแกรม Snort ต่อไป

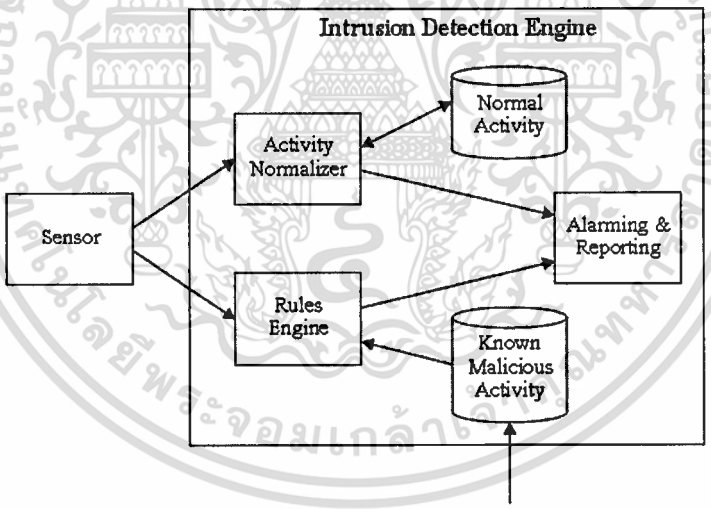
บทที่ 2
ระบบตรวจจับการบุกรุก

2.1 ความหมายของระบบตรวจจับการบุกรุก

ระบบตรวจจับการบุกรุก (Intrusion Detection System) คือระบบที่ทำหน้าที่ติดตามดูการทำงานที่เกิดขึ้นบนระบบคอมพิวเตอร์ เพื่อหาสัญญาณบ่งชี้ว่ามีการบุกรุกระบบคอมพิวเตอร์ อาทิ เช่นการพยายามเจาะเข้าระบบ หรือ การเข้าใช้ระบบเกินขอบเขตที่ได้รับอนุญาต

2.2 องค์ประกอบของระบบตรวจจับการบุกรุก

องค์ประกอบทั่วๆไปของระบบตรวจจับการบุกรุกจะเป็นดังรูปที่ 2.1



รูปที่ 2.1 องค์ประกอบของระบบตรวจจับการบุกรุก

- Sensors เป็นส่วนที่ทำหน้าที่รวบรวมข้อมูลต่างๆเช่น แพ็กเก็ตที่ไหลอยู่บนเครือข่าย, ไฟล์บันทึกกิจกรรมที่เกิดขึ้นบนระบบ (System log files) โดยข้อมูลเหล่านี้จะถูกส่งต่อไปให้ Analyzer
- Normal Activity เป็นฐานข้อมูลที่เก็บสถิติการใช้งานระบบจากกิจกรรมต่างๆในสภาวะปกติ
- Known Malicious Activity เป็นฐานข้อมูลที่เก็บรูปแบบของการบุกรุกหรือโจมตีระบบ
- Rules Engine เป็นส่วนที่วิเคราะห์ว่ามีการบุกรุกเกิดขึ้นในระบบหรือไม่ โดยเปรียบเทียบกิจกรรมที่เกิดขึ้นกับกับ Signature ที่เก็บไว้ Know Malicious Activity

- Activity Normalizer เป็นส่วนที่วิเคราะห์ว่าเกิดกิจกรรมที่ผิดปกติขึ้นในระบบหรือไม่ โดยพิจารณาจากค่า Base-Line ของข้อมูลใน Normal Activity กับ สถิติของกิจกรรมที่เกิดขึ้นในปัจจุบัน ตัวอย่างของค่าที่ใช้เก็บเป็นสถิติได้แก่ การใช้งาน CPU การใช้งานสื่อบันทึกข้อมูล จำนวนแพ็กเก็ตที่วิ่งเข้าออกในระบบ
- Alarming and Reporting เป็นส่วนที่ใช้แจ้งเตือนและรายงานให้ผู้ดูแลระบบทราบว่ามีการบุกรุกระบบเกิดขึ้น

2.3 วิธีการตรวจจับการบุกรุก

2.3.1 วิธีเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก (Misuse Intrusion Detection)

วิธีนี้จะอาศัยรูปแบบของการบุกรุกที่เคยเกิดขึ้นแล้วเป็นตัวตรวจสอบ โดยจะรวบรวมรูปแบบของการบุกรุกแล้วเก็บเป็นกฎไว้ เมื่อมีการใช้งานระบบคอมพิวเตอร์ ข้อมูลที่เป็นพฤติกรรมการใช้งานจะถูกนำมาเปรียบเทียบกับกฎ หากตรงกับกฎระบบตรวจจับการบุกรุกจะแจ้งเตือน วิธีการนี้ผู้ดูแลระบบสามารถเปลี่ยนแปลงหรือเพิ่มกฎได้ แต่สิ่งที่จะละเลยไม่ได้คือต้องมีการปรับปรุงกฎให้ทันสมัยอยู่เสมอ

2.3.2 วิธีตรวจสอบการใช้งานทรัพยากรระบบที่ผิดปกติ (Anomaly Intrusion Detection)

วิธีนี้จะอยู่บนสมมติฐานที่ว่ากิจกรรมใดๆที่เป็นการบุกรุกจะมีการใช้งานทรัพยากรของระบบอย่างผิดปกติ ดังนั้นจะมีการเก็บบันทึกข้อมูลการใช้งานระบบคอมพิวเตอร์เอาไว้สำหรับใช้เปรียบเทียบเพื่อหาว่ากิจกรรมใดเป็นกิจกรรมที่ผิดปกติ โดยการเปรียบเทียบจะใช้หลักการของสถิติเข้าช่วย

2.4 ประเภทของระบบการตรวจจับการบุกรุก

2.4.1 ระบบตรวจจับการบุกรุกเฉพาะเครื่อง (Host-based Intrusion Detection System)

Host-based Intrusion Detection System (HIDS) เป็นซอฟต์แวร์ที่ติดตั้งลงบนเครื่องที่ต้องการความปลอดภัยสูงอย่างเช่นเครื่อง Server ทำหน้าที่วิเคราะห์กิจกรรมต่างๆที่เกิดขึ้นบนเครื่องที่ได้ติดตั้ง HIDS เท่านั้น กิจกรรมที่มุ่งร้ายต่อระบบสามารถตรวจสอบได้จาก Operating System logs และ Application logs บนเครื่อง ดังนั้น HIDS จะต้องนำ Logs เหล่านั้นมาตรวจสอบโดยใช้ทรัพยากรของระบบให้น้อยที่สุด ผลิตภัณฑ์ที่เป็น HIDS ในปัจจุบันไม่มีตัวไหนที่จะนำ Log ทั้งหมดมาวิเคราะห์เพราะจะทำให้สิ้นเปลืองทรัพยากรและเวลา มากจนเกินไป ด้วยเหตุนี้การเลือกใช้งานก็ต้องพิจารณาว่า Log ที่ HIDS นำไปวิเคราะห์นั้นมีโอกาสที่จะเจอการบุกรุกอยู่ในเกณฑ์ที่สามารถยอมรับได้หรือไม่ โดยไม่ใช้ทรัพยากรและเวลามากจนเกินไป

จากที่ได้กล่าวมาข้างต้นจะเห็นว่า Log ต่างๆเป็นปัจจัยสำคัญในการตรวจจับการบุกรุก ถ้ามีการเปลี่ยนแปลง Log เหล่านั้น ก็จะทำให้การตรวจจับผิดพลาดดังนั้น HIDS ที่ดีจะต้องตรวจสอบการเปลี่ยนแปลงของ Log ด้วย ตัวอย่างของ Log ที่ใช้ในการตรวจจับการบุกรุกได้แก่ syslog บนระบบปฏิบัติการ UNIX หรือ system events บนระบบปฏิบัติการ Windows

ข้อดีของ HIDS

- ไม่ต้องจัดเตรียมฮาร์ดแวร์เพิ่มเติม สำหรับการเฝ้าระวังการบุกรุก เพราะติดตั้งลงบนเครื่องคอมพิวเตอร์ที่ใช้งานในระบบอยู่แล้ว
- สามารถตรวจสอบกิจกรรมที่ NIDS ไม่สามารถตรวจสอบได้เช่น ข้อมูลที่มีการเข้ารหัสจากต้นทางและถอดรหัสออกที่เครื่องปลายทางเท่านั้น
- ตรวจสอบการบุกรุกรูปแบบใหม่ๆได้ดีกว่า NIDS
- ตอบสนองต่อการบุกรุกได้อย่างรวดเร็ว
- เกิดความผิดพลาดแบบ False Positive น้อยกว่า NIDS

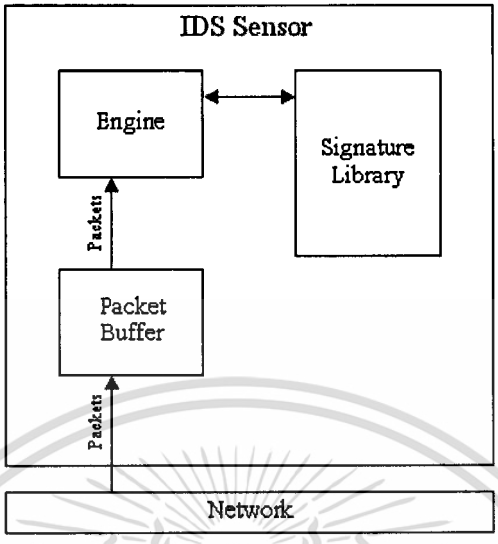
ข้อเสียของ HIDS

- เครื่องที่ติดตั้ง HIDS จะต้องแบ่งปันทรัพยากรมาให้ HIDS แทนที่จะนำไปใช้ในงานบริการหลักอย่างเต็มที่
- รับรู้หรือตรวจสอบการโจมตีได้เพียงบางส่วนของระบบเครือข่ายคอมพิวเตอร์
- อาจถูกขัดขวางการทำงาน จากการโจมตีแบบ Denial of Service (DOS)

2.4.2 ระบบตรวจจับการบุกรุกในเครือข่าย (Network-based Intrusion Detection System)

Network-based Intrusion Detection System (NIDS) มีทั้งแบบที่เป็นฮาร์ดแวร์และซอฟต์แวร์ ทำหน้าที่วิเคราะห์ข้อมูลบนเครือข่ายที่รับผิดชอบว่ามีการบุกรุกเกิดขึ้นหรือไม่ และเนื่องจากต้องนำข้อมูลบนเครือข่ายมาวิเคราะห์ดังนั้น IDS ประเภทนี้จะต้องมีกลไกในการดักจับข้อมูลเหมือนกับ Sniffer ระบบตรวจจับการบุกรุกในเครือข่าย (NIDS) ยังแบ่งย่อยเป็น 2 ประเภท คือ Signature-Based Intrusion Detection และ Analysis-Based Intrusion Detection

2.4.2.1 Signature-Based Intrusion Detection เป็น NIDS ที่นำมาสร้างเป็นผลิตภัณฑ์มากที่สุดในปัจจุบัน มีองค์ประกอบดังรูปที่ 2.2



รูปที่ 2.2 องค์ประกอบของระบบตรวจจับการบุกรุกแบบ Signature-Based NIDS

หลักการทำงานคือจะทำการดักจับข้อมูลบนเครือข่ายที่รับผิดชอบ แล้วนำข้อมูลที่ได้นั้นมาเปรียบเทียบกับ สัญลักษณ์ของการบุกรุก (Signature) ที่เก็บอยู่ในฐานข้อมูล ถ้าข้อมูลที่เข้ามาตรงกับ สัญลักษณ์ของการบุกรุก NIDS ก็จะแจ้งเตือนว่ามีการบุกรุก แต่ถ้าไม่ตรงกันอาจเป็นไปได้ 2 ทางคือ ข้อมูลนั้นเป็นข้อมูลปกติ หรือ ข้อมูลนั้นเป็นการบุกรุกแต่ IDS ไม่สามารถตรวจจับได้เพราะในฐานข้อมูลไม่มีข้อมูลแบบนี้อยู่ เพราะฉะนั้น NIDS ประเภทนี้จะต้องมีการปรับปรุงสัญลักษณ์ของการบุกรุก (Signature) ในฐานข้อมูลให้ทันสมัยอยู่เสมอ

2.4.2.2 Analysis-Based Intrusion Detection เป็น NIDS ที่สร้างมาจากแนวคิดว่าระบบเครือข่ายคอมพิวเตอร์ที่ถูกโจมตีจะมีปริมาณการรับส่งข้อมูลผิดไปจากเดิม ดังนั้นจะต้องมีการเก็บสถิติการใช้งานภายในเครือข่ายคอมพิวเตอร์ไว้ตลอด หลักการทำงานของ NIDS ประเภทนี้คือนำค่าสถิติของการใช้งานเครือข่ายคอมพิวเตอร์ในตอนสภาวะปกติ มาเปรียบเทียบกับค่าสถิติของการใช้งานเครือข่ายคอมพิวเตอร์ในปัจจุบัน ถ้าเปรียบเทียบแล้วมีค่าแตกต่างไป ก็แสดงว่ามีการบุกรุกเกิดขึ้นในระบบ

ข้อดีของ NIDS

- สามารถติดตั้งบน Switching Hub แทนการติดตั้งบนเครื่องคอมพิวเตอร์
- สามารถจัดการระบบตรวจจับการบุกรุกแบบรวมศูนย์ได้ (กรณีที่มีตัวตรวจจับการบุกรุกมากกว่า 1 บนเครือข่าย)
- รับรู้หรือตรวจสอบการโจมตีบนระบบเครือข่ายคอมพิวเตอร์ได้ในวงกว้างกว่า HIDS

ข้อเสียของ NIDS

- ไม่สามารถวิเคราะห์ข้อมูลที่มีการเข้ารหัส
- ความยุ่งยากในการจัดการจะแปรผันตามขนาดของระบบเครือข่ายคอมพิวเตอร์
- ไม่สามารถระบุได้ว่าการบุกรุกนั้นประสบความสำเร็จหรือล้มเหลว

2.5 ความผิดพลาดในระบบตรวจจับการบุกรุก

ความผิดพลาดที่เกิดขึ้นในระบบตรวจจับการบุกรุกมีอยู่ 2 แบบคือ false positive กับ false negative

2.5.1 false positive เป็นความผิดพลาดที่เกิดจากการที่ระบบตรวจจับการบุกรุกแจ้งเตือนว่ามีการบุกรุกแต่แท้ที่จริงแล้วไม่มีการบุกรุกเกิดขึ้น ความผิดพลาดแบบนี้ไม่ก่อให้เกิดอันตรายเพียงแต่ผู้ดูแลระบบต้องมีความรู้เพียงพอ

2.5.2 false negative เป็นความผิดพลาดที่อันตรายเพราะมีการบุกรุกเกิดขึ้นบนระบบคอมพิวเตอร์แต่ระบบตรวจจับการบุกรุกตรวจไม่เจอ

บทที่ 3

โปรแกรม Snort

Snort เป็นระบบตรวจจับการบุกรุกประเภท Signature-Based NIDS ใช้สัญลักษณ์ที่จัดเตรียมไว้ล่วงหน้าสำหรับตรวจจับการบุกรุก พัฒนาด้วยภาษาซี โดย Martin Roesch ตั้งแต่ปีค.ศ. 1998 และมีการพัฒนาเพิ่มเติมจากสมาชิกในกลุ่มสังคมโอเพนซอร์สมาถึงทุกวันนี้

3.1 โหมดการทำงานของ Snort

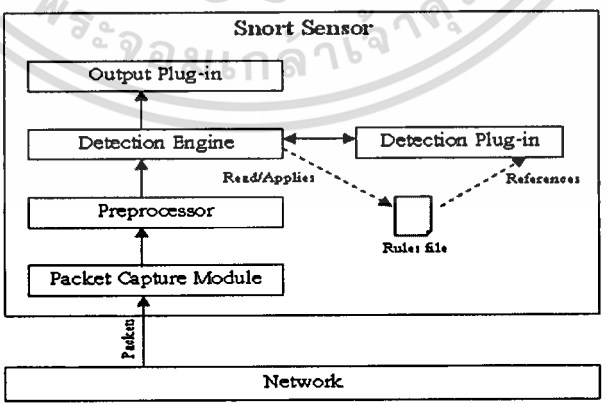
การทำงานของ Snort จะมีอยู่ด้วยกัน 3 โหมดดังนี้

3.1.1 Sniffer Mode เป็นการดักจับข้อมูลที่วิ่งบนเครือข่ายคอมพิวเตอร์แล้วแสดงผลบนจอมอนิเตอร์

3.1.2 Packet Logger Mode เป็นการบันทึกข้อมูลจากการดักจับข้อมูลที่วิ่งบนเครือข่ายคอมพิวเตอร์ลงสู่สื่อบันทึกข้อมูลที่เตรียมไว้

3.1.3 Network Intrusion System Mode เป็นการวิเคราะห์ข้อมูลบนเครือข่ายคอมพิวเตอร์เพื่อตรวจหาว่ามีการบุกรุกหรือไม่ โดยจะอาศัยกฎในการตัดสินใจว่าข้อมูลใดเป็นการโจมตี

3.2 สถาปัตยกรรมของ Snort



รูปที่ 3.1 สถาปัตยกรรมของโปรแกรม Snort

สถาปัตยกรรมของโปรแกรม Snort จะเป็นดังรูปที่ 3.1 ซึ่งประกอบด้วยองค์ประกอบหลัก 4 ส่วนด้วยกันคือ

3.2.1 Packet Capture Module ทำหน้าที่ในการดักจับข้อมูลบนเครือข่ายมาใส่ไว้ในโครงสร้างข้อมูลที่โปรแกรม Snort เข้าใจ จากนั้นก็ทำการตีความว่าเป็นข้อมูลชนิดใดในชั้นดาต้าลิงก์ องค์ประกอบที่สำคัญในส่วนนี้คือ ไลบารี่ libcap และ ฟังก์ชัน ProcessPacket( ) โดยไลบารี่ libcap จะเปลี่ยนโหมดการทำงานของ Network interface card (NIC) ให้เป็น Promiscuous เพื่อรับแพ็กเก็ตทั้งหมดที่ไหลอยู่บนเครือข่าย จากนั้นทำสำเนาข้อมูลที่ไหลอยู่บนเครือข่ายลงบน Network device driver สำหรับฟังก์ชัน ProcessPacket( ) จะทำการแยกข้อมูลตามชนิดของข้อมูลในชั้นดาต้าลิงก์ (ชนิดของข้อมูลที่สามารถแยกได้จะมี Ethernet Wi-fi Token-ring เป็นต้น) จากนั้นจะส่งข้อมูลเหล่านั้นไปให้ Preprocessor

3.2.2 Preprocessor ทำหน้าที่แปลงข้อมูลให้อยู่ในรูปแบบที่ Dectection engine เข้าใจ ตารางที่ 3.1 จะแสดง Preprocessor ทั้งหมดของโปรแกรม snort

ตารางที่ 3.1 Preprocessor บนโปรแกรม Snort

Type of Preprocessor			
Preprocessor Name	Reassembling	Decoding and Normalizing	NonRule/Anomaly Detection
frag2	/		
stream4	/		
stream4_reassemble	/		
http_decode		/	
rpc_decode		/	
telnet_decode		/	
bo			/
portscan			/
portscan_ignorhosts			/
arpspoof			/
arpspoof_detect_host			/
conversation			/
portscan2			/

3.2.3 Detection Engine เป็นส่วนสำคัญที่สุดของโปรแกรม Snort ใช้ตรวจสอบว่าข้อมูลที่เข้ามาในเครือข่ายเป็นการบุกรุกหรือไม่ การทำงานของ Detection Engine คือนำข้อมูลที่ได้รับจาก Preprocessor มาเปรียบเทียบกับกฎ ถ้าตรงกับกฎก็จะทำงานตาม Rule action ที่ได้ตั้งไว้ใน Rule header แต่ถ้าข้อมูลไม่ตรงกับกฎ ก็จะตรวจสอบข้อมูลถัดไป

3.2.4 Output Plug-in ทำหน้าที่แสดงผลของการตรวจสอบการบุกรุกในกรณีที่ข้อมูลที่เข้ามาตรงกับกฎที่ตั้งไว้ใน Detection engine ให้ผู้ดูแลระบบทราบ

3.3 การสร้างกฎสำหรับตรวจจับการบุกรุก (Snort Rules)

กฎสำหรับตรวจจับการบุกรุกประกอบด้วย 2 ส่วนคือ Rule Header และ Rule Body

3.3.1 Rule Header จะเก็บข้อมูลเกี่ยวกับกิจกรรมที่ Snort ต้องกระทำ หรือข้อกำหนดที่ใช้ในการจับคู่กฎกับแพ็กเก็ต ซึ่งจะประกอบไปด้วยฟิลด์ต่างๆดังนี้

3.3.1.1 Rule Action เป็นสิ่งที่ใช้บอกว่า Snort ต้องทำอย่างไรกับแพ็กเก็ต ที่ตรงกับเงื่อนไขของกฎที่ตั้งไว้ โดยมีการตั้งค่ากระทำไว้ 5 รูปแบบคือ

- Pass ปล่อย ไม่สนใจแพ็กเก็ต ใช้กับแพ็กเก็ตที่มาจากเครือข่ายที่เชื่อถือได้
- Alert สร้างการแจ้งเตือนตามวิธีการเตือนที่เลือกไว้ พร้อมทั้งบันทึกข้อมูลจากการดักจับแพ็กเก็ตลงในบริเวณ (Location) ที่ผู้ใช้ที่กำหนดไว้
- Log บันทึกข้อมูลจากการดักจับแพ็กเก็ตลงในบริเวณ (Location) ที่ผู้ใช้ที่กำหนดไว้
- Activate ทำการแจ้งเตือน และสั่งให้ กฎแบบไดนามิกที่เกี่ยวข้องทำงาน
- Dynamic ไม่ทำอะไรจนกว่าจะถูกกระตุ้นจากกฎแบบ activate โดยเมื่อทำงานแล้วจะทำการบันทึก แพ็กเก็ตในบริเวณ (Location) ที่ผู้ใช้ที่กำหนดไว้

3.3.1.2 Protocol คือโพรโตคอลที่ต้องการให้กฎตรวจสอบ ซึ่งในโปรแกรม Snort เวอร์ชัน 2.0 จะสนับสนุนโพรโตคอล IP ICMP TCP และ UDP

3.3.1.3 Source Information คือหมายเลขไอพีแอดเดรสหรือหมายเลขเครือข่าย (IP/Network Address) และหมายเลขพอร์ต (Port Number) ของต้นทาง

3.3.1.4 Destination Information คือหมายเลขไอพีแอดเดรสหรือหมายเลขเครือข่าย (IP/Network Address) และหมายเลขพอร์ต (Port Number) ของปลายทาง

3.3.1.5 Direction Operation คือทิศทางในการเคลื่อนที่ของแพ็กเก็ตที่จะตรวจสอบ มีอยู่ 2 ทิศทาง ด้วยกันดังนี้ -> และ <>

-> หมายถึงพิจารณาเพียงทิศทางเดียว

<> หมายถึงพิจารณาทั้งสองทิศทาง

3.3.2 Rule Body เป็นข้อความที่เขียนต่อจาก Rule Header ของกฎโดยจะอยู่ในวงเล็บ Rule Body นั้นเกิดจากการนำ Rule Option ต่างๆมาประกอบกันโดยจะใช้เครื่องหมาย " ; " เป็นตัวแบ่งแยก **Rule Option** แบ่งออกเป็น 6 กลุ่มด้วยกันดังต่อไปนี้

3.3.2.1 Meta-data Rule Option

- Msg ใช้สำหรับกำหนดข้อความที่ต้องการเก็บไว้ใน log หรือข้อความสำหรับแจ้งเตือน

รูปแบบ **msg:**"< ข้อความที่ต้องการ >";

- SID ใช้สำหรับกำหนดหมายเลขให้กับกฎของ Snort ค่า SID ต้องไม่ซ้ำกันและมีค่าเสมอ ค่าที่เป็นไปได้ของหมายเลขกฎมีดังนี้

< 100 สงวนไว้ใช้ในอนาคต

100 – 1,000,000 ใช้กับกฎที่สร้างโดย Snort.org

> 1,000,000 ใช้กับกฎที่ผู้ใช้สร้างขึ้นเอง (Local Rules)

รูปแบบ **sid:**< หมายเลขกฎ >;

- Rev ใช้สำหรับกำหนดหมายเลขเวอร์ชันให้กับกฎของ Snort

รูปแบบ **rev:**< หมายเลขเวอร์ชันของกฎ >;

- Priority ใช้สำหรับกำหนดความสำคัญให้กับกฎ

รูปแบบ **priority:**< หมายเลขลำดับความสำคัญ >;

- Classtype ใช้สำหรับจัดกลุ่มให้กับกฎ โดยแต่ละกลุ่มนั้นจะแบ่งโดยใช้ประเภทของการโจมตี

รูปแบบ **classtype:**< ชื่อประเภทของการโจมตี >;

- Reference ใช้กำหนดแหล่งที่มาของกฎ ในกรณีที่กฎนี้ได้มาจากระบบอื่น

รูปแบบ **ref:**< id system >, < id >;

3.3.2.2 Payload Rule Option ใช้กำหนดเนื้อหาที่ต้องการให้ Snort ตรวจสอบ

- Content เป็นส่วนที่สำคัญของโปรแกรม Snort ช่วยให้ผู้ใช้งานตรวจสอบข้อมูลภายใน Payload ของแพ็กเก็ตได้ โดยข้อมูลหรือข้อความที่ต้องการตรวจสอบนั้นเป็นได้ทั้งตัวอักษรธรรมดา และแบบไบนารี ถ้าข้อความเป็นแบบไบนารีจะต้องใช้เครื่องหมาย " | " ครอบข้อความไว้ ดังตัวอย่าง ต้องการตรวจสอบข้อความ 5C 00 ก็ให้พิมพ์ดังนี้ content: "|5C 00|";

รูปแบบ **content:** [!] "< ข้อความที่ต้องการให้ตรวจจับ >";

หมายเหตุ ถัดจากนี้ไปจะใช้ประโยคว่า รูปแบบที่ถูกกำหนด (The specified pattern) แทนข้อความที่ต้องการให้ตรวจจับของ Content

- Content list ใช้ตรวจสอบข้อความที่ต้องการ หลายๆข้อความ

รูปแบบ **content-list:** < แฟ้มข้อมูล >;

- Offset เป็นส่วนที่อนุญาตให้ผู้ใช้งานสามารถกำหนดจุดเริ่มต้นในการค้นหารูปแบบที่ถูกกำหนดภายในแพ็กเก็ต โดยจะบอกเป็นจำนวนไบท์ที่ต้องการข้าม ซึ่งจะนับจากไบท์แรก

รูปแบบ **offset:**< จำนวนไบท์ที่ข้าม >;

- Depth ใช้เพื่อกำหนดว่าจะให้ค้นหารูปแบบที่ถูกกำหนด ภายในแพ็กเก็ตเป็นจำนวนเท่าไหร่ (ความลึก)

รูปแบบ **depth:**< จำนวนไบท์ที่ให้ค้นหา >;

- Distance ใช้กำหนดจำนวนไบท์ที่ต้องการข้ามก่อนที่จะเริ่มค้นหารูปแบบที่ถูกกำหนด การนับนั้นจะเริ่มนับ ณ จุดสุดท้ายที่ค้นพบรูปแบบที่ถูกกำหนดตัวก่อนหน้า (Previous pattern match)

รูปแบบ **distance:**< จำนวนไบท์ >;

- Within ใช้กำหนดจำนวนไบท์ภายในแพ็กเก็ตที่ต้องการค้นหารูปแบบที่ถูกกำหนด ออกแบบมาให้ใช้คู่กับ Distance

รูปแบบ **within:**< จำนวนไบท์ >;

- No case ใช้เพื่อกำหนดให้ Snort ค้นหาข้อความในส่วนของ Content โดยไม่ต้องสนใจว่าตัวอักษรนั้นจะ เป็นตัวอักษรพิมพ์เล็กหรือใหญ่

รูปแบบ **nocase;**

- Regex เป็นส่วนที่สามารถเพิ่มความยืดหยุ่นในการกำหนดข้อความในส่วนของ content โดยผู้ใช้สามารถพิมพ์ * เพื่อแทนตัวอักษรใดๆจำนวนเท่าไหร่ก็ได้ และพิมพ์ ? เพื่อแทนตัวอักษรใดๆ จำนวน 1 ตัวอักษร

รูปแบบ **regex;**

- Rawbyte ใช้เพื่อกำหนดให้ Snort ตรวจสอบแพ็กเก็ตจากข้อมูลดิบ (Raw packet data) ซึ่งเป็นข้อมูลที่ยังไม่ได้ผ่านการถอดรหัส

รูปแบบ **rawbyte;**

- URI content ใช้ตรวจหาข้อมูลในส่วนที่เป็น URI

รูปแบบ **uricontent:**[!] < ข้อมูลในส่วนของ URI ที่ต้องการ >;

- Byte test ใช้ทดสอบข้อมูลแบบไบนารี กับค่าที่กำหนด

รูปแบบ **byte_test:**< bytes to convert >, [!]< operator >, < value >, < offset > [,**relative**] [,<endian>] [,<number type>, string];

- Byte jump ใช้เพื่อให้ Snort ข้ามข้อมูลบางส่วนซึ่งตรงกับเงื่อนไขที่ใส่ลงไป

รูปแบบ **byte_jump:** < bytes_to_convert >, < offset > **[,relative]** [**,multiplier** < multiplier value >] [**,big**] [**,little**][**,string**][**,hex**] [**,dec**] [**,oct**] [**,align**] [**,from_beginning**];

3.3.2.3 IP Option บอกให้ Snort ทราบว่าต้องการตรวจสอบข้อมูลภายใน IP Header ด้วย

- TTL ใช้ตรวจสอบค่า Time To Live ที่ส่งมาในแพ็กเก็ต

รูปแบบ **ttl:** < จำนวน Time To Live ที่ต้องการตรวจสอบ >;

- TOS ใช้ตรวจสอบฟิลด์ Type Of Service (TOS) ใน IP Header ฟิลด์ TOS นี้ใช้ในการแสดงคำร้องขอการดำเนินการแบบพิเศษ การดำเนินการแบบไม่ต้องหน่วงเวลา หรือการดำเนินการที่ต้องการความรวดเร็วสูง และอื่นๆ แต่อย่างไรก็ตามปัจจุบันข้อมูลในฟิลด์นี้ไม่ได้รับความสนใจจากเราเตอร์แล้ว

รูปแบบ **tos:** < หมายเลข Type Of Service >;

- ID ใช้ตรวจสอบฟิลด์ fragment ID ใน IP Header

รูปแบบ **id:** < หมายเลข fragment >;

- Fragbits ใช้ตรวจสอบ fragment bit ซึ่งอยู่ที่ฟิลด์ flag ของ IP Header ฟิลด์นี้มีขนาด 3 บิตด้วยกัน ดังนี้

- R (Reserved Bit) มีค่าเป็น 0 เสมอ
- D (Don't Fragment) ถ้ามีค่าเป็น 1 แสดงว่าแพ็กเก็ตนี้ไม่สามารถแตกชิ้นส่วนได้
- M (More Fragment) แสดงว่าแพ็กเก็ตนี้มีการแตกชิ้นส่วน จะมีค่าเป็น 0 เมื่อเป็นชิ้นส่วนย่อยอันสุดท้าย

รูปแบบ **fragbits:** < ค่าของ fragment bit >;

- Fragment offset ใช้สำหรับตรวจจับแพ็กเก็ตที่มีค่าในฟิลด์ Fragment Offset ตรงกับเงื่อนไขที่กำหนดลงไป ซึ่งเงื่อนไขนั้นสามารถใส่เครื่องหมาย " < " หรือ " > " ตามด้วยค่า fragment offset หรือจะใส่แค่ค่า fragment offset เพียงอย่างเดียวก็ได้

รูปแบบ **fragoffset:**[< | >] < ค่า fragment offset >;

- IP protocol ใช้เพื่อให้กฎนั้นตรวจจับแพ็กเก็ตซึ่งมีค่าที่อยู่ในฟิลด์ Protocol ตรงกับค่าที่กำหนดไว้ โดยค่าที่ใส่ไว้จะเป็นชื่อโปรโตคอลหรือหมายเลขที่ใช้แทนโปรโตคอลก็ได้ (ท่านสามารถดูว่าหมายเลขใดแทนโปรโตคอลอะไรได้ที่ /etc/protocols)

รูปแบบ **ip_proto:**[! < >] < ชื่อโปรโตคอล หรือหมายเลขที่ใช้แทน >;

- IPoption เป็นส่วนเพิ่มเติมของกฎที่ช่วยให้ผู้ใช้สามารถระบุ IP Option ที่ต้องการตรวจจับได้ โดยปกติแล้วแพ็กเก็ตจะไม่มีข้อมูลในส่วนของ IP option ตารางที่ **3.2** แสดงรายการของ IP Option ที่ Snort ได้จัดเตรียมไว้

ตารางที่ 3.2 IP Option บนโปรแกรม Snort

IP Option	คำอธิบายอย่างสั้น
eol	End of list ใช้แสดงว่าเป็นจุดสิ้นสุดในรายการของ Option
lsrr	Loose source routing กำหนดการเลือกเส้นทางที่ฝั่งต้นทางและการบันทึกเส้นทางเป็นแบบหละหลวมโดยใช้เดตาแกรม
nop	No option ใช้เติมฟิลด์วางในรายการของ Option ให้เต็ม
rr	Record route ให้บันทึกเส้นทางที่เลือก แต่ละเส้นทางคือแอดเดรสที่อยู่ในช่องว่างที่จัดเตรียมไว้ และเป็นพื้นที่ในการปรับปรุงพอยเตอร์ที่ชี้ไปยังเราเตอร์
satid	Stream identifier ถูกกำหนดขึ้นมาใช้กับเครือข่าย Atlantic Satellite Network ซึ่งปัจจุบันเลิกการใช้งานไปแล้ว
sec	IP security option หรือที่รู้จักกันในชื่อ IPSec
ssrr	Strict source routing กำหนดการเลือกเส้นทางที่ฝั่งต้นทางและการบันทึกเส้นทางเป็นแบบเคร่งครัด ความแตกต่างของการเลือกแบบนี้กับแบบหละหลวมคือ เราเตอร์ ระหว่างทางมีความยืดหยุ่นในการเลือกเส้นทางให้แพ็กเก็ตได้น้อยกว่า
ts	Time Stamp กำหนดให้บันทึกเวลาที่เดตาแกรมถูกประมวลผล

รูปแบบ ipopts: < option >;

ข้อจำกัดของส่วนเพิ่มเติมนี้คือจะใส่ ipoption ได้เพียง 1 อย่างต่อกฎ 1 ข้อ

- Dsize ใช้สำหรับทดสอบขนาดของ Payload ในแพ็กเก็ตว่ามีขนาดผิดแปลกไปจากปกติหรือไม่

รูปแบบ dsize: [< >] < ขนาดของ Payload > [< ขนาดของ Payload >];

- Same IP ใช้เพื่อให้กฎนั้นตรวจจับแพ็กเก็ตที่มี หมายเลขไอพีต้นทางและปลายทางเหมือนกัน

รูปแบบ sameip;

3.3.2.4 TCP Option บอกให้ Snort ทราบว่าต้องการตรวจสอบข้อมูลภายใน TCP Header ด้วย

- Seq เป็นส่วนเพิ่มเติมที่ใช้ตรวจสอบค่า TCP Sequence number ในแพ็กเก็ต

รูปแบบ **seq:** < หมายเลข Sequence >;

- Ackใช้ตรวจสอบแพ็กเก็ตที่ผ่านกระบวนการ 3 way handshaking แล้ว (ACK Flag = 1) แต่มี Acknowledgment number เป็น 0 ซึ่งส่วนใหญ่จะเป็นแพ็กเก็ตที่เกิดจากโปรแกรม NMAP

รูปแบบ **ack:** < หมายเลขของ Acknowledgment >;

- Window ใช้ตรวจสอบขนาดของ window เพราะโปรแกรม back door บางตัวจะกำหนดขนาดของ window ให้มีขนาดใหญ่มากๆ

รูปแบบ **window:** < ขนาดของ window >;

- Flags ใช้สำหรับตรวจสอบ TCP Flags โดยค่า flag ที่โปรแกรม snort ตรวจสอบได้มีอยู่ 9 ค่าดังตารางที่ 3.3

ตารางที่ 3.3 TCP Flag ที่โปรแกรม Snort สามารถตรวจสอบได้

TCP Flags	คำอธิบายอย่างสั้น
A (ACK)	ตรวจสอบเมื่อ ACK Flag มีค่าเป็น 1
F (FIN)	ตรวจสอบเมื่อ FIN Flag มีค่าเป็น 1
P (PSH)	ตรวจสอบเมื่อ PSH Flag มีค่าเป็น 1
R (RST)	ตรวจสอบเมื่อ RST Flag มีค่าเป็น 1
S (SYN)	ตรวจสอบเมื่อ SYN Flag มีค่าเป็น 1
U (URG)	ตรวจสอบเมื่อ URG Flag มีค่าเป็น 1
0	ตรวจสอบถ้า TCP Flag ไม่มีการเซตค่า (เป็น 0 หมดทุกตัว)
1	ตรวจสอบเมื่อบิตที่ 1 ในฟิลด์ Reserved ของ TCP Header มีค่าเป็น 1
2	ตรวจสอบเมื่อบิตที่ 2 ในฟิลด์ Reserved ของ TCP Header มีค่าเป็น 1

รูปแบบ **flags:** < TCP Flag >;

3.3.2.5 ICMP Option บอกให้ Snort ทราบว่าต้องการตรวจสอบข้อมูลภายใน ICMP Header ด้วย

- Itype ใช้ตรวจสอบค่าฟิลด์ ICMP Typeสำหรับค่าที่สามารถกำหนดได้นั้นดูได้ที่ decode.h แพ็กเก็ตที่มีค่า ICMP Type ผิดปกติบางครั้งอาจมาจากการโจมตีแบบ Denial of service (DOS)

รูปแบบ **itype:** < หมายเลขของ ICMP Type >;

- Icode ใช้กำหนดค่าฟิลด์ ICMP Code ที่จะตรวจสอบโดยค่านี้จะเกี่ยวข้องกับค่าของ ICMP Type

รูปแบบ **icode:** [< | >] < หมายเลขของ ICMP Code > [< > < หมายเลขของ ICMP Code >];

- ICMP id ใช้กำหนดค่าฟิลด์ ICMP ID ที่จะตรวจสอบ

รูปแบบ **icmp_id:** < หมายเลขของ ICMP ID >;

- ICMP seq ใช้กำหนดค่าฟิลด์ ICMP Sequence ที่จะตรวจสอบ

รูปแบบ **icmp_seq:** < หมายเลขของ ICMP Sequence >;

3.3.2.6 Miscellaneous เป็น Rule Option ที่ไม่สามารถจัดเข้ากลุ่มใดได้

- Logto ใช้เพื่อกำหนดให้โปรแกรม Snort เก็บบันทึกแพ็กเก็ตซึ่งตรงกับกฎ ลงในไฟล์ที่กำหนด (Output log file) ข้อควรระวังคือส่วนเพิ่มเติมนี้ ไม่สามารถใช้งานได้ถ้า Snort อยู่ในโหมดการทำงานแบบ Binary logging

รูปแบบ **logto:** < ชื่อไฟล์ที่ต้องการเก็บผลลัพธ์ >;

- Session ใช้สำหรับดึงข้อมูลภายในแอพพลิเคชันที่ใช้ TCP Session ออกมา ตัวอย่างของข้อมูลเหล่านั้นได้แก่ ชื่อ รหัสผ่านของผู้ใช้ เป็นต้น

รูปแบบ **session: [printable | all];**

- RPC ใช้สำหรับตรวจหาการให้บริการแบบ RPC บนเครือข่าย

รูปแบบ **rpc:**< หมายเลขแอพพลิเคชัน >, < [< หมายเลขเวอร์ชัน > | *] >, < [< หมายเลขโพรซีเจอร์ > | *] >;

หมายเหตุ ในส่วนของหมายเลขเวอร์ชันและหมายเลขโพรซีเจอร์สามารถใส่ครื่องหมาย * ได้ โดยมีความหมายว่าเป็นเวอร์ชันและโพรซีเจอร์ใดก็ได้ สนใจเฉพาะหมายเลขแอพพลิเคชันก็พอ

- Resp เป็นส่วนที่จะพยายามปิดการเชื่อมต่อถ้ามีการแจ้งเตือนเกิดขึ้น

ตารางที่ 3.4 กลไกในการตอบสนอง

กลไกในการตอบสนอง	คำอธิบายอย่างสั้น
rst_snd	ส่งแพ็กเก็ต TCP-RST ผ่านทางซ็อกเก็ตที่ทำหน้าที่ส่งข้อมูล
rst_rcv	ส่งแพ็กเก็ต TCP-RST ผ่านทางซ็อกเก็ตที่ทำหน้าที่รับข้อมูล
rst_all	ส่งแพ็กเก็ต TCP-RST ไปทั้ง 2 ทิศทาง
icmp_net	ส่ง ICMP Net Unreach ไปยังผู้ส่งแพ็กเก็ต
icmp_host	ส่ง ICMP Host Unreach ไปยังผู้ส่งแพ็กเก็ต
icmp_port	ส่ง ICMP Port Unreach ไปยังผู้ส่งแพ็กเก็ต
icmp_all	ส่ง ICMP ที่กล่าวมาข้างต้นทั้งหมดไปยังผู้ส่งแพ็กเก็ต

หมายเหตุ: ผู้ใช้สามารถกำหนดกลไกในการตอบสนองได้มากกว่า 1

รูปแบบ **resp:**< กลไกในการตอบสนอง > [,< กลไกในการตอบสนอง > [,< กลไกในการตอบสนอง >]];

- React เป็นส่วนที่กำหนดว่าจะให้ทำอะไรหลังจากกลไกการตอบสนองทำงานไปแล้ว ใช้ตอบสนองนี้จะใช้งานได้ก็ต่อเมื่อมีการใช้ resp ค่าที่เป็นไปได้ของ react มี 2 ค่าคือ block กับ warn และส่วนเพิ่มเติมอีก 2 ค่าคือ msg กับ proxy รายละเอียดดังข้างล่างนี้

block จะทำการปิดการเชื่อมต่อ และส่งข้อความที่ผู้ใช้สังเกตเห็นได้

warn ส่งข้อความเตือนที่ผู้ใช้สังเกตเห็นได้

msg จะรวมข้อความในส่วนเพิ่มเติม msg ไปกับข้อความที่เกิดจากการ block

proxy : < หมายเลขพอร์ตของ proxy > จะส่งข้อความที่ผู้ใช้สังเกตเห็นได้ไปทางหมายเลขพอร์ตที่กำหนด

รูปแบบ **react:** < **block** | **warn** [, **msg** | **proxy:** < หมายเลขพอร์ต >];

- Tag นำมาใช้เพื่อบันทึกแพ็กเก็ตอื่นๆที่มีความเกี่ยวข้องกับแพ็กเก็ตที่กระตุ้นให้กฎทำงาน (trig) อาทิเช่นแพ็กเก็ตที่มาจากเครื่องเดียวกันกับแพ็กเก็ตที่กระตุ้นให้กฎทำงาน

รูปแบบ **tag:** < type >, < count >, < metric >, [direction];

type มีค่าได้ 2 อย่างคือ session หรือ host

count คือจำนวนที่ต้องการ tag ไว้ (หน่วยของ count นั้นจะอยู่ที่ metric)

metric คือหน่วยของจำนวนที่ต้องการ tag มีให้เลือก 2 อย่างคือ second กับ packet

3.4 ประเภทของกฎในโปรแกรม Snort

กฎทั้งหมดในโปรแกรม Snort แบ่งออกเป็น 48 ไฟล์ และ 2,039 สัญญาณดังตารางที่ 3.5

ตารางที่ 3.5 ชื่อไฟล์ของกฎที่ใช้ในการตรวจจับการบุกรุก

Rule Files Name	Number of Signature
attack-responses.rules	16
backdoor.rules	58
bad-traffic.rules	14
chat.rules	18
ddos.rules	33
deleted.rules	217
dns.rules	19
dos.rules	18
experimental.rules	0
exploit.rules	37
finger.rules	13
ftp.rules	50
icmp-info.rules	22
icmp.rules	93
imap.rules	16
info.rules	7
local.rules	0
misc.rules	45
multimedia.rules	6
mysql.rules	2
netbios.rules	26
nntp.rules	2
oracle.rules	25
other-ids.rules	3

ตารางที่ 3.5 ชื่อไฟล์ของกฎที่ใช้ในการตรวจจับการบุกรุก (ต่อ)

Rule Files Name	Number of Signature
p2p.rules	16
policy.rules	22
pop2.rules	4
pop3.rules	18
porn.rules	27
rpc.rules	125
rservices.rules	13
scan.rules	25
shellcode.rules	19
smtp.rules	25
snmp.rules	17
sql.rules	43
telnet.rules	14
tftp.rules	9
virus.rules	19
web-attacks.rules	47
web-cgi.rules	314
web-client.rules	6
web-coldfusion.rules	35
web-frontpage.rules	34
web-iis.rules	127
web-misc.rules	279
web-php.rules	59
X11.rules	2
Total of rule files	2,039

ไฟล์ที่เกี่ยวข้องกับการทำงานของโปรแกรม Snort ในโหมด NIDS

- Rule files (*.rules) คือไฟล์ที่บรรจุสัญลักษณ์หรือสัญญาณที่บ่งบอกถึงการบุกรุก
- snort.conf คือไฟล์ที่เก็บค่าพารามิเตอร์ต่างๆบนโปรแกรม Snort
- reference.config คือไฟล์ที่ใช้จับคู่ระหว่าง ชื่อแหล่งอ้างอิงที่ปรากฏบนกฎ (หลัง Rule Option "reference") กับ URL ของแหล่งที่มาของกฎ ประโยชน์ของไฟล์ reference.config คือช่วยให้การจัดการกับแหล่งอ้างอิงหรือแหล่งที่มาของกฎทำได้ง่าย อย่างเช่นกรณีมีการเปลี่ยนแปลง URL ของแหล่งที่มาของกฎ ก็เพียงเข้าไปแก้ไขในไฟล์นี้เท่านั้น ไม่ต้องไปตามแก้ไขในไฟล์กฎทุกๆไฟล์ที่เกี่ยวข้อง
- classification.config คือไฟล์ที่เก็บรายละเอียดประเภทของสัญลักษณ์หรือสัญญาณสำหรับตรวจ-จับการบุกรุก ซึ่งรายละเอียดเหล่านั้นก็จะมี ชื่อ คำอธิบายสั้นๆ และค่าความสำคัญของกลุ่มสัญลักษณ์(Rule files) โดยไฟล์นี้มีความสัมพันธ์โดยตรงกับ Rule Option ที่ชื่อ "classtype" ในการใช้งานถ้าต้องการสร้าง ลบ หรือแก้ไข ประเภทของกลุ่มสัญลักษณ์สำหรับตรวจจับการบุกรุก ผู้ดูแลระบบจะต้องเข้าไปแก้ไขในไฟล์นี้

3.5 รายละเอียดของไฟล์ Snort.conf

Snort.conf เป็นไฟล์ ที่ใช้กำหนดค่าพารามิเตอร์หลักให้กับโปรแกรม Snort จะประกอบไปด้วยส่วนประกอบหลัก 4 ส่วนคือ

3.5.1 ส่วนที่ใช้กำหนดค่าพารามิเตอร์ของเครือข่ายที่ต้องการตรวจจับการบุกรุก

3.5.2 ส่วนที่ใช้กำหนดค่า Preprocessors

3.5.3 ส่วนที่ใช้กำหนด Output plugins

3.5.4 ส่วนที่ใช้กำหนดไฟล์กฎในการตรวจจับการบุกรุก

3.6 สภาพแวดล้อมของ Snort Sensor

ในส่วนนี้จะกล่าวถึงรายละเอียดของซอฟต์แวร์ที่นำมาสร้าง Snort Sensor ซึ่งมีดังต่อไปนี้

3.6.1 Snort 2.0.4 คือไฟล์สำหรับติดตั้งโปรแกรม Snort

3.6.2 MySQL 4.0.16 คือไฟล์สำหรับติดตั้งโปรแกรม MySQL เป็น DBMS สำหรับจัดการกับฐานข้อมูล

3.6.3 Apache 1.3.33 คือไฟล์สำหรับติดตั้งโปรแกรม Apache มีทำหน้าที่จำลองการทำงานของเครื่องที่ติดตั้งให้เป็นเว็บเซิร์ฟเวอร์

3.6.4 PHP 4.3.4 คือไฟล์ที่ติดตั้งเพื่อทำให้เครื่องเว็บเซิร์ฟเวอร์สามารถรันสคริปต์ได้ ใช้ในการสร้าง ไดนามิกเว็บเพจ

3.6.5 ACID 0.9.6b23 คือไฟล์สำหรับติดตั้ง Analysis Console for Intrusion Database (ACID) ซึ่งมีหน้าที่สำหรับวิเคราะห์ข้อมูลจากฐานข้อมูลของ IDS เขียนขึ้นโดยใช้ PHP เป็นผลงานของ Roman Danyliw ซึ่งเป็นส่วนหนึ่งของโครงการ AIRCERT

3.6.6 LibPcap 0.7.2 คือไฟล์ที่ติดตั้งลงไปเพื่อทำให้เครื่องคอมพิวเตอร์นั้นสามารถดักจับข้อมูลบนเครือข่ายได้

3.6.7 ADODB 4.01 คือไฟล์ที่ติดตั้งลงไปเพื่อทำให้ PHP สามารถติดต่อกับฐานข้อมูลได้

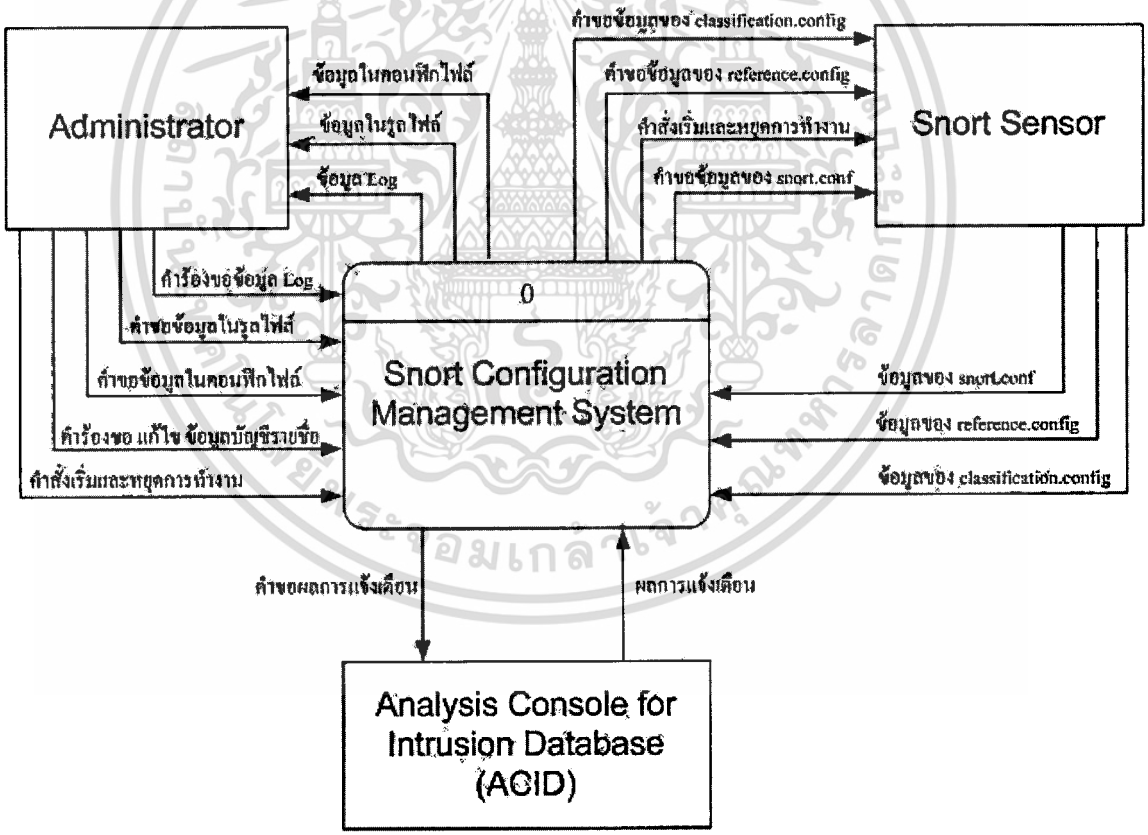
บทที่ 4

การออกแบบและพัฒนาระบบ

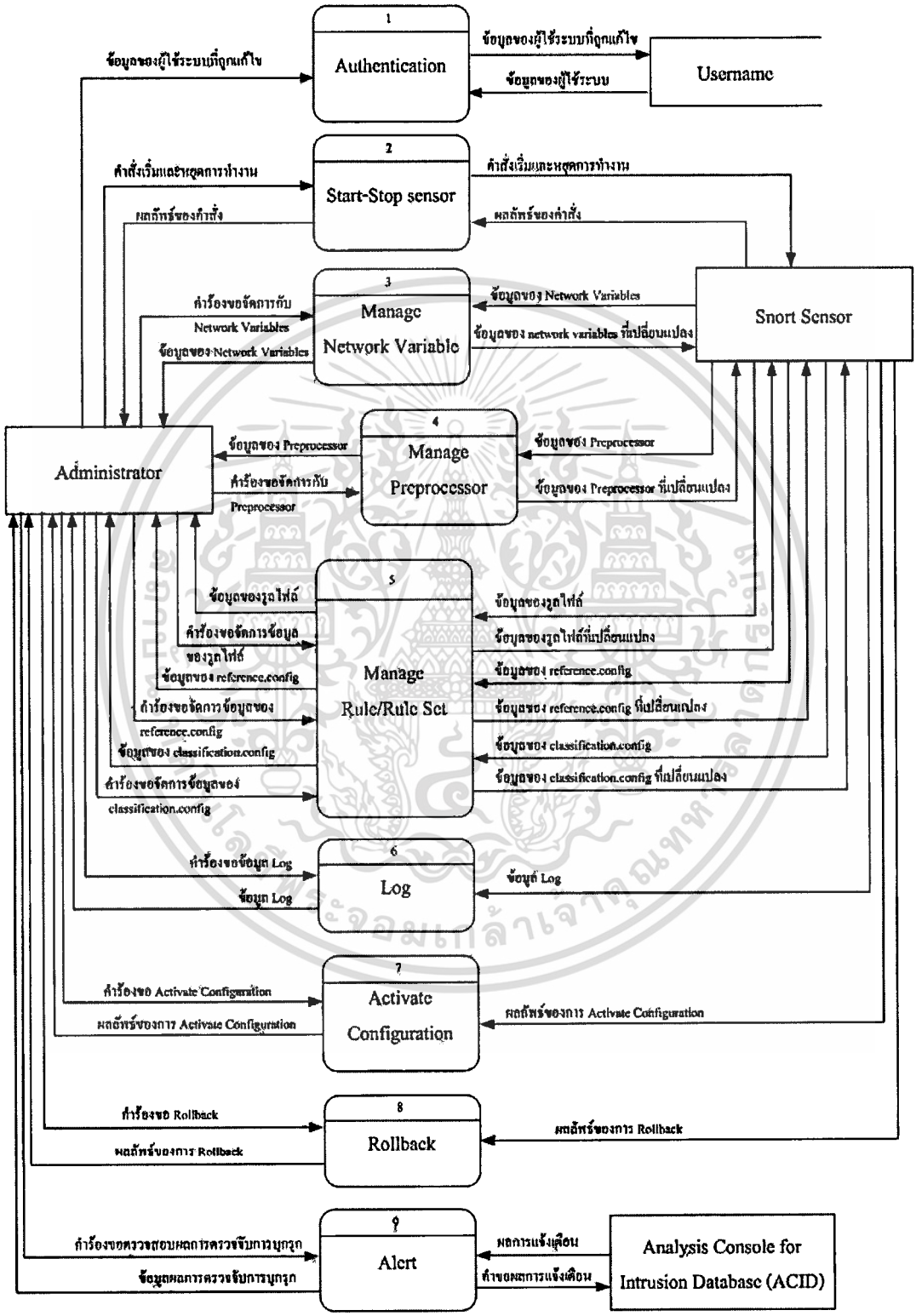
4.1 ออกแบบฟังก์ชันการทำงานของระบบ

จากการที่ได้ศึกษาการทำงานของโปรแกรม Snort รวมถึงไฟล์ที่ใช้เก็บคอนฟิกูเรชันและกฎ ทำให้เห็นภาพรวมของระบบที่จะพัฒนาและเขียนออกมาเป็นไดอะแกรมต่างๆดังรูปที่ 4.1 – 4.2

4.1.1 แผนภาพกระแสข้อมูล (Data Flow Diagram)



รูปที่ 4.1 แผนภาพกระแสข้อมูลระดับ 0 ของระบบ



รูปที่ 4.2 แผนภาพกระแสข้อมูลระดับ 1 ของระบบ

จากแผนภาพกระแสข้อมูลระดับที่ 1 ระบบจะประกอบไปด้วยฟังก์ชันการทำงาน 9 ฟังก์ชันดังนี้

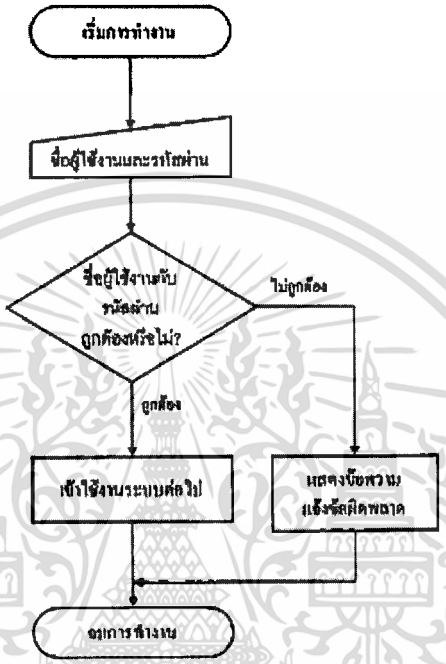
- Authentication ใช้สำหรับตรวจสอบว่าผู้เข้ามาใช้งานมีสิทธิ์เข้าใช้ระบบหรือไม่ ระบบมีสิทธิ์เพียงหนึ่งประเภทเท่านั้นคือ สิทธิ์แบบผู้ดูแลระบบ (Administrator) การตรวจสอบจะนำชื่อผู้ใช้งานและรหัสผ่านที่ผู้ใช้กรอกไว้ในช่องรับข้อความ ไปตรวจกับข้อมูลในฐานข้อมูล ในกรณีที่ผู้ใช้งานกรอกรายละเอียดไม่ครบก็จะมีข้อความเตือนขึ้นมา
- Start/Stop Sensor สำหรับสั่งให้ Snort Sensor ทำงานหรือหยุดทำงาน โดยระบบจะส่งคำสั่งไปยัง Snort sensor และตรวจสอบ Process บน Snort Sensor ว่าคำสั่งนั้นทำงานแล้วหรือไม่ จากนั้นจะแสดงผลให้ผู้ใช้งาน โดยถ้า Snort Sensor ทำงานอยู่ จะแสดงที่เมนูว่า Stop Sensor แต่ถ้าปิดการทำงานอยู่จะแสดงข้อความที่เมนูว่า Start Sensor
- Manage Network Variable ใช้สำหรับจัดการกับค่าตัวแปรต่างๆบนเครือข่ายที่ต้องกำหนดให้กับโปรแกรม Snort โดยจะแสดงค่าเหล่านั้นให้ผู้ดูแลระบบ ผู้ดูแลระบบสามารถเพิ่มเติม แก้ไข ลบ หรือ ไม่ใช้งานค่านั้นชั่วคราวได้
- Manage Preprocessor ทำหน้าที่รับและแสดงข้อมูลต่างๆที่ใช้สำหรับปรับแต่ง Preprocessor ผู้ดูแลระบบสามารถแก้ไข หรือ ไม่ใช้งาน Preprocessor ชั่วคราวได้
- Manage Rule/Rule_Set ทำหน้าที่รับและแสดงข้อมูลต่างๆที่ใช้สำหรับปรับแต่งกฎ เมื่อมีการปรับแต่งกฎ
- Manage Log ทำหน้าที่เก็บบันทึกกิจกรรมที่ผู้ใช้สั่งงานที่เกี่ยวข้องกับค่าต่างๆใน snort.conf ข้อมูลจะถูกบันทึกไฟล์ที่มีชื่อว่า log.txt
- Manage Alert ทำหน้าที่แสดงการแจ้งเตือนต่างๆให้อยู่บนรูปแบบของ HTML ซึ่งจำเป็นต้องทำงานร่วมกับโปรแกรม Analysis Console for Intrusion Databases (ACID)
- Activate Configuration ทำหน้าที่นำค่าคอนฟิกูเรชัน และกฎที่ผู้ใช้ได้เพิ่มเติม แก้ไข ไปใช้งานจริง พร้อมทั้งเก็บค่าคอนฟิกูเรชัน และกฎอันเก่าไว้ เพื่อนำไปใช้ในการกู้ข้อมูลคืนภายหลัง (Rollback)
- Rollback ทำหน้าที่กู้ค่าคอนฟิกูเรชัน และกฎเก่า คืนกลับมา

หมายเหตุ การ Activate Configuration และ Rollback นั้นจะต้องมีการหยุดการทำงานของ Sensor ก่อน และเพื่อป้องกันความผิดพลาดนี้ ทั้ง 2 ฟังก์ชันจะมีการตรวจสอบการทำงานของ Sensor ก่อน ถ้ายังไม่ได้หยุดการทำงาน จะมีข้อความเตือนขึ้นมา

4.1.2 แผนผังแสดงการทำงานของฟังก์ชันหลักในระบบ (Flow Chart)

4.1.2.1 ฟังก์ชันสำหรับตรวจสอบสิทธิ์ผู้ใช้งาน (Authentication)

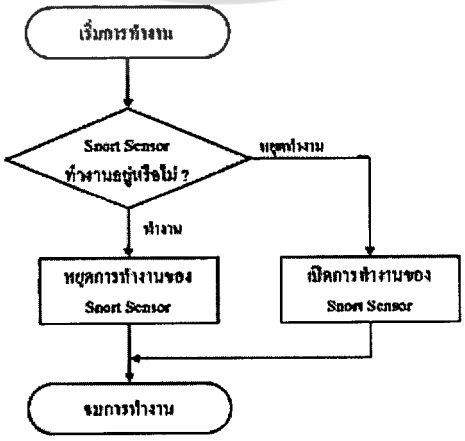
ผู้ใช้จะต้องกรอกชื่อผู้ใช้งานและรหัสผ่านให้ถูกต้องจึงจะมีสิทธิ์เข้าใช้ระบบ ฟังก์ชันนี้จะมีกระบวนการทำงานดังแสดงในรูปที่ 4.3



รูปที่ 4.3 แผนผังแสดงการทำงานของฟังก์ชันตรวจสอบสิทธิ์ผู้ใช้งาน

4.1.2.2 ฟังก์ชันเปิด-ปิดการทำงานของ Snort Sensor (Start-Stop Snort Sensor)

เพื่อไม่ให้การทำงานของ Sensor มีข้อผิดพลาดเกิดขึ้น จึงต้องมีการปิดหรือหยุด Sensor ก่อนทำการ Activate Configuration หรือ Rollback กระบวนการทำงานของฟังก์ชันเปิด-ปิด Snort Sensor จะแสดงไว้ในรูปที่ 4.4



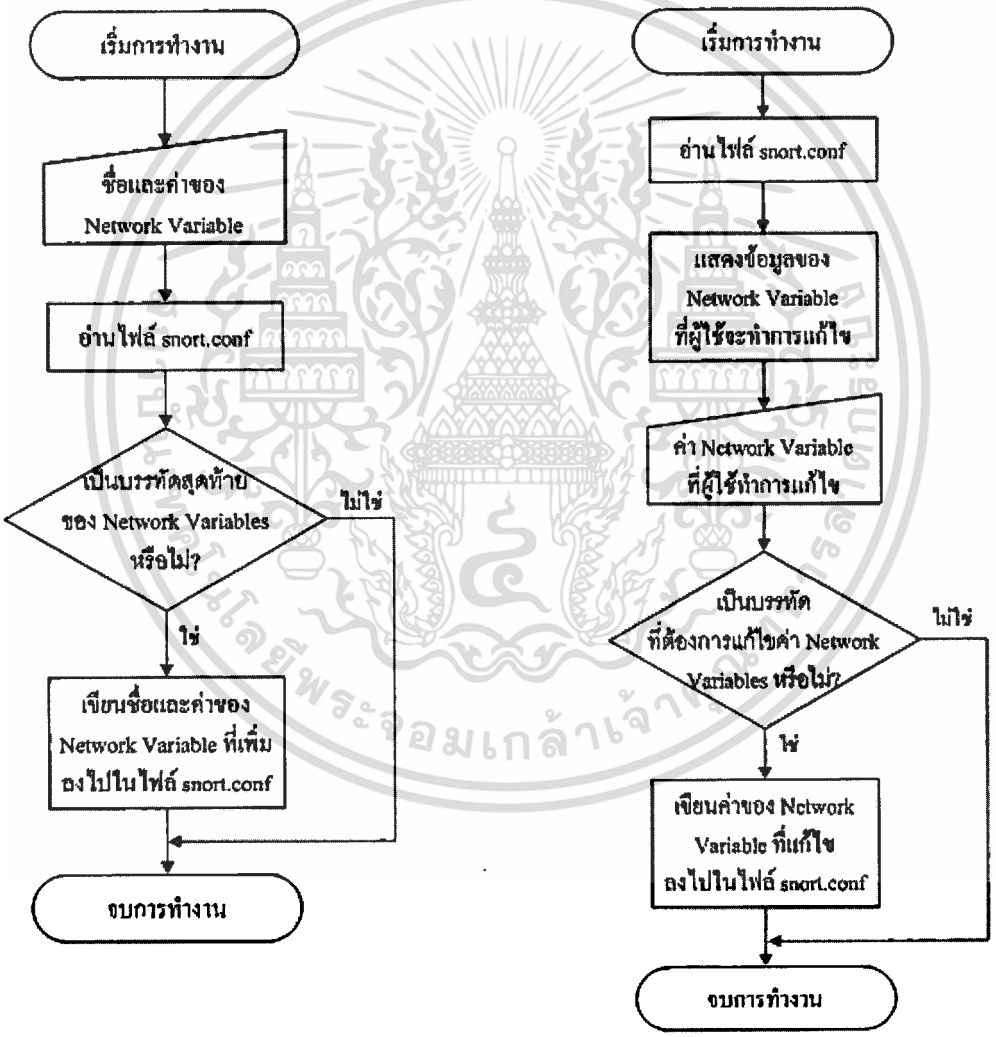
รูปที่ 4.4 แผนผังแสดงการทำงานของฟังก์ชันเปิด-ปิด Snort Sensor

4.1.2.2 ฟังก์ชันสำหรับจัดการกับค่าตัวแปรเครือข่าย (Network Variable Management)

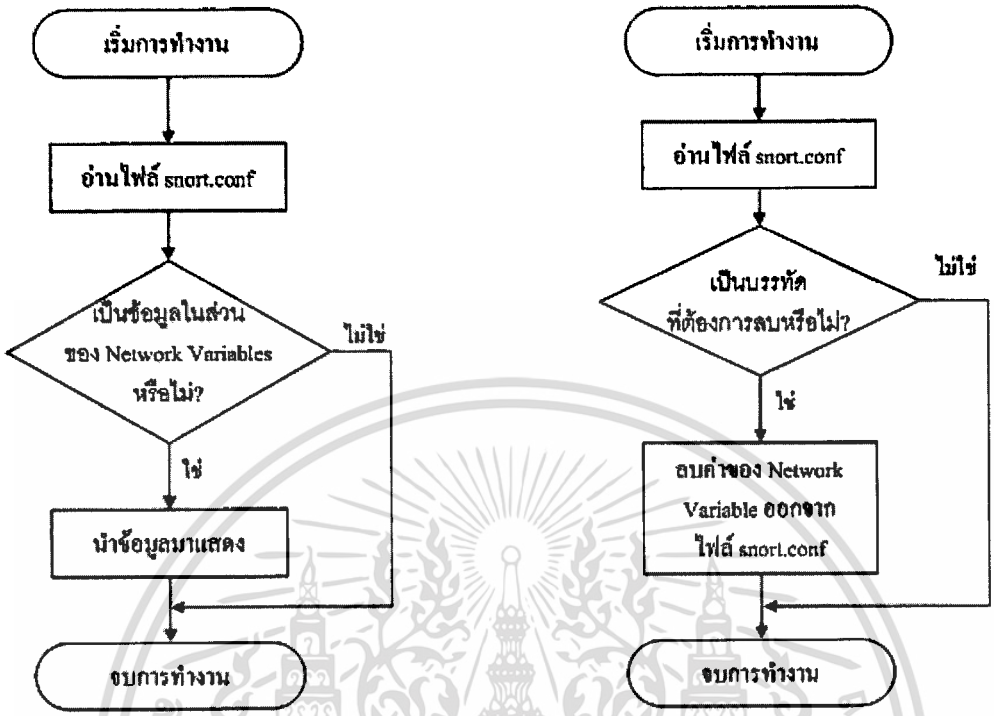
ฟังก์ชันส่วนนี้จะประกอบด้วยฟังก์ชันย่อย 4 ฟังก์ชันด้วยกันคือ

- ฟังก์ชันย่อยสำหรับแสดงค่าตัวแปรเครือข่าย
- ฟังก์ชันย่อยสำหรับเพิ่มตัวแปรเครือข่าย
- ฟังก์ชันย่อยสำหรับแก้ไขค่าตัวแปรเครือข่าย
- ฟังก์ชันย่อยสำหรับลบตัวแปรเครือข่าย

กระบวนการทำงานของฟังก์ชันย่อยทั้งหมดจะแสดงไว้ในรูปที่ 4.5 และ 4.6 ตามลำดับ



รูปที่ 4.5 แผนผังแสดงฟังก์ชันย่อยสำหรับเพิ่มและแก้ไขค่าตัวแปรเครือข่าย



รูปที่ 4.6 แผนผังแสดงการทำงานของฟังก์ชันย่อย แสดงและลบค่าตัวแปรเครือข่าย

4.1.2.3 ฟังก์ชันสำหรับจัดการกับ Preprocessor (Preprocessor Management)

จะประกอบไปด้วยฟังก์ชันย่อย 2 ฟังก์ชันคือ ฟังก์ชันย่อยสำหรับแสดงค่า Preprocessor และฟังก์ชันสำหรับแก้ไขค่าเพิ่มเติมของ Preprocessor กระบวนการทำงานก็จะคล้ายกับฟังก์ชันย่อยสำหรับแสดงและแก้ไขค่าตัวแปรเครือข่าย จะแตกต่างกันก็เพียงตัวอักษรที่นำมาแยกแยะข้อมูล ว่าส่วนใดเป็นของ Preprocessor เท่านั้นเอง

4.1.2.4 ฟังก์ชันสำหรับจัดการกับกลุ่มกฎและกฎ (Rule Set and Rule Management)

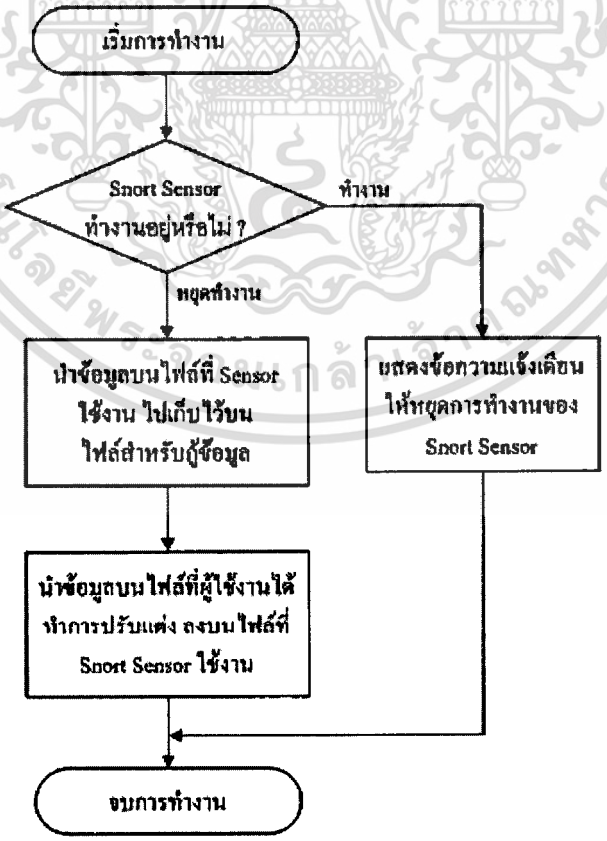
จะประกอบไปด้วยฟังก์ชันย่อย 7 ฟังก์ชันคือ

- ฟังก์ชันย่อยสำหรับแสดงกลุ่มกฎ
- ฟังก์ชันย่อยสำหรับแสดงกฎ
- ฟังก์ชันย่อยสำหรับเพิ่มเติมกฎ
- ฟังก์ชันย่อยสำหรับแก้ไขกฎ
- ฟังก์ชันย่อยสำหรับลบกฎ
- ฟังก์ชันย่อยสำหรับจัดการประเภทของกฎ
- ฟังก์ชันย่อยสำหรับจัดการแหล่งที่มาของกฎ

ฟังก์ชันย่อยสำหรับแสดงกลุ่มกฎ จะมีกระบวนการทำงานก็คล้ายคลึงกับฟังก์ชันสำหรับจัดการค่าตัวแปรเครือข่าย กล่าวคือฟังก์ชันย่อยสำหรับแสดงกลุ่มกฎก็จะนำข้อมูลที่ได้จากการอ่านไฟล์ snort.conf แล้วแยกเอาเฉพาะส่วนที่เกี่ยวกับกลุ่มกฎมาแสดง แต่จะต้องมีการเก็บข้อมูลที่เป็นชื่อไฟล์กลุ่มกฎ (*.rules) ไว้เพราะฟังก์ชันย่อยสำหรับแสดง เพิ่มเติม แก้ไข และลบกฎ จะต้องนำชื่อไฟล์กลุ่มกฎไปเปิดหาข้อมูลของกฎอีกทอดนึง สำหรับฟังก์ชันที่ใช้จัดการประเภทและแหล่งที่มาของกฎ กระบวนก็คล้ายคลึงกับฟังก์ชันสำหรับจัดการค่าตัวแปรเครือข่ายด้วยเช่นกัน เพียงแต่ว่าอ่านข้อมูลจากไฟล์คนละไฟล์กันอีกทั้งใช้ตัวอักษรในการแยกแยะข้อมูลต่างกัน โดยฟังก์ชันสำหรับจัดการประเภทของกฎจะอ่านข้อมูลจากไฟล์ classification.config ส่วนฟังก์ชันสำหรับจัดการแหล่งที่มาของกฎจะอ่านข้อมูลจากไฟล์ reference.config

4.1.2.4 ฟังก์ชันสำหรับนำข้อมูลที่ผู้ใช้ปรับแต่งไว้ไปใช้งานจริง (Activate Configuration)

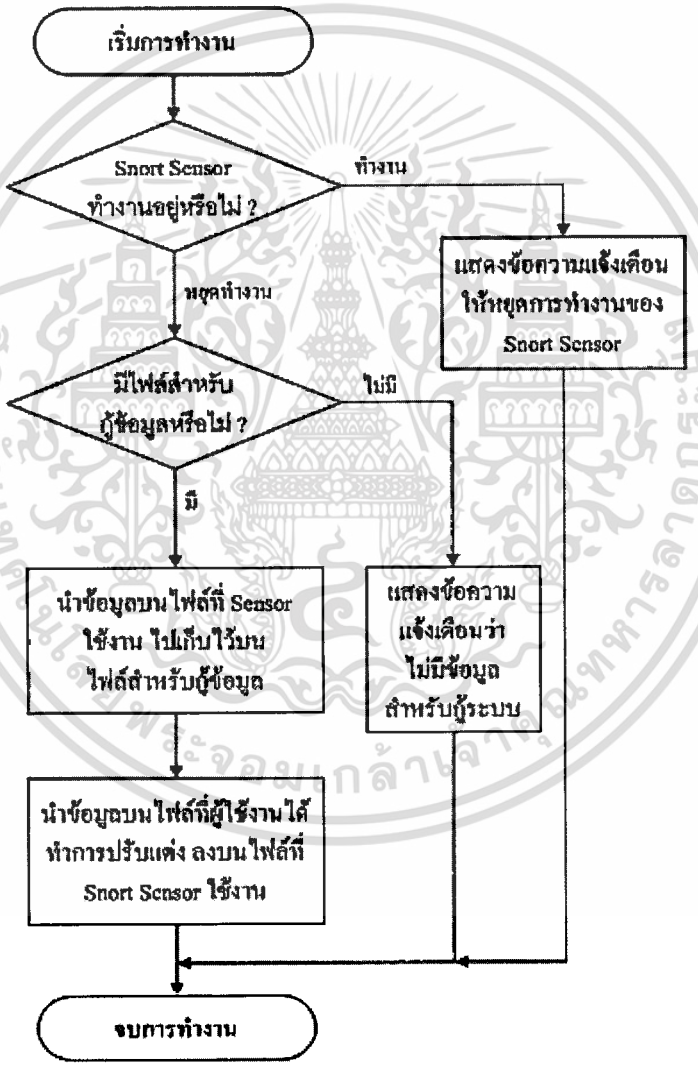
ทำหน้าที่นำค่าคอนฟิกูเรชัน และกฎที่ผู้ใช้ได้เพิ่มเติม แก้ไข ไปใช้งานจริง พร้อมทั้งเก็บค่าคอนฟิกูเรชัน และกฎอันเก่าไว้ เพื่อนำไปใช้ในการกู้ข้อมูลคืนภายหลัง (Rollback) กระบวนการทำงานจะแสดงไว้ในรูปที่ 4.7



รูปที่ 4.7 แผนผังแสดงการทำงานของฟังก์ชันActivate Configuration

4.1.2.5 ฟังก์ชันสำหรับกู้คืนข้อมูล (Rollback)

ในกรณีค่าคอนฟิกูเรชันหรือกฎที่ผู้ใช้ปรับแต่งไปแล้วทำให้เกิดปัญหา และคิดว่าการนำค่าเก่ามาใช้น่าจะเป็นทางออกที่ดีที่สุดแล้ว ฟังก์ชันที่เกี่ยวข้องโดยตรงก็คือฟังก์ชันกู้คืนข้อมูล แต่ฟังก์ชันนี้จะทำงานได้ก็ต่อเมื่อผู้ใช้มีการเก็บค่าคอนฟิกูเรชันหรือกฎเก่าๆเอาไว้แล้ว กล่าวอีกนัยก็คือได้มีการ Activate Configuration แล้วนั่นเอง กระบวนการทำงานของฟังก์ชันจะแสดงไว้ในรูปที่ 4.8



รูปที่ 4.8 แผนผังแสดงการทำงานของฟังก์ชันกู้คืนข้อมูล

4.2 ออกแบบโครงสร้างของส่วนที่ใช้ติดต่อกับผู้ใช้งาน

เนื่องจากระบบที่พัฒนาขึ้นมีจุดประสงค์ให้ผู้ใช้งานโปรแกรม Snort สามารถปรับแต่งค่าคอนฟิกูเรชันและกฎได้ง่ายขึ้น ส่วนติดต่อกับผู้ใช้จึงมีความสำคัญเป็นอย่างมาก ผู้พัฒนาจึงได้นำเอา WND มาใช้ในการออกแบบ

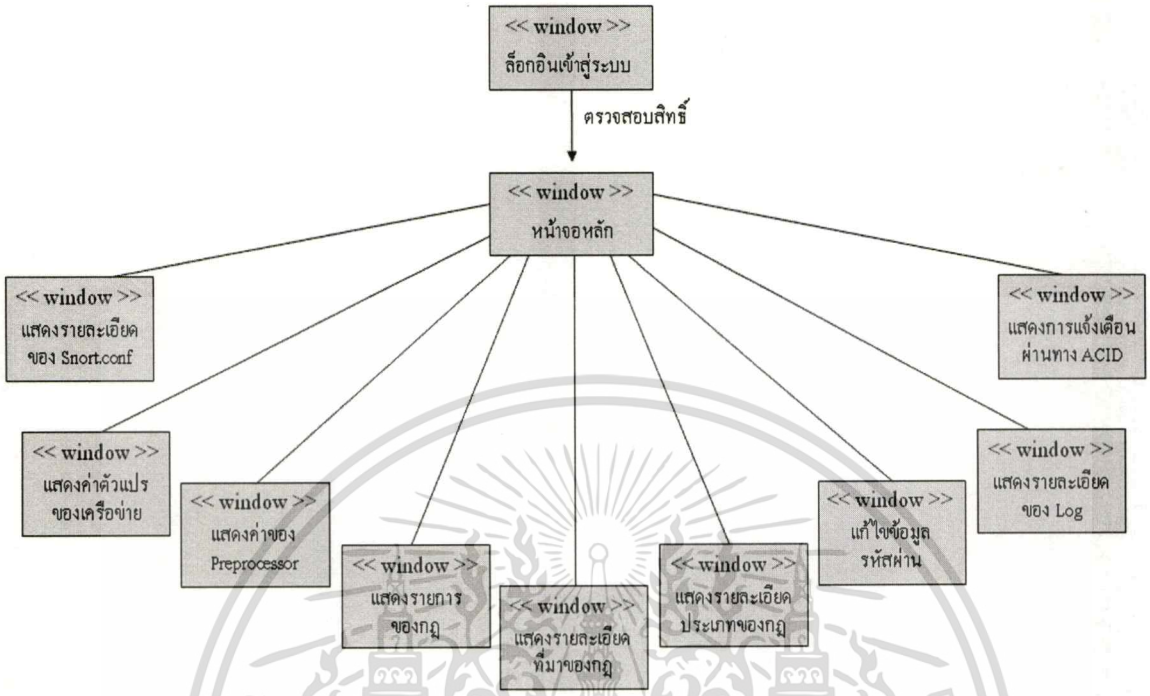
4.2.1 Window Navigation Diagram (WND)

4.2.1.1 ส่วนของการทำงานหลัก

เมื่อทำการล็อกอินเข้าสู่ระบบจะเข้าสู่เมนูหลักซึ่งสามารถเลือกการทำงานได้ดังนี้

1. สั่งเปิดหรือปิดการทำงานของ Snort Sensor
2. แสดงรายละเอียดของไฟล์ snort.conf
3. แก้ไขข้อมูลรหัสผ่าน
4. แสดงรายละเอียดของ Log
5. แสดงค่าตัวแปรของเครือข่าย
6. แสดงค่า Preprocessor
7. แสดงรายการกฎ
8. แสดงที่มาของกฎ
9. แสดงประเภทของกฎ
10. แสดงผลการแจ้งเตือนผ่านทาง ACID
11. ออกจากระบบ

แต่ไม่ใช่ว่าเมนูที่ถูกเลือกนั้นจะขึ้นหน้าใหม่เสมอไป จะมีเพียง 9 เมนูเท่านั้นที่จะขึ้นหน้าใหม่ สำหรับรายละ เอียดนั้นได้แสดงไว้ในรูปที่ 4.9



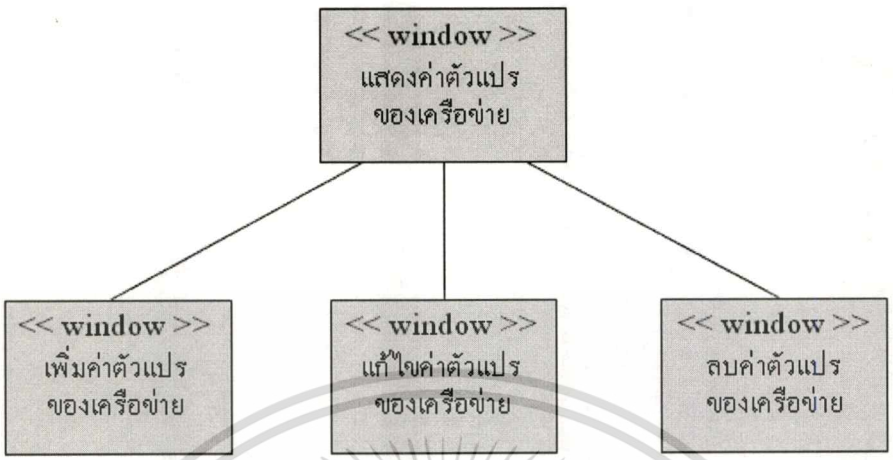
รูปที่ 4.9 Window Navigation Diagram ของส่วนเมนูหลัก

4.2.1.2 ส่วนปรับแต่งค่าคอนฟิกูเรชันและกฎ

ส่วนปรับแต่งค่าคอนฟิกูเรชันและกฎนั้นจะแบ่งออกเป็น 5 ส่วนย่อยดังนี้

1. ส่วนปรับแต่งค่าตัวแปรเครือข่าย
2. ส่วนปรับแต่งค่า Preprocessor
3. ส่วนปรับแต่งกฎ
4. ส่วนปรับแต่งที่มาของกฎ
5. ส่วนปรับแต่งประเภทของกฎ

ส่วนปรับแต่งค่าตัวแปรเครือข่ายนั้นใช้สำหรับเพิ่มเติม แก้ไข หรือลบค่าตัวแปรที่เกี่ยวข้องกับเครือข่าย ซึ่งค่าทั้งหมดจะปรากฏอยู่ที่ไฟล์ snort.conf ตัวอย่างค่าตัวแปรเครือข่ายเช่น $EXTERNAL_NET $HOME_NET $DNS_SERVER $SMTP_SERVER เป็นต้น

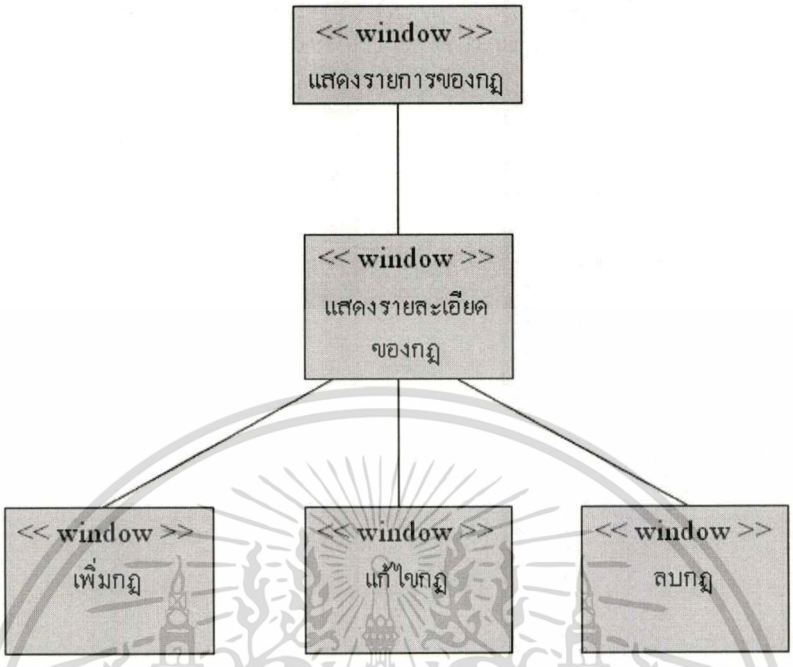


รูปที่ 4.10 Window Navigation Diagram ของส่วนปรับแต่งค่าตัวแปรเครือข่าย

ส่วนปรับแต่งค่า Preprocessor ใช้สำหรับแก้ไขส่วนเพิ่มเติมของ Preprocessor ที่มีอยู่ในโปรแกรม Snort ซึ่งรายละเอียดทั้งหมดจะปรากฏอยู่ที่ไฟล์ snort.conf โดยจะอยู่ต่อจากค่าตัวแปรเครือข่าย

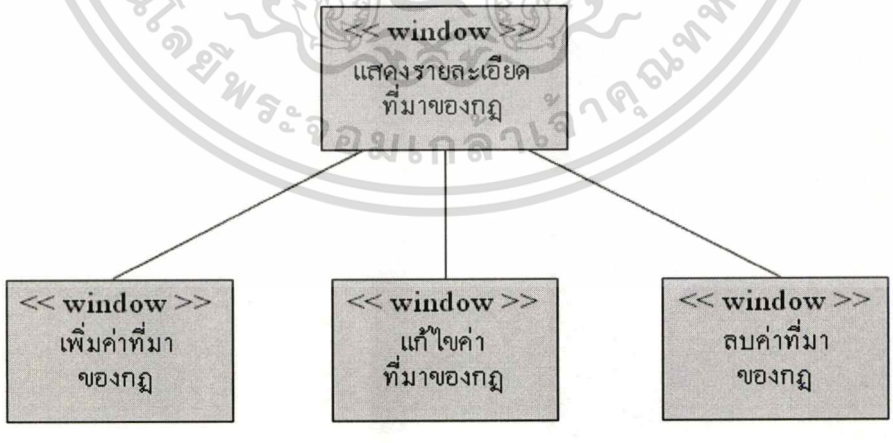
รูปที่ 4.11 Window Navigation Diagram ของส่วนปรับแต่งค่า Preprocessor

ส่วนปรับแต่กฎ ใช้สำหรับจัดการกับกฎ (Rule) ที่อยู่ภายในกลุ่มของกฎ (Rule Set) การจัดการก็จะมีการเพิ่ม แก้ไข และลบกฎ ส่วนนี้ถือเป็นส่วนสำคัญที่สุดในการจัดการกับ Snort Sensor เพราะถ้ากฎที่เขียนขึ้นมาไม่ดีก็จะทำให้การบุกรุกบางอย่างหลุดรอดจากการตรวจสอบไปได้ โดยในส่วนเริ่มต้นจะมีหน้าต่างแสดงรายการของกลุ่มกฎปรากฏขึ้นมา ผู้ใช้สามารถเข้าไปดูกฎในกลุ่มกฎที่ต้องการได้ซึ่งจะปรากฏเป็นหน้าต่างใหม่ ในหน้าต่างแสดงกฎนี้ผู้ใช้สามารถ เพิ่มเติม แก้ไข หรือลบกฎได้ การทำงานในส่วนนี้จะเกี่ยวข้องกับไฟล์ snort.conf และไฟล์กฎที่ผู้ใช้เลือก (ไฟล์กฎทั้งหมดนั้นจะต้องเก็บไว้ใน /etc/snort/)



รูปที่ 4.12 Window Navigation Diagram ของส่วนปรับแต่งกฎ

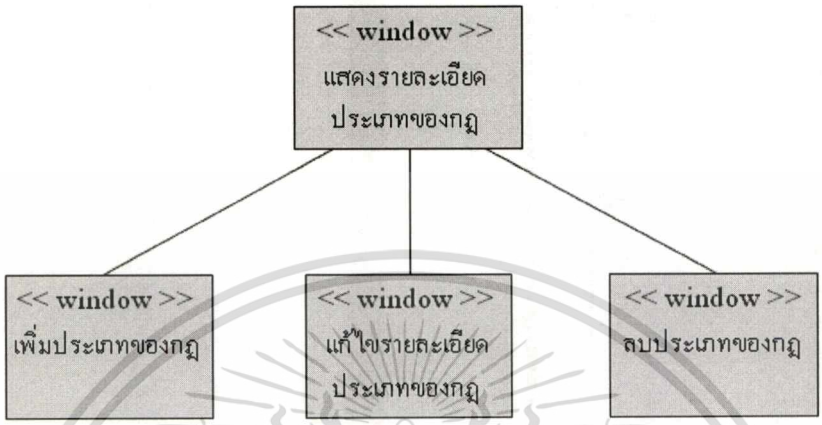
ส่วนปรับแต่งที่มาของกฎ ใช้ปรับปรุงแก้ไข เพิ่มเติมหรือลบแหล่งอ้างอิงของกฎที่ได้มาจากภายนอก(นอกเหนือจากกฎที่สร้างโดย snort.org) เพื่อทำให้ข้อมูลนั้นถูกต้องทันสมัยอยู่เสมอการทำงานในส่วนนี้จะเกี่ยวข้องกับไฟล์ reference.config



รูปที่ 4.13 Window Navigation Diagram ของส่วนปรับแต่งที่มาของกฎ

โดยปกติกฎทุกข้อของโปรแกรม Snort จะต้องมีการจัดเข้าอยู่ในประเภทใดประเภทหนึ่ง ซึ่งรายละเอียดประเภทของกฎทั้งหมดนั้นแสดงไว้ในไฟล์ classification.config ส่วนสุดท้ายที่ใช้ในการปรับตั้งค่าคอนฟิกูเรชันและกฎก็คือส่วนที่ใช้ปรับแต่งประเภทของกฎนั่นเอง ส่วนนี้มีหน้าที่ใน

การเพิ่มเติม แก้ไข หรือลบ ประเภทของกฎที่อยู่ในไฟล์ classification.config รายละเอียดของ WND จะแสดงไว้ในรูปที่ 4.14



รูปที่ 4.14 Window Navigation Diagram ของส่วนปรับแต่งประเภทของกฎ

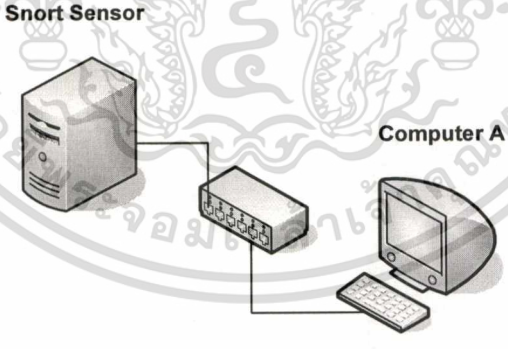
บทที่ 5
การทดสอบระบบ

5.1 วัตถุประสงค์ในการทดสอบระบบ

- เพื่อแสดงให้เห็นว่าระบบสามารถเปลี่ยนแปลง แก้ไขค่าบนไฟล์คอนฟิกูเรชันและกฎบนไฟล์กฎที่เกี่ยวข้องของโปรแกรม Snort ได้
- เพื่อแสดงให้เห็นว่าระบบสามารถตรวจสอบและแจ้งข้อผิดพลาดจากการปรับตั้งค่าคอนฟิกู-เรชันที่ไม่สอดคล้องได้

5.2 วิธีทดสอบระบบ

การทดสอบระบบมีอยู่ด้วยกัน 7 ตอน โดยจะต่อระบบคอมพิวเตอร์ดังรูปที่ 5.1 สำหรับการทดสอบตอนที่ 1 ถึง 5 นั้นเราจะเปรียบเทียบผลที่ปรากฏบนหน้าเว็บบราวเซอร์กับข้อมูลในไฟล์ ที่เกี่ยวข้อง ส่วนตอนที่ 6 จะแสดงให้เห็นข้อความแจ้งเตือนจากการปรับแต่งค่าไม่เหมาะสม ท้ายสุดในตอนที่ 7 จะทำให้เซ็นเซอร์แจ้งเตือนตามกฎที่ได้เพิ่มเข้าไปในกลุ่มกฎ local

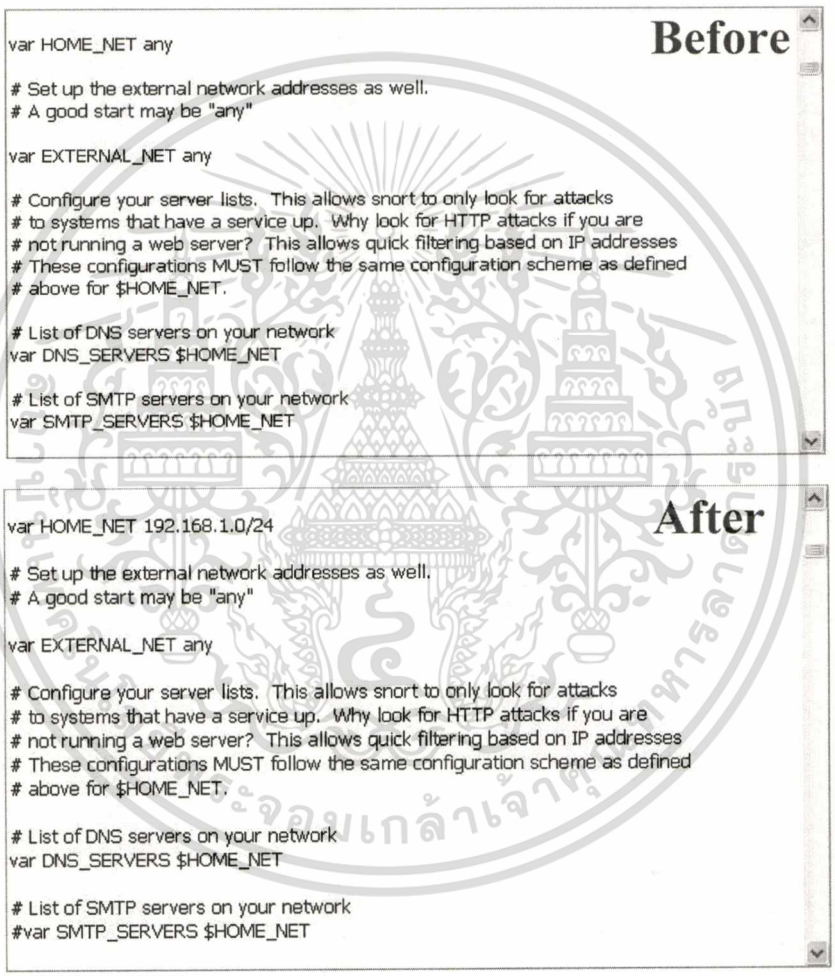


รูปที่ 5.1 ระบบคอมพิวเตอร์ที่ใช้ทดสอบ

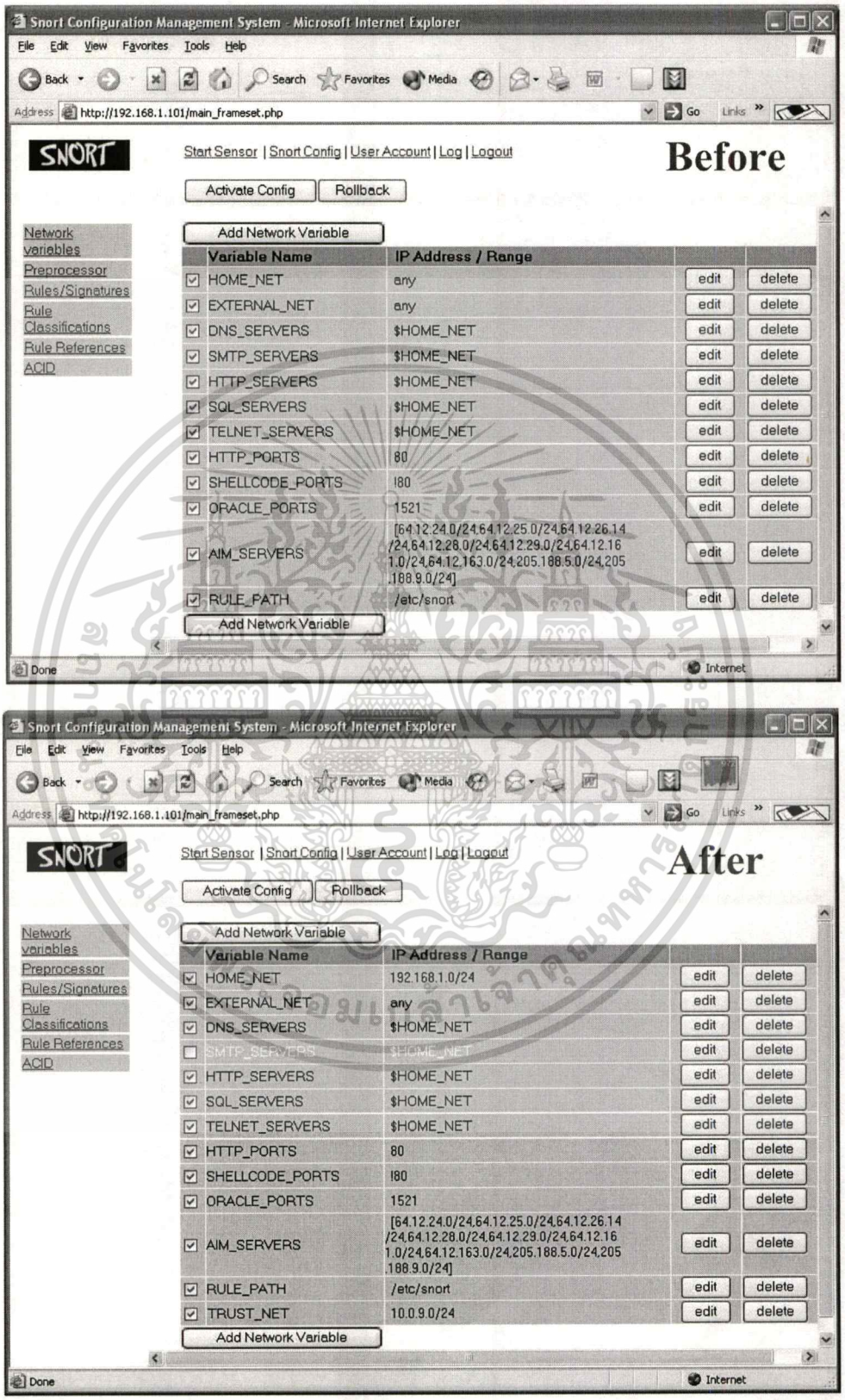
ที่เครื่อง A ให้เปิดเว็บบราวเซอร์แล้วพิมพ์ URL ดังนี้ http://192.168.1.101/index.html เมื่อเว็บเพจแสดงหน้าจอสำหรับการเข้าใช้งานระบบ (Login) ให้ใส่ชื่อผู้ใช้งานและรหัสผ่านเป็น snort และ itsnort ตามลำดับ

การทดลองตอนที่ 1 การปรับแต่งค่าตัวแปรเครือข่าย (Network Variables)

การทดลองนี้จะทำการแก้ไขค่าของ HOME_NET จากเดิมคือ any ไปเป็น 192.168.1.0/24 เปลี่ยนสถานะของ SMTP_SERVER จากทำงานเป็นไม่ทำงาน (เอาเครื่องหมายถูกที่เช็คบ็อกซ์ออก) และเพิ่มตัวแปรเครือข่ายที่ชื่อ TRUST_NET เข้าไป โดยค่าของ TRUST_NET คือ 10.0.9.0/24



รูปที่ 5.2 เปรียบเทียบข้อมูลส่วนที่เกี่ยวข้องกับตัวแปรเครือข่าย ใน snort.conf ก่อนและหลังการปรับเปลี่ยนค่า



รูปที่ 5.3 เปรียบเทียบหน้าจอแสดงข้อมูลของตัวแปรเครือข่ายก่อนและหลังการปรับเปลี่ยนค่า

การทดลองตอนที่ 2 การปรับแต่งค่า Preprocessor

การทดลองนี้จะเพิ่มค่า Option ของ Preprocessor ที่ชื่อว่า frag2 โดยค่า Option ที่เพิ่มเข้าไปคือ timeout 90

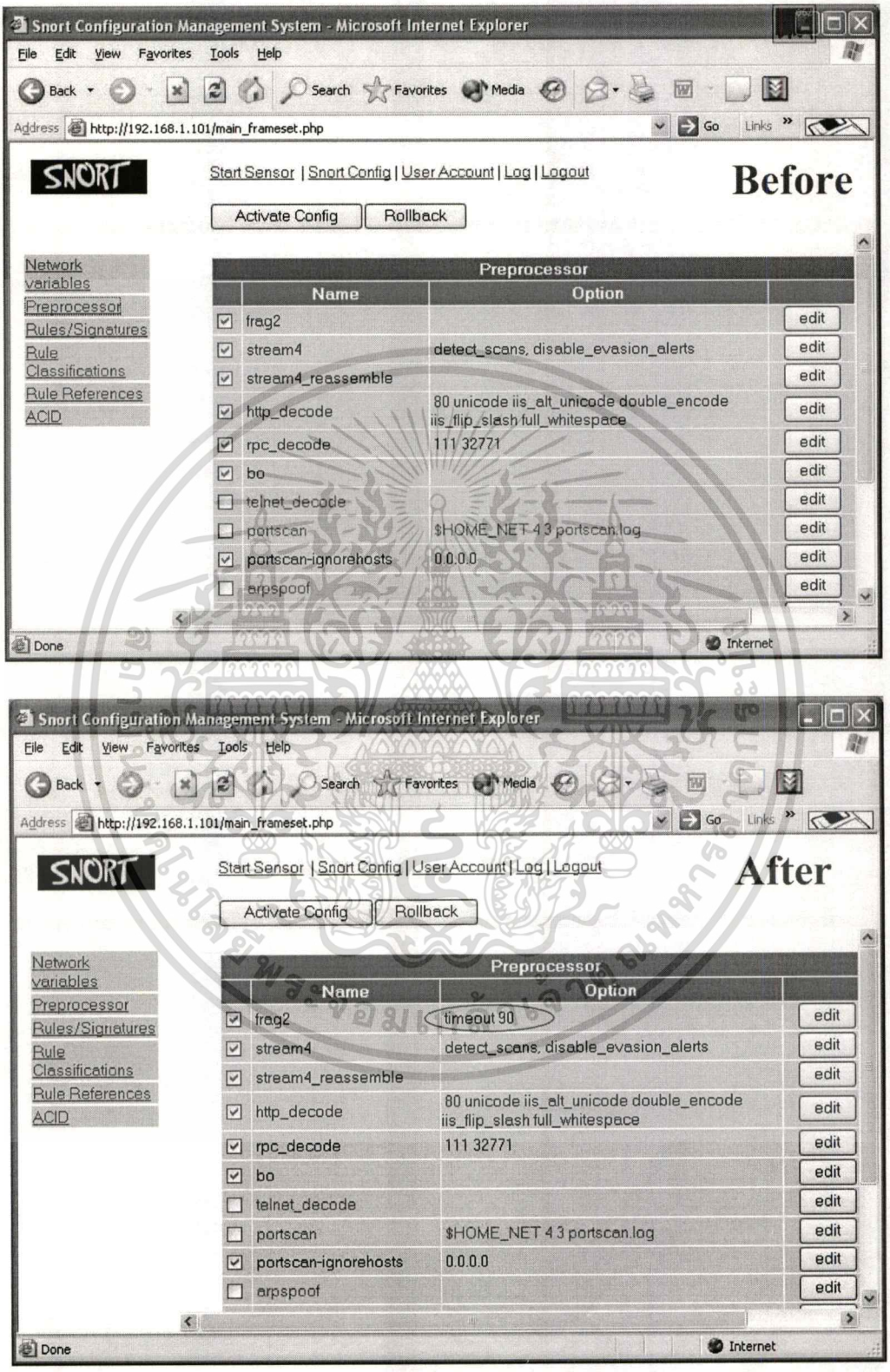
Before

```
# Frag2 uses Generator ID 113 and uses the following SIDS
# for that GID:
#  SID    Event description
# -----  -------------------
#   1     Oversized fragment (reassembled frag > 64k bytes)
#   2     Teardrop-type attack

preprocessor frag2

# stream4: stateful inspection/stream reassembly for Snort
#--------------------------------------------------------------------
# Use in concert with the -z [all|est] command line switch to defeat
# stick/snot against TCP rules.  Also performs full TCP stream
# reassembly, stateful inspection of TCP streams, etc.  Can statefully
# detect various portscan types, fingerprinting, ECN, etc.

# stateful inspection directive
# no arguments loads the defaults (timeout 30, memcap 8388608)
# options (options are comma delimited):
#   detect_scans - stream4 will detect stealth portscans and generate alerts
```

After

```
# Frag2 uses Generator ID 113 and uses the following SIDS
# for that GID:
#  SID    Event description
# -----  -------------------
#   1     Oversized fragment (reassembled frag > 64k bytes)
#   2     Teardrop-type attack

preprocessor frag2: timeout 90

# stream4: stateful inspection/stream reassembly for Snort
#--------------------------------------------------------------------
# Use in concert with the -z [all|est] command line switch to defeat
# stick/snot against TCP rules.  Also performs full TCP stream
# reassembly, stateful inspection of TCP streams, etc.  Can statefully
# detect various portscan types, fingerprinting, ECN, etc.

# stateful inspection directive
# no arguments loads the defaults (timeout 30, memcap 8388608)
# options (options are comma delimited):
#   detect_scans - stream4 will detect stealth portscans and generate alerts
```

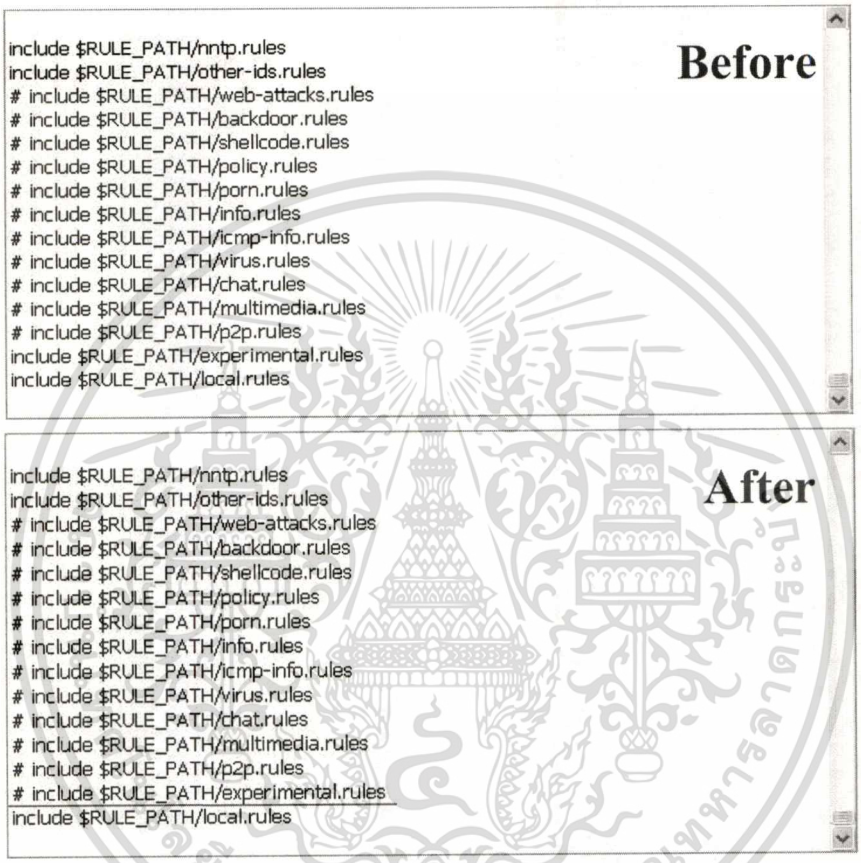
รูปที่ 5.4 เปรียบเทียบข้อมูลส่วนที่เกี่ยวข้องกับPreprocessor ใน snort.conf ก่อนและหลังการปรับเปลี่ยนค่า



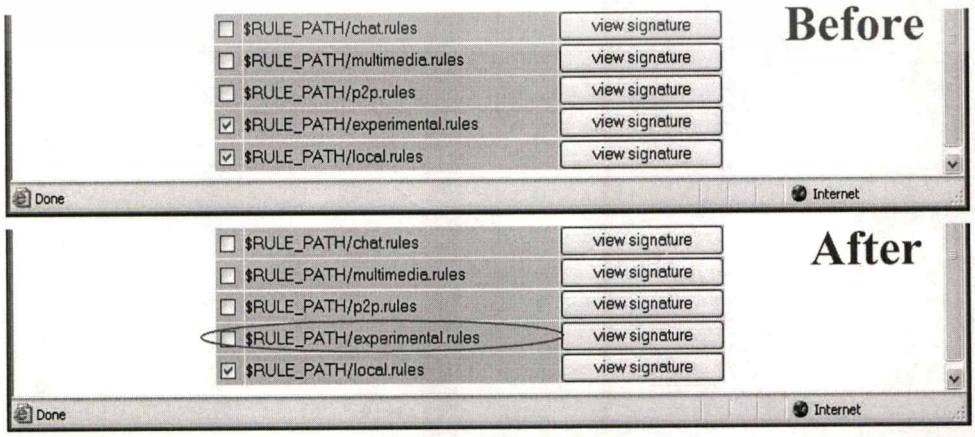
รูปที่ 5.5 เปรียบเทียบหน้าจอแสดงข้อมูลของ Preprocessor ก่อนและหลังการปรับเปลี่ยนค่า

การทดลองตอนที่ 3 การปรับแต่งกฎและกลุ่มของกฎ (Signature and Rule set)

การทดลองนี้จะไม่ใช้งานกลุ่มของกฎที่ชื่อว่า experimental (เอาเครื่องหมายถูกที่เช็คบ็อกซ์ออก) และเข้าไปเพิ่มกฎในกลุ่มกฎ local กฎที่เพิ่มเข้าไปเป็นดังนี้ alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg"Free Serial.com"; nocase; sid: 1000010; rev:1;)



รูปที่ 5.6 เปรียบเทียบข้อมูลส่วนที่เกี่ยวข้องกับกลุ่มของกฎใน snort.conf ก่อนและหลังการปรับเปลี่ยนสถานะ



รูปที่ 5.7 เปรียบเทียบหน้าจอแสดงกลุ่มของกฎที่ใช้งาน ก่อนและหลังการปรับเปลี่ยน

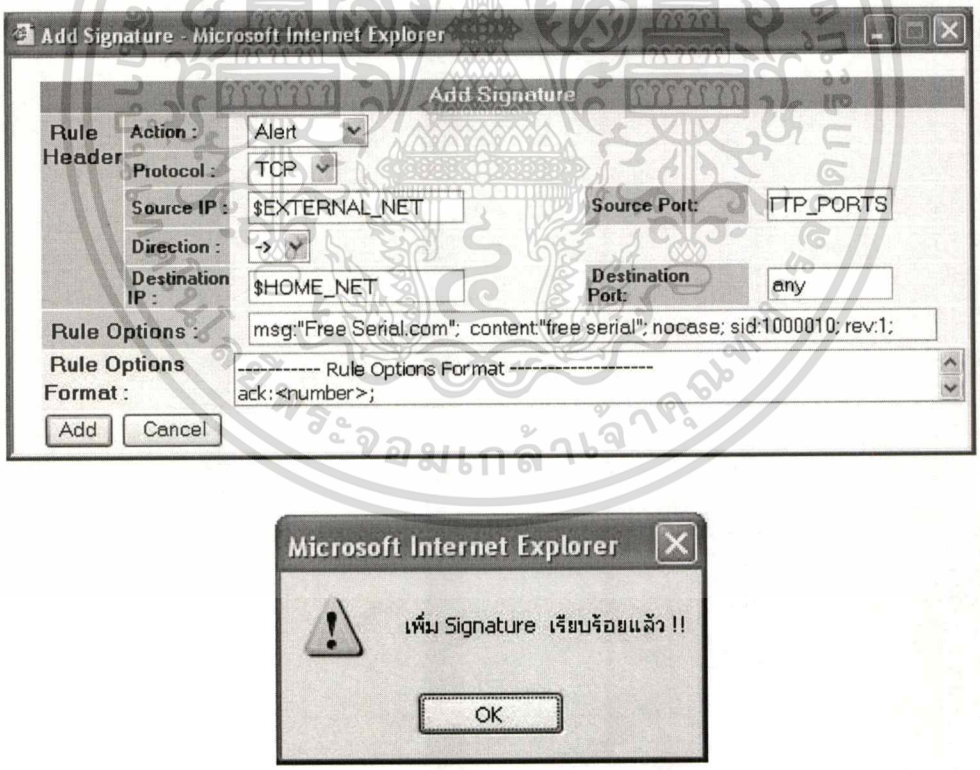
Before

```
# $Id: local.rules,v 1.5 2001/12/19 18:40:05 cazz Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
# test comment
```

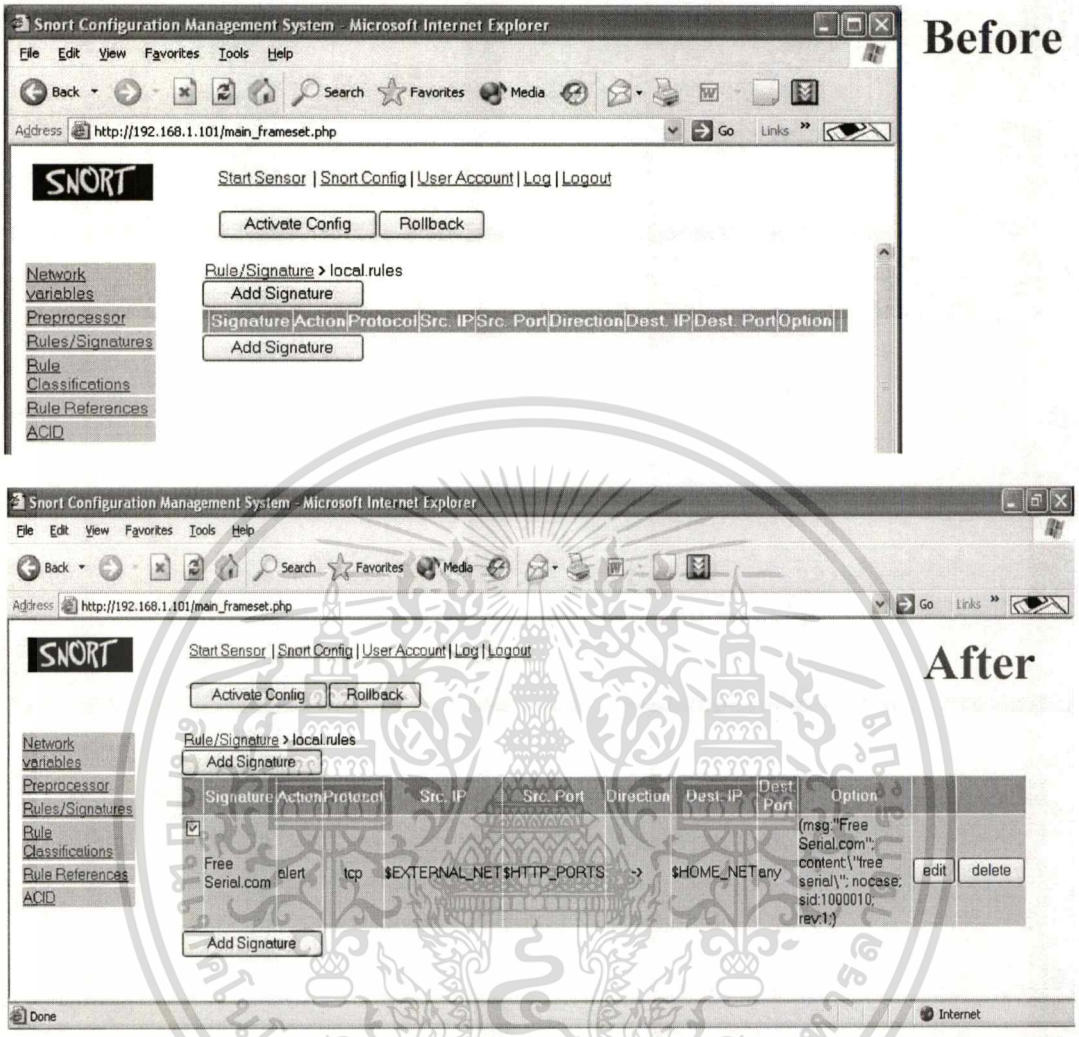
After

```
[root@sensor newvalue]# cat local.rules.new
# $Id: local.rules,v 1.5 2001/12/19 18:40:05 cazz Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
# test comment
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"Free Serial.com"; con
tent:"free serial"; nocase; sid:1000010; rev:1;)
[root@sensor newvalue]# _
```

รูปที่ 5.8 เปรียบเทียบข้อมูลในไฟล์ local.rules ก่อนและหลังเพิ่มกฎ



รูปที่ 5.9 หน้าจอสำหรับเพิ่มกฎเข้าไปในกลุ่มกฎ local (local.rules)



รูปที่ 5.10 เปรียบเทียบหน้าจอแสดงกฎในกลุ่ม local ก่อนและหลังการเพิ่มกฎ

การทดลองตอนที่ 4 การปรับแต่งประเภทของกฎ (Rule Classification)

การทดลองตอนนี้จะทำการเพิ่มประเภทของกฎเข้าไปโดยมีข้อมูลดังนี้ ชื่อประเภทคือ internal-attack คำอธิบายคือ Attack from internal organize ระดับความสำคัญเท่ากับ 1 หลังจากนั้นให้ทำการแก้ไขคำอธิบายกับระดับความสำคัญเป็น Attack from internal organization และ 2 ตามลำดับ

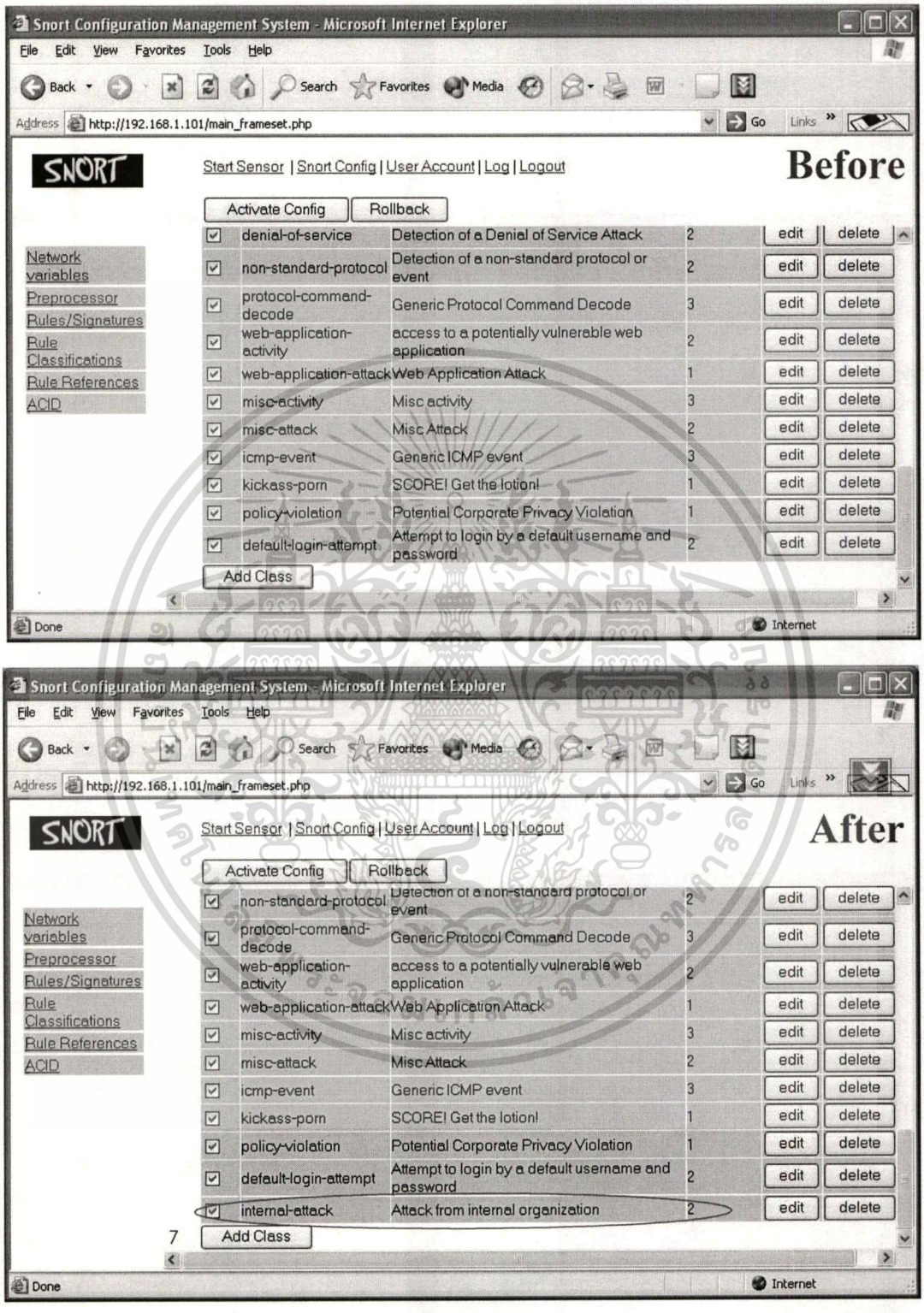
```
cted,2
config classification: suspicious-login,An attempted login using a suspicious us
ername was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unus
ual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack
,2
config classification: non-standard-protocol,Detection of a non-standard protoco
l or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerab
le web application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: kickass-porn,SCORE! Get the lotion!,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default usern
ame and password,2
[root@sensor root]# _
```

Before

```
config classification: suspicious-login,An attempted login using a suspicious us
ername was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unus
ual port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack
,2
config classification: non-standard-protocol,Detection of a non-standard protoco
l or event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerab
le web application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: kickass-porn,SCORE! Get the lotion!,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default usern
ame and password,2
config classification: internal-attack,Attack from internal organization,2
[root@sensor root]# _
```

After

รูปที่ 5.11 เปรียบเทียบข้อมูลในไฟล์ classification.config ก่อนและหลังปรับเปลี่ยนประเภทของกฎ



รูปที่ 5.12 เปรียบเทียบหน้าจอแสดงรายละเอียดประเภทของกฎ
ก่อนและหลังการปรับเปลี่ยนประเภทของกฎ

การทดลองตอนที่ 5 การปรับแต่งแหล่งที่มาของกฎ (Rule Reference)

จะทำการเพิ่มแหล่งอ้างอิงของกฎ ซึ่งมีชื่อระบบว่า FrSIRT และ URL คือ http://www.frsirt.com/english/advisories/2005/3080 เมื่อเพิ่มเข้าไปแล้วให้ทำการแก้ไขค่า URL เป็น http://www.frsirt.com/english/advisories/2005/3086 แล้วทำการเปิดเว็บเพจด้วย URL ใหม่

Before

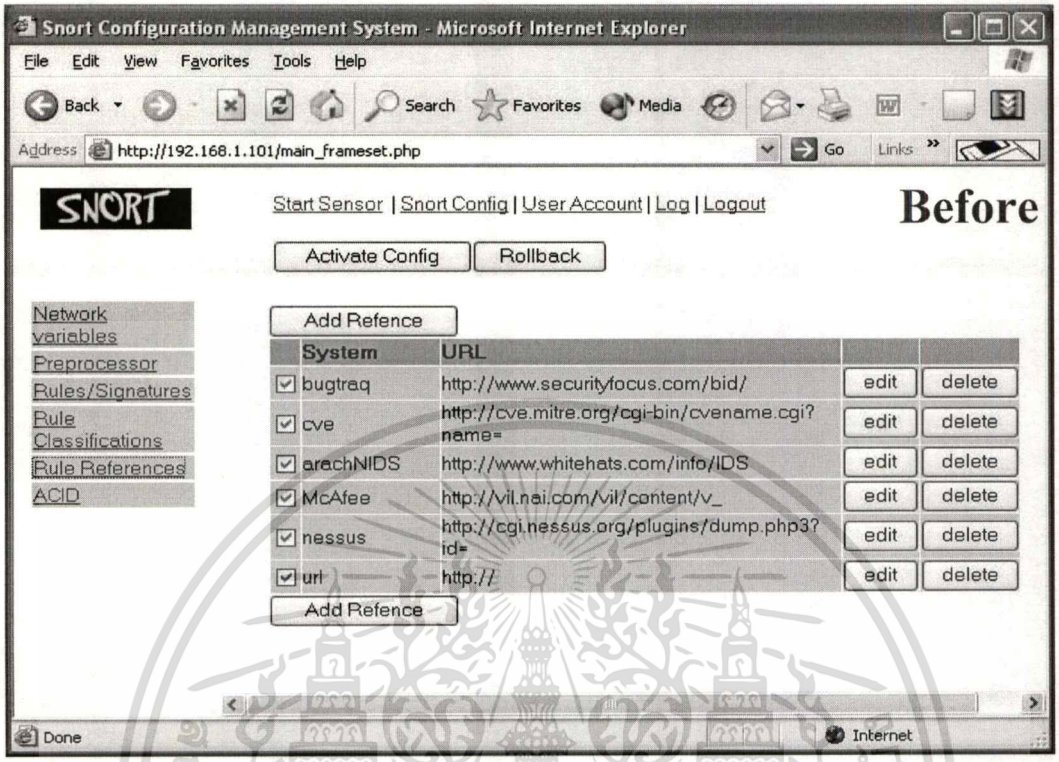
```
[root@sensor root]# cat /etc/snort/newvalue/reference.config.new
# $Id: reference.config,v 1.3 2002/08/28 14:19:15 chrisgreen Exp $
# The following defines URLs for the references found in the rules
#
# config reference: system URL

config reference: bugtraq    http://www.securityfocus.com/bid/
config reference: cve        http://cve.mitre.org/cgi-bin/cvename.cgi?name=
config reference: arachNIDS http://www.whitehats.com/info/IDS

# Note, this one needs a suffix as well.... lets add that in a bit.
config reference: McAfee     http://vil.nai.com/vil/content/v_
config reference: nessus     http://cgi.nessus.org/plugins/dump.php3?id=
config reference: url        http://
```

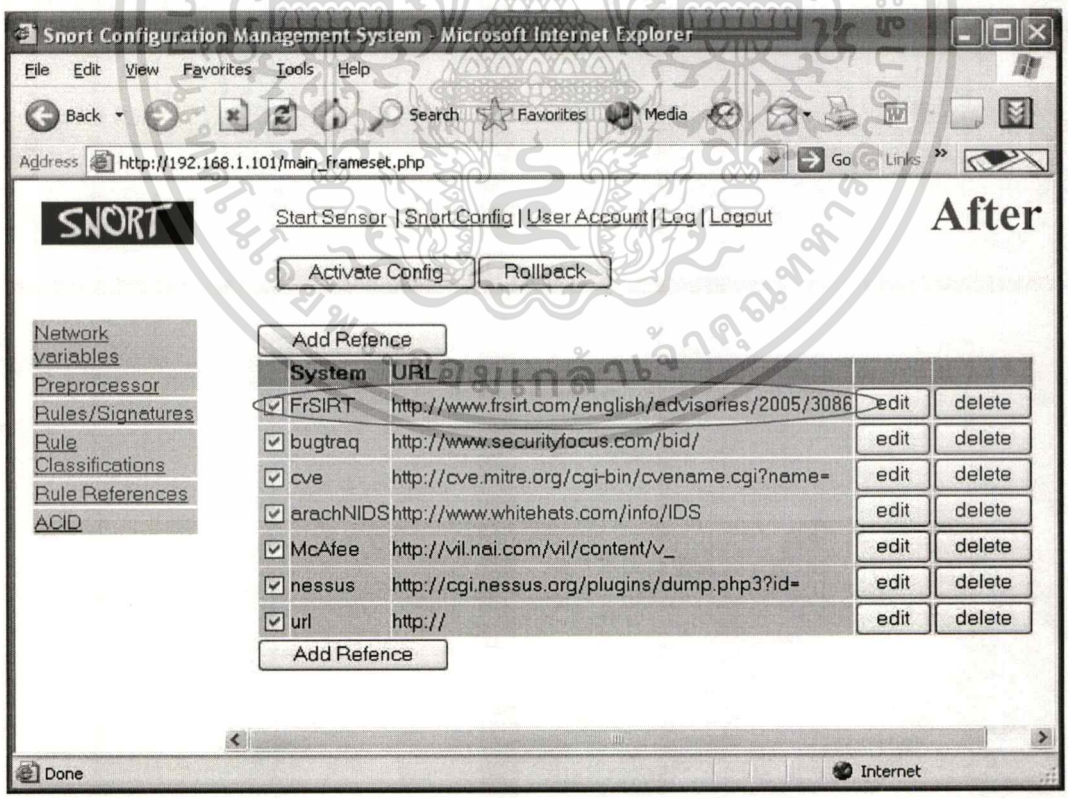
After

```
[root@sensor root]# cat /etc/snort/newvalue/reference.config.new
# $Id: reference.config,v 1.3 2002/08/28 14:19:15 chrisgreen Exp $
# The following defines URLs for the references found in the rules
#
# config reference: system URL

config reference: FrSIRT     http://www.frsirt.com/english/advisories/2005/3086
config reference: bugtraq    http://www.securityfocus.com/bid/
config reference: cve        http://cve.mitre.org/cgi-bin/cvename.cgi?name=
config reference: arachNIDS http://www.whitehats.com/info/IDS

# Note, this one needs a suffix as well.... lets add that in a bit.
config reference: McAfee     http://vil.nai.com/vil/content/v_
config reference: nessus     http://cgi.nessus.org/plugins/dump.php3?id=
config reference: url        http://
```

รูปที่ 5.13 เปรียบเทียบข้อมูลในไฟล์ reference.config ก่อนและหลังปรับเปลี่ยนแหล่งที่มาของกฎ

รูปที่ 5.14 เว็บไซต์แหล่งที่มาของกฎที่เพิ่มเข้าไปในระบบ

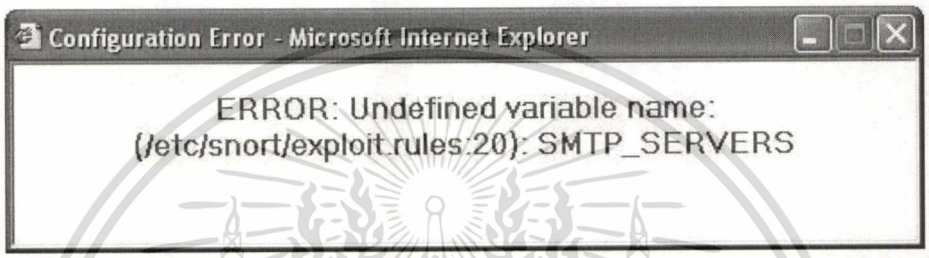




รูปที่ 5.15 เปรียบเทียบหน้าจอแสดงรายละเอียดแหล่งที่มาของกฎ ก่อนและหลังการปรับเปลี่ยน

การทดลองตอนที่ 6 การปรับแต่งค่าคอนฟิกูเรชันที่ทำให้เกิดความผิดพลาดต่อการทำงานของ Snort

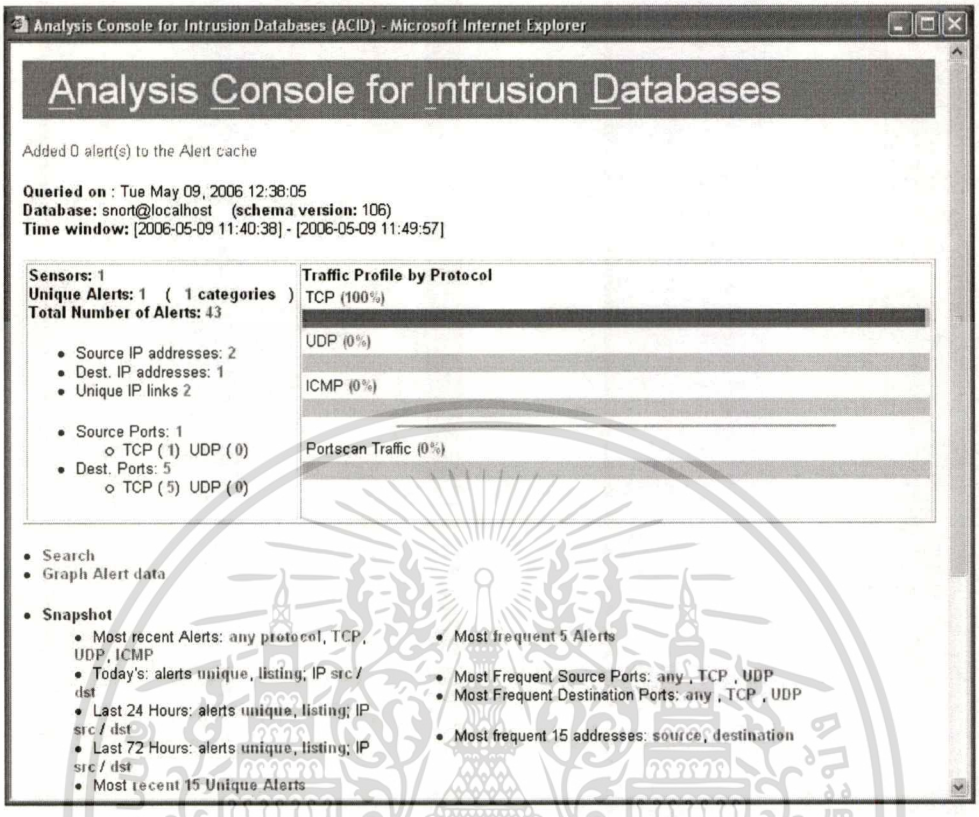
การทดลองตอนนี้จะอาศัยค่าต่างๆที่เราได้ทำการเพิ่มเติมหรือปรับแต่งไว้ก่อนหน้านี้ โดยเฉพาะตัวแปรเครือข่าย SMTP_SERVERS ที่เราได้ปิดการใช้งานไปนั้น เริ่มต้นด้วยการ Activate Configuration เพื่อนำค่าที่ได้แก้ไขเอาไว้ในตอนที่ 1 ถึง 5 ไปใช้งานจริง แล้วให้ทำการเปิดใช้งานเซ็นเซอร์ ผลคือเราจะไม่สามารถเปิดใช้งานเซ็นเซอร์ได้ และจะมีข้อความแจ้งเตือนข้อผิดพลาดว่ากฎในกลุ่มกฎ exploit มีการใช้ตัวแปรเครือที่ไม่ได้นิยามไว้ซึ่งก็คือ SMTP_SERVERS นั่นเอง



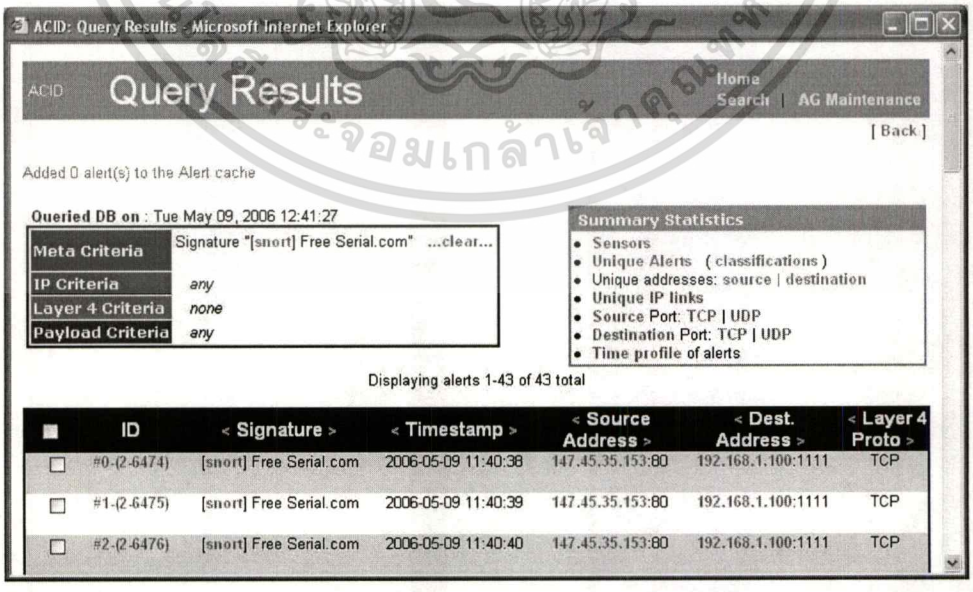
รูปที่ 5.16 หน้าจอแจ้งเตือนข้อความผิดพลาดจากการไม่ใช้งานตัวแปรเครือข่าย SMTP_SERVERS

การทดลองตอนที่ 7 การแจ้งเตือนของเซ็นเซอร์ต่อการบุกรุกที่ตรงกับกฎที่ใช้งาน

การทดลองตอนนี้จะอาศัยกฎที่ได้เพิ่มเข้าไปในกลุ่มกฎ local (ทดลองแล้วในตอนที่ 3) ในการทดสอบ เริ่มต้นให้เลือกใช้งานตัวแปรเครือข่าย SMTP_SERVERS เพื่อจะได้เปิดใช้งานเซ็นเซอร์ได้ เพราะในการทดลองตอนที่ 6 เกิดข้อผิดพลาดจากส่วนนี้ จากนั้นทำการ Activate Configuration เพื่อนำค่าที่แก้ไขไปให้งาน แล้วสั่งเปิดใช้งานเซ็นเซอร์ เมื่อเซ็นเซอร์ทำงานแล้ว ให้เปิดเว็บน์เซอร์ขึ้นมาอีกหน้าต่าง พิมพ์ URL ดังนี้ http://freeserials.com เมื่อเว็บเพจปรากฏเรียบร้อยแล้วให้กลับไปยังเว็บบราวเซอร์ที่เปิดระบบการปรับตั้งค่าและกฎไว้ ตรวจสอบการแจ้งเตือนที่เมนู ACID ก็จะได้ผลการแจ้งเตือนดังรูปที่ 5.17 - 5.18



รูปที่ 5.17 หน้าจอหลักสำหรับแสดงการตรวจจับที่ตรงกับกฎที่ตั้งไว้



รูปที่ 5.18 หน้าจอแสดงจำนวนการแจ้งเตือนทั้งหมดซึ่งตรงกับกฎที่ตั้งไว้

บทที่ 6
สรุปผลการออกแบบและพัฒนาระบบ

6.1 สรุปผลโครงการพัฒนาระบบ

จากการทดสอบระบบในบทที่ 5 แสดงให้เห็นว่าระบบสามารถปรับแต่งค่าคอนฟิกูเรชั่นและกฎบนเครื่อง Snort Sensor ได้จริง โดยการทดลองตอนที่ 1 ถึง 5 จะแสดงให้เห็นว่าระบบเข้าไปแก้ไขค่าในไฟล์ที่เกี่ยวข้องแต่ไฟล์เหล่านั้นยังไม่ถูกนำมาใช้งานจนกว่าผู้ใช้จะทำการ Activate Configuration เสียก่อน หลังจากทำการ Activate Configuration ผู้ใช้ก็ต้องเปิดการทำงานของ Snort Sensor ซึ่งในขั้นตอนนี้ถ้าค่าที่ปรับแต่งไม่สอดคล้องหรือถูกต้อง ระบบก็จะแจ้งข้อผิดพลาดและสาเหตุที่น่าจะเป็นให้ทราบ ดังผลในการทดลองตอนที่ 6 แต่ถ้าไม่มีข้อผิดพลาดใดๆเกิดขึ้นเครื่อง Snort Sensorก็จะทำการตรวจจับการบุกรุกตามกฎที่ได้ตั้งไว้ ดังผลการทดลองตอนที่ 7 สำหรับกรณีที่เกิดความผิดพลาดในการปรับแต่งค่าหรือกฎ ผู้ใช้สามารถเรียกค่าเก่ากลับมาได้ด้วยการทำ Rollback

6.2 ปัญหาและข้อจำกัดของระบบ

- ใช้ได้กับระบบเครือข่ายขนาดเล็กเท่านั้นเนื่องจากปรับแต่งค่าและกฎบน Snort Sensor ได้เพียงเครื่องเดียวเท่านั้น
- ถึงแม้ว่าจะมีรูปแบบและตัวอย่างของค่า ที่ผู้ใช้สามารถกรอกลงในระบบให้ดู แต่การเปิดหรือค้นหาก็ยังไม่สะดวกนัก (ผู้ใช้ต้องเลื่อนหาด้วยตนเอง)
- การกู้ข้อมูลทำได้ในระดับเบื้องต้น กล่าวคือจะกู้ได้เฉพาะค่าเก่าล่าสุดก่อนทำการ Activate Config เท่านั้น และเนื่องจากใช้ไฟล์ในการสำรองข้อมูลจึงไม่มีกลไกในการจัดการเมื่อเกิดปัญหาจากแหล่งจ่ายไฟฟ้าหรือฮาร์ดแวร์
- ระบบไม่มีการปรับปรุงกฎที่ออกโดย snort.org แบบอัตโนมัติ ผู้ใช้งานต้องกระทำด้วยตนเอง

บรรณานุกรม

กิตติศักดิ์ เจริญโภคานนท์. 2545. **คัมภีร์การสร้าง E-Commerce Application PHP4**. กรุงเทพฯ: ซัคเซส มีเดีย จำกัด.

บัณฑิต จามรภูติ. 2547. **คัมภีร์ REDHAT ENTERPRISE เล่ม 2**. กรุงเทพฯ: Bandhit press.

พันจันทร์ ธนวัฒนเสถียร และคณะ. 2546. **Macromedia DREAMWEAVER MX ฉบับเรียนลัด**. กรุงเทพฯ: ซัคเซส มีเดีย จำกัด.

Caswell, Brain. 2002. **SNORT 2.0 Intrusion Detection Systems**. Rockland: Syngress Pub.

Crothers, Tim. 2002. **Implementing Intrusion Detection System : A Hand-On Guide for Securing the Network**. Indiana: Wiley Pub.

Harper, Patrick. 2003. **Snort, Apache, PHP, MySQL, ACID on Redhat 9.0 Installation Guide**. [Online]. Available: http://www.snort.org/docs/snort_acid_rh9.pdf.

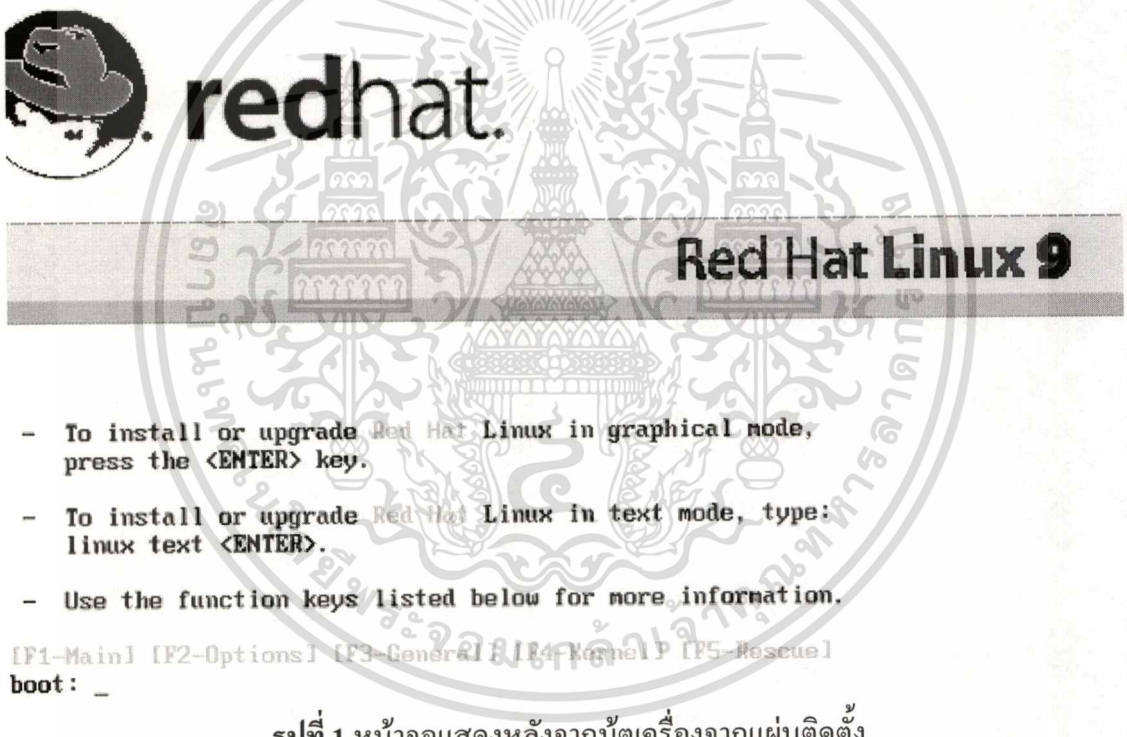
Orebaugh, Angela. et al. 2005. **Snort Cookbook**. California: O'Reilly.

Scott, Steven. 2003. **Snort Enterprise Implementation Snort, MySQL, SnortCenter and ACID on Redhat 9.0**. [Online]. Available: http://www.superhac.com/docs/snort_enterprise.pdf.

ภาคผนวก ก.
การติดตั้งระบบปฏิบัติการ

1. การติดตั้งระบบปฏิบัติการ Linux Redhat 9.0

ในการติดตั้งระบบปฏิบัติการนั้นเราจะติดตั้งเฉพาะแพ็กเกจที่จำเป็นเท่านั้นเพื่อให้เครื่อง Snort Sensor มีคล่องตัวและมีความปลอดภัยในการทำงานมากขึ้น โดยเริ่มแรกปรับแต่งค่าคอนฟิกในไบออสให้เครื่องคอมพิวเตอร์สามารถบู้ตจากแผ่นซีดี จากนั้นใส่แผ่นติดตั้งลงในซีดีรอม หน้าจอสำหรับการติดตั้งระบบปฏิบัติการจะแสดงดังรูปที่ 1



รูปที่ 1 หน้าจอแสดงหลังจากบู้ตเครื่องจากแผ่นติดตั้ง

ให้ทำการกดปุ่ม Enter เพื่อติดตั้งในโหมดกราฟฟิก

หน้าจอ Welcome ให้คลิ้ก next

หน้าจอสำหรับเลือก Language ให้เลือกเป็น English

หน้าจอสำหรับเลือก Keyboard ให้เลือกเป็น U.S. English

หน้าจอสำหรับเลือก Mouse ให้เลือกตามชนิดของ mouse ที่คุณใช้

หน้าจอสำหรับเลือกชนิดของการติดตั้ง (Install Type) ให้คลิ้กที่ custom

หน้าจอสำหรับแบ่งพาร์ติชั่น (Disk Partitioning)

- ให้เลือก Automatically partition the hard drive
- เลือก Remove all partitions from this hard drive (อยู่บนสมมติฐานที่ว่าเครื่องคอมพิวเตอร์ไม่ได้ลงแบบ Dual boot)
- ต้องมีเครื่องหมายถูกตรง checkbox ของ review button
- กำหนดให้ SWAP มีขนาดเป็น 2 เท่าของหน่วยความจำ (RAM)
- กำหนดให้ /boot มีขนาดเป็น 100 MB
- พื้นที่ที่เหลือทั้งหมดเป็นของ /

หน้าจอกำหนด Boot Loader ให้เลือกตาม default

หน้าจอสำหรับกำหนดค่าเกี่ยวกับเครือข่าย

- กดปุ่ม Edit
- ที่ checkbox หน้า Configure with DHCPให้เอาเครื่องหมายถูกออก
- กำหนดหมายเลขไอพี และ ซับเน็ตมาร์ค
- กำหนดชื่อเครื่อง
- กำหนดหมายเลขไอพีของ DNS และ Gateway

หน้าจอสำหรับกำหนด Firewall ให้เลือก no firewall

หน้าจอสำหรับเลือกภาษาเพิ่มเติม (Additional Language) ให้เลือก US English

หน้าจอสำหรับตั้งเวลา ให้เลือกตามประเทศที่คุณอยู่

หน้าจอสำหรับกำหนดรหัสผ่าน (Root Password) ก็กำหนดตามใจชอบแต่ควรเป็นคำที่คาดเดาได้ยากและมีความยาวไม่ต่ำกว่า 8 ตัวอักษร

หน้าจอสำหรับกำหนด Authentication นั้นให้เลือกทั้ง MD5 และ Shadow Password

หน้าจอสำหรับเลือก Packages

Desktop:

X Window System ให้คลิ้กที่ details และเอาเครื่องหมายถูกออกสำหรับหัวข้อดังนี้

- xisdnload

Gnome Desktop Environment เลือก (default)

KDE Desktop Environment ไม่เลือก (default)

Application:

Editors ไม่เลือก

Engineering and Scientific ไม่เลือก (default)

Graphical Internet ทำการเลือก และคลิ้กที่ details เพื่อเลือกติดตั้งเฉพาะโปรแกรมต่อไปนี้

- evolution ในกรณีที่ต้องการอ่าน e-mail ด้วยโปรแกรม Outlook
- Mozilla
- Mozilla-psm

Text based internet ทำการเลือก และคลิ้กที่ details เพื่อเลือกติดตั้งเฉพาะโปรแกรมต่อไปนี้

- Lynx เป็นโปรแกรมเว็บบราวน์เซอร์
- Pine เป็น e-mail client

Office/Productivity ให้เลือกแค่ xpdf เท่านั้น

Sound and Video ไม่จำเป็นต้องเลือก

Authoring and Publishing ไม่เลือก

Graphics เลือกเฉพาะโปรแกรมต่อไปนี้

- Gimp
- Gimp data extras
- Gimp print plugin

Games and Entertainment ไม่เลือก

Server Section: เลือก ftp server

Development:

Development tools ทำการเลือก และคลิ้กที่ details แล้วทำการเลือก Expect กับ Gcc-objc เพิ่มจากค่า default

Kernel development ทำการเลือกและใช้ตามค่า default

System:

Administration ไม่เลือก

System Tools เลือกเฉพาะโปรแกรมต่อไปนี้

- Ethereal
- Ethereal gnome
- Nmap
- Nmap frontend

Printing support ไม่เลือก

Miscellaneous: ไม่เลือกอะไรเลยในหัวข้อนี้

หลังจากติดตั้ง Package เสร็จแล้วจะมีหน้าจอสำหรับกำหนดค่าอีกเล็กน้อยดังนี้

หน้าจอสำหรับสร้าง Boot Disk ให้เลือก no

หน้าจอสำหรับกำหนดค่าให้กับ X Window ก็ให้ทำการเลือกรุ่นของการ์ดแสดงผล ชนิดของจอคอมพิวเตอร์ ความละเอียดและขนาดของหน้าจอ ตามอุปกรณ์ที่มี ท้ายสุดคือเลือกชนิดของการ Login (Login Type) เป็น Text

เมื่อกำหนดค่าต่างๆเรียบร้อยแล้วระบบจะทำการ Reboot เมื่อระบบ reboot เรียบร้อยแล้วให้ login ด้วย Root แล้วสร้าง user ที่ชื่อว่า package ขึ้นมาด้วยคำสั่งดังนี้

```
[root@sensor root]# useradd package
[root@sensor root]# passwd package
                    รหัสที่ต้องการ
                    ยืนยันรหัสที่ต้องการ
```

เนื่องจากโปรแกรมอื่นๆที่จำเป็นต่อการติดตั้ง Snort Sensor อันได้แก่ Zlib Libpcap MySQL Apache PHP Snort JPGraph ADODB และ ACID จะอยู่คนละเครื่องเราจึงต้องใช้ ftp client ส่งมาให้กับเครื่อง Snort Sensor ดังนั้นเครื่อง Snort จึงต้องทำการเปิดบริการ ftp ด้วยคำสั่งต่อไปนี้

```
[root@sensor root]# services vsftpd start
```

หลังจากนั้นทำการส่งโปรแกรมที่จำเป็นทั้งหมดมายังเครื่อง Snort Sensor ซึ่งโปรแกรมเหล่านั้นจะถูกเก็บไว้ที่ /home/package ให้เข้าไปยังไดเรกทอรี่นั้นแล้วทำการติดตั้งโปรแกรมทั้งหมดต่อไป

2. การติดตั้งโปรแกรมอื่นๆเพื่อทำให้เครื่องคอมพิวเตอร์สามารถตรวจจับการบุกรุก

การติดตั้ง Zlib มีวิธีการดังนี้

```
[root@sensor package]# tar –xvfz  zlib-1.1.4.tar.gz
[root@sensor package]# cd zlib-1.1.4
[root@sensor zlib-1.1.4]# ./configure; make test
[root@sensor zlib-1.1.4]# make install
[root@sensor zlib-1.1.4]# cd ..
```

การติดตั้ง LibPcap มีวิธีการดังนี้

```
[root@sensor package]# tar –xvzf  libpcap-0.7.2.tar.gz
[root@sensor package]# cd libpcap-0.7.2
[root@sensor libpcap-0.7.2]# ./configure
[root@sensor libpcap-0.7.2]# make
[root@sensor libpcap-0.7.2]# make install
[root@sensor libpcap-0.7.2]# cd /root
```

จากนั้นทำการสร้าง User และ Group ให้กับ mysql ดังนี้

```
[root@sensor root]# groupadd  mysql
[root@sensor root]# useradd –g  mysql  mysql
[root@sensor root]# vi  .bash_profile
```

ทำการเพิ่ม Path ให้กับ MySQL โดยเพิ่มข้อความ :/usr/local/mysql/bin ต่อท้ายบรรทัดที่มีข้อความดังนี้ PATH=$PATH:$HOME/bin ในไฟล์ที่ชื่อว่า .bash_profile แล้วจึงทำการติดตั้ง MySQL

```
[root@sensor root]# cd /home/package
[root@sensor package]# tar –xvzf  mysql-4.0.16.tar.gz
[root@sensor package]# cd mysql-4.0.16
[root@sensor mysql-4.0.16]# ./configure --prefix=/usr/local/mysql
[root@sensor mysql-4.0.16]# make
[root@sensor mysql-4.0.16]# make install
[root@sensor mysql-4.0.16]# scripts/mysql_install_db
[root@sensor mysql-4.0.16]# chown –R root  /usr/local/mysql
[root@sensor mysql-4.0.16]# chown –R mysql  /usr/local/mysql/var
[root@sensor mysql-4.0.16]# chgrp –R mysql  /usr/local/mysql
[root@sensor mysql-4.0.16]# cp support-files/my-medium.cnf  /etc/my.cnf
```

ทำการเพิ่มบรรทัด user = mysql ลงในไฟล์ /etc/my.cnf ตรงส่วนที่เป็น [mysqld] จากนั้นเพิ่มบรรทัด /usr/local/mysql/lib/mysql และ /usr/local/lib ลงในไฟล์ /etc/ld.so.conf

```
[root@sensor mysql-4.0.16]# vi /etc/my.cnf
[root@sensor mysql-4.0.16]# vi /etc/ld.so.conf
[root@sensor mysql-4.0.16]# ldconfig -v
```

ถัดมาจะเป็นการทดสอบการทำงานของ MySQL ด้วยคำสั่งต่อไปนี้

```
[root@sensor mysql-4.0.16]# /usr/local/mysql/bin/mysqld_safe --user=mysql &
                        --- กด Enter 1 ครั้ง ---
[root@sensor mysql-4.0.16]# ps –ef | grep mysql
```

ถ้าไม่มีข้อผิดพลาดจะได้ผลลัพธ์คล้ายกับข้อความต่อไปนี้

root 2099 2053 0 12:00 tty1 00:00:00:00 /bin/sh /usr/local/mysql/bin/mysqld_safe --user=mysql

mysql 2121 2099 0 12:00 tty1 00:00:00:00 /usr/local/mysql/libexec/mysqld --basedir=/usr/local/mysql --datadir=/usr/local/mysql/var --user=mysql --pid-file= /usr/local /mysql/var/IDS.pid --skip-locking

หลังจากทดสอบผ่านแล้วจะตั้งค่าให้ MySQL เริ่มทำงานโดยอัตโนมัติโดยมีขั้นตอนดังนี้

```
[root@sensor mysql-4.0.16]# cd support-files
[root@sensor support-files]# cp mysql.server  /etc/init.d/mysql
[root@sensor support-files]# cd /etc/rc3.d
[root@sensor rc3.d]# ln –s ../init.d/mysql S85mysql
[root@sensor rc3.d]# ln –s ../init.d/mysql K85mysql
[root@sensor rc3.d]# cd /etc/rc5.d
[root@sensor rc5.d]# ln –s ../init.d/mysql S85mysql
[root@sensor rc5.d]# ln –s ../init.d/mysql K85mysql
[root@sensor rc5.d]# cd ../init.d
[root@sensor init.d]# chmod 755 mysql
```

ลำดับต่อไปคือการติดตั้งโปรแกรมเว็บเซิร์ฟเวอร์ที่มีชื่อว่า Apache บริการของเว็บเซิร์ฟเวอร์บนระบบปฏิบัติการลีนุกซ์นั้นจะเรียกว่า httpd ซึ่งย่อมาจากคำว่า HTTP Daemon ในการติดตั้งซึ่งมีวิธีการติดตั้งดังนี้

```
[root@sensor package]# tar -xvfz apache_1.3.33.tar.gz
[root@sensor package]# cd apache_1.3.33
[root@sensor apache_1.3.33]# env CFLAGS="-DBIG_SECURITY_HOLE" ./configure --
prefix=/www --enable-module=so (ทำการ compile และ config ให้รันเป็น user root ได้)
[root@sensor apache_1.3.33]# make
[root@sensor apache_1.3.33]# make install
```

ในโครงการนี้เราทำการติดตั้ง Apache ไว้ในไดเร็กทอรี่ /www เมื่อติดตั้งเสร็จเรียบร้อยแล้วให้แก้ไขค่าคอนฟิกใน /www/conf/httpd.conf ดังนี้เพื่อที่จะให้ Apache รันเป็น root ใส่ชื่อเครื่องเซิร์ฟเวอร์เป็นชื่อเครื่องที่ตั้งไว้ และ กำหนด Path ที่เก็บเอกสารที่จะแสดงบนเว็บไว้ที่ ftp user folder

```
User root
Group root

ServerName sensor
DocumentRoot "/home/package"
```

เมื่อติดตั้งเสร็จเรียบร้อยแล้ว ให้ทำการรันเซอร์วิสก่อนเว็บเซิร์ฟเวอร์จึงจะทำงานได้

```
[root@router root]# /www/bin/apachectl start
Starting Apache:                    [OK]
```

การจะติดตั้ง PHP ให้ทำงานได้นั้นบนระบบปฏิบัติการลีนุกซ์ติดตั้งบริการเว็บเซิร์ฟเวอร์ก่อนจึงจะสามารถทำการติดตั้ง PHP ลงไปได้ ซึ่งการติดตั้งนั้นสามารถดาวน์โหลดไฟล์จากอินเทอร์เน็ตได้ ซึ่งสามารถติดตั้งโดยใช้คำสั่งดังนี้

```
[root@sensor package]# tar -xvfz php-4.3.4.tar.gz
[root@sensor package]# cd php-4.3.4/
[root@sensor php-4.3.4]# ./configure --prefix=/www/php --with-apxs=/www/bin/apxs --with-
config-file-path=/www/php --enable-sockets --with-mysql=/usr/local/mysql --with-zlib-
dir=/usr/local --with-gd
[root@sensor php-4.3.4]# make
[root@sensor php-4.3.4]# make install
```

```
[root@sensor php-4.3.4]# cp php.ini-dist  /www/php/php.ini
```

ซึ่งเมื่อทำการติดตั้งเสร็จจะต้องเพิ่มค่าในไฟล์ httpd.conf ด้วย (อยู่ใน /www/conf)

```
#LoadModule foo_module  libexec/mod_foo.so
LoadModule php4_module  libexec/libphp4.so

#AddType allows you to tweak mime.types without actually editing it, or $
# make certain files to be certain types.
AddType application/x-tar  .tgz
AddType image/x-icon  .ico
AddType application/x-httpd-php  .php

#DirectoryIndex: Name of the file or files to use as a pre-written HTML
#directory index. Separate multiple entries with spaces.
<IfModule mod_dir.c
    DirectoryIndex index.php index.html index.html.var
```

จากนั้นตั้งค่าให้ Apache เริ่มทำงานโดยอัตโนมัติโดยมีขั้นตอนดังนี้

```
[root@sensor php-4.3.4]# cd /www/bin
[root@sensor bin]# cp apachectl  /etc/init.d/httpd
[root@sensor bin]# cd /etc/rc3.d
[root@sensor rc3.d]# ln –s ../init.d/httpd S85httpd
[root@sensor rc3.d]# ln –s ../init.d/httpd K85httpd
[root@sensor rc3.d]# cd /etc/rc5.d
[root@sensor rc5.d]# ln –s ../init.d/httpd S85httpd
[root@sensor rc5.d]# ln –s ../init.d/httpd K85httpd
```

เมื่อทำการติดตั้งเสร็จเรียบร้อยแล้วสามารถตรวจสอบการทำงานได้ โดยทำการรีสตาร์ต Apache อีกครั้ง แล้วนำเว็บเพจที่เขียนด้วยภาษา PHP นำไปไว้ในไดเร็กทอรี่ /home/package จากนั้นลองทำการเรียกจากบราวเซอร์ดู

```
[root@sensor rc5.d]# /etc/rc3.d/S85httpd  restart
```

การติดตั้ง Snort ทำได้ดังนี้

```
[root@sensor rc5.d]# cd  /home/package
[root@sensor package]# groupadd snort
[root@sensor package]# useradd –g snort snort
[root@sensor package]# mkdir  /etc/snort
[root@sensor package]# mkdir  /var/log/snort
[root@sensor package]# tar –xvzf  snort-2.0.4.tar.gz
[root@sensor package]# cd  snort-2.0.4
[root@sensor snort-2.0.4]# ./configure --with-mysql=/usr/local/mysql
[root@sensor snort-2.0.4]# make
[root@sensor snort-2.0.4]# make install
[root@sensor snort-2.0.4]# cd  rules
[root@sensor rules]# cp *  /etc/snort
[root@sensor rules]# cd  ../etc
[root@sensor etc]# cp snort.conf  /etc/snort
[root@sensor etc]# cp *.config  /etc/snort
```

หลังจากนั้นให้เข้าไปแก้ไข snort.conf ที่อยู่ใน /etc/snort ที่บรรทัด var HOME_NET ให้ใส่หมายเลขเครือข่ายที่ติดตั้ง snort ที่บรรทัด var RULE_PATH ให้ใส่เป็น /etc/snort และที่บรรทัด output database: ให้ใส่ log, mysql, user=snort password=รหัสที่ต้องการ dbname=snort host=localhost

```
var HOME_NET หมายเลขเครือข่ายขององค์กร

var RULE_PATH /etc/snort

output database: log, mysql, user=snort password=itsnort  dbname=snort host=localhost
```

จากนั้นเราจะทำการคัดลอกสคริปต์ไฟล์ที่ใช้รัน Snort ไปไว้ใน Path ที่สามารถทำการสั่งงานสคริปต์นั้นได้

```
[root@sensor etc]# cd ..
[root@sensor snort-2.0.4]# cp contrib/S99snort  /etc/init.d/snort
[root@sensor snort-2.0.4]# cd /etc/init.d
[root@sensor init.d]# chmod 755 snort
```

หลังจากนั้นให้เข้าไปแก้ไขไฟล์ /etc/init.d/snort โดยเปลี่ยนบรรทัดที่ขึ้นต้นด้วย CONFIG และ SNORT_GID ดังนี้

```
CONFIG=/etc/snort/snort.conf
SNORT_GID=snort
```

เมื่อทำเสร็จแล้วเราจะสามารถสั่งให้ Snort ทำงานในโหมดตรวจจับการบุกรุก (IDS) ด้วยคำสั่ง /etc/init.d/snort start หรือสั่งให้ Snort หยุดทำงานด้วยคำสั่ง /etc/init.d/snort stop}

ลำดับต่อมาจะเป็นการสร้างฐานข้อมูลใน MySQL

```
[root@sensor root]# /usr/local/mysql/bin/mysql
mysql > SET PASSWORD FOR root@localhost=PASSWORD('itroot');
mysql > create database snort;
mysql > grant INSERT, SELECT on root.* to snort@localhost;
mysql > SET PASSWORD FOR snort@localhost=PASSWORD('itsnort');
mysql > grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to
snort@localhost;
mysql > grant CREATE, INSERT, SELECT, DELETE, UPDATE on snort.* to snort
mysql > exit
[root@sensor root]# cd /home/package/snort-2.0.4
[root@sensor snort-2.0.4]# /usr/local/mysql/bin/mysql –u root –p < ./contrib./create_mysql
snort
[root@sensor snort-2.0.4]# cd contrib
[root@sensor contrib]# zcat snortdb-extra.gz | /usr/local/mysql/bin/mysql –p snort
```

```
[root@sensor contrib]# /usr/local/mysql/bin/mysql –p
mysql > SHOW DATABASES;
mysql > use snort
mysql > SHOW TABLES;
mysql > exit
[root@sensor contrib]# cd /home/package
```

ทำการติดตั้ง JPGraph

```
[root@sensor package]# tar –xvzf jpgraph-1.13.tar.gz
[root@sensor package]# cd  jpgraph-1.13
[root@sensor jpgraph-1.13]# rm –rf README
[root@sensor jpgraph-1.13]# rm –rf QPL.txt
[root@sensor jpgraph-1.13]# cd ..
```

ทำการติดตั้ง ADODB และ ACID

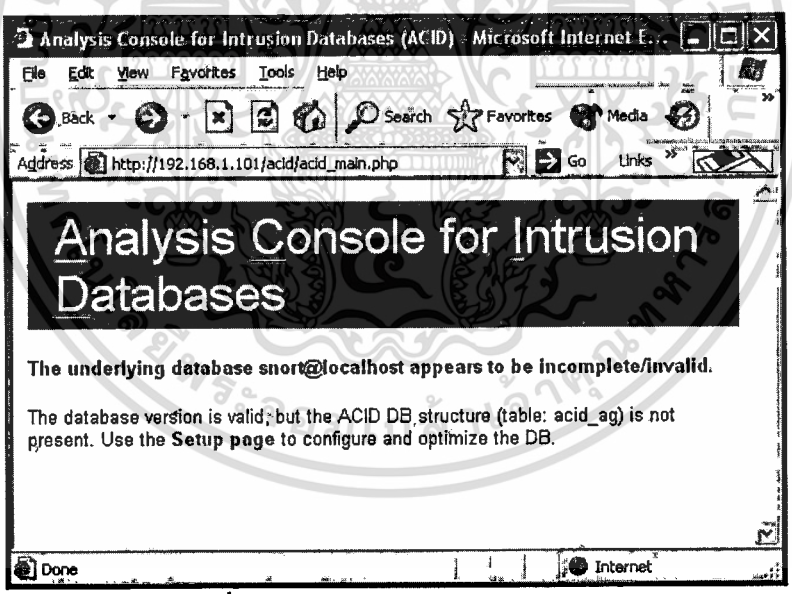
```
[root@sensor package]# tar –xvzf adodb401.tgz
[root@sensor package]# tar –xvzf acid-0.9.6b23.tar.gz
```

หลังจากนั้นเข้าไปแก้ไขค่าในไฟล์ acid_conf.php ซึ่งอยู่ใน /home/package/acid บรรทัดที่ต้องแก้ไขมีดังนี้

```
$DBlib_path = "/home/package/adodb";
$DBtype = "mysql";

$alert_dbname = "snort";
$alert_host = "localhost";
$alert_port = "";
$alert_user = "snort";
$alert_password = "itsnort";
```

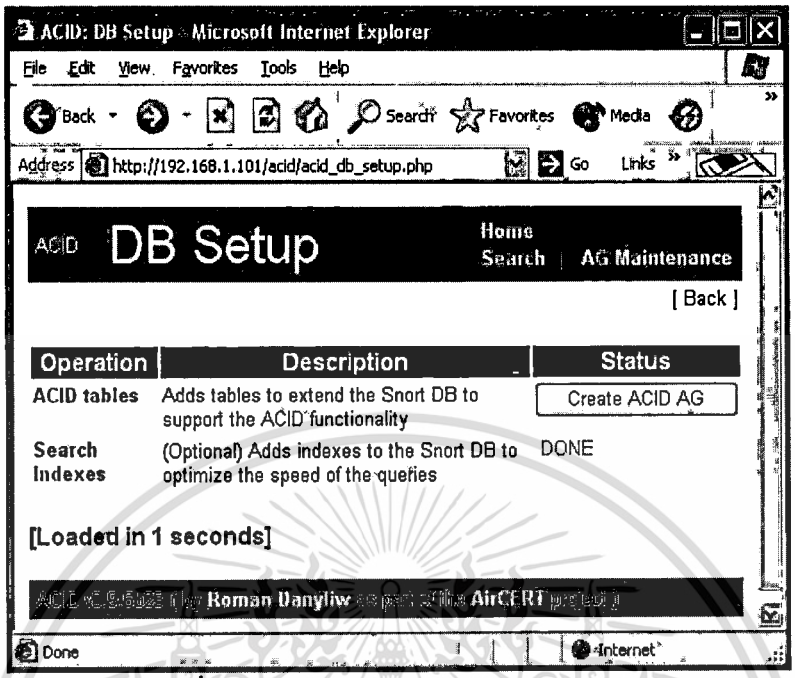
```
$archive_dbname = "snort";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "snort";
$archive_password = "itsnort";

$ChartLib_path = "/home/package/jpgraph-1.13/scr";
$chart_file_format = "png";
```

สุดท้ายให้ทดสอบการทำงานของเว็บเซิร์ฟเวอร์ และ ACID โดยเข้าไปที่เว็บบราวเซอร์ พิมพ์ http://your_sensor_ip_address/acid/acid_main.php ถ้าการทำงานของเว็บเซิร์ฟเวอร์ และ ACID ถูกต้องจะแสดงผลดังรูปที่ 2

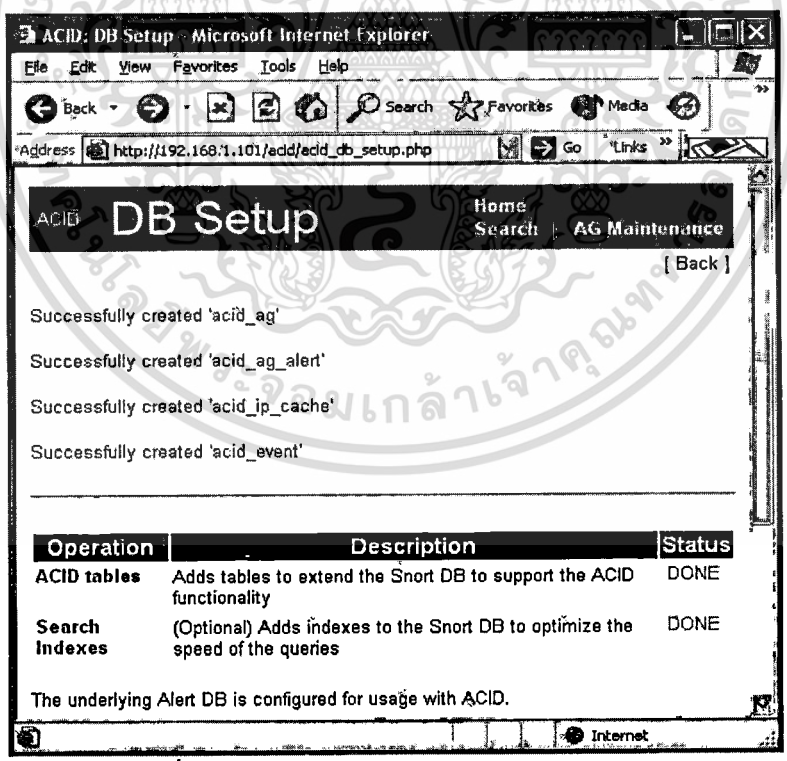


รูปที่ 2 หน้าจอสำหรับสร้าง ACID DB

ให้เลือกที่ Setup page แล้วกดปุ่ม Create ACID AG ที่ปรากฏอยู่ในรูปที่ 3 ก็จะได้ผลลัพธ์ดังรูปที่ 4

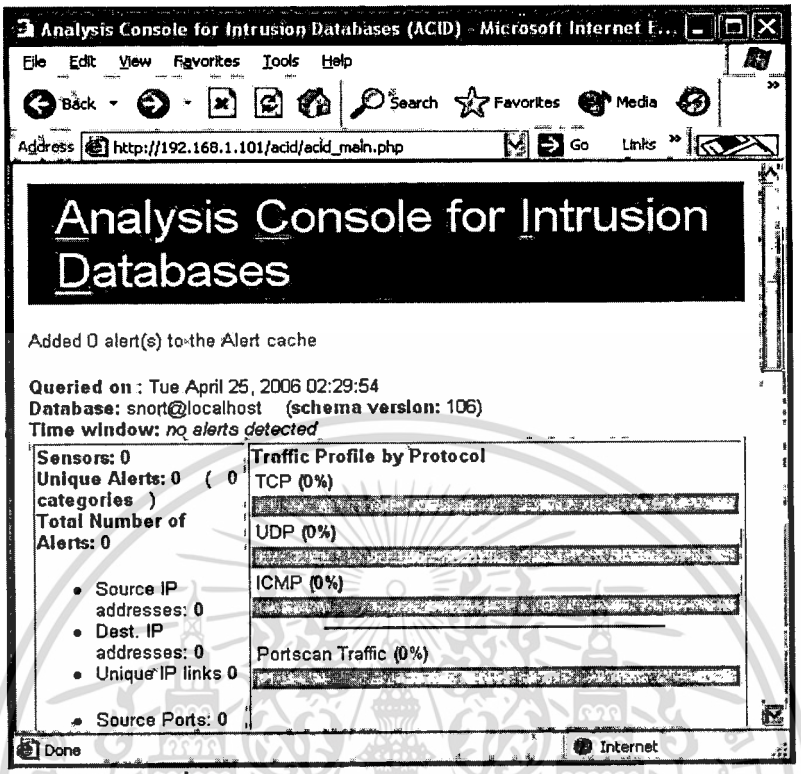


รูปที่ 3 หน้าจอสำหรับสร้าง ACID DB (ต่อ)



รูปที่ 4 หน้าจอแสดงผลของการสร้าง ACID DB

จากนั้นในช่อง URL ให้พิมพ์ http://your sensor ip address/acid/ จะได้ผลลัพธ์ดังรูปที่ 5



รูปที่ 5 หน้าจอแสดงผลการตรวจจับโดย ACID

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายจตุพล นิลพรัตน์
วุฒิการศึกษาระดับปริญญาตรี	วศ.บ. (วิศวกรรมไฟฟ้า)
	คณะวิศวกรรมศาสตร์
	มหาวิทยาลัยเชียงใหม่