สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การพัฒนาโปรแกรมตัวแปลภาษาเอนเอเอสแอลสำหรับวินเนสซัส

Implementation The NASL-Interpreter for win-nessus

T061409

จัดทำโดย

นาย อภิศักดิ์ สีหามาตย์

นาย ณัฐวุฒิ ฐินถาวร

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

อาจารย์ธนัญชัย ตรีภาค

เลขหมู่.....
เลขทะเบียน 61409
วัน,เดือน,ปี 17 ก.ค. 2549

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การพัฒนาโปรแกรมตัวแปลภาษาเอนเอเอสแอลสำหรับวินเนสซัส

นาย อภิศักดิ์ สีหามาตย์ รหัสประจำตัว 45015399
นาย ณัฐวุฒิ ฐินถาวร รหัสประจำตัว 45015364
อาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
อาจารย์ธนัญชัย ตรีภาค อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

ปริญญานิพนธ์นี้จึงมุ่งเน้นการศึกษาโปรแกรมที่ทำหน้าที่สแกนระบบที่ชื่อว่า Nessus ซึ่งโปรแกรมนี้มีหน้าที่ตรวจจับหาจุดบกพร่องในเครือข่าย ซึ่งมีการทำงานลักษณะ ลูกข่าย(Client) กับแม่ข่าย (Server) และเนื่องจากมีจุดอ่อนเกิดขึ้นใหม่ได้ทุกๆ วัน เพื่อให้สามารถตรวจสอบช่องโหว่ใหม่ๆ ได้ทันการพัฒนาของผู้ประสงค์ร้ายที่ต้องการโจมตีจุดอ่อนที่เกิดขึ้นใหม่ได้ โปรแกรมนี้ได้อนุญาตให้ผู้ใช้เขียนสคริปที่คล้ายกับลักษณะของไวรัสในโปรแกรมสแกนหาไวรัส ซึ่งก็สามารถเขียนได้ในหลายภาษาแต่ที่นิยมเขียนกันมากคือ ภาษาเอนเอเอสแอล (Nessus Attack Script Language : NASL) ภาษาเอนเอเอสแอลนี้เป็นภาษาที่ Nessus ออกแบบมาสำหรับให้เขียนทำการทดสอบหาช่องโหว่ Nessus เป็นโปรแกรมที่เป็นแบบ GPL ทำให้มีการนำไปใช้กันมาก และมีประสิทธิภาพสูง แต่จุดอ่อนของระบบนี้คือส่วนแม่ข่ายจะต้องทำงานในยูนิกส์เท่านั้น ทำให้ผู้ใช้ที่ไม่คุ้นเคยกับยูนิกส์มีความลำบากในการนำไปใช้ โดยเฉพาะคนไทย ซึ่งจากการศึกษาเบื้องต้นพบว่ามีนาย Ahmed El Deeb เป็นผู้ที่มีแนวคิดที่จะพัฒนาโปรแกรม Nessus ให้สามารถใช้งานได้กับระบบที่ใช้วินโดวส์เช่นเดียวกัน ชื่อโครงงาน Win-nessus หากแต่ยังขาดผู้มีความถนัดในการสร้างตัวแปลภาษา ดังนั้นโครงงานนี้จึงนำเสนอที่จะพอร์ตโปรแกรมแปลภาษาเอนเอเอสเอล (ซึ่งเป็นส่วนหนึ่งที่สำคัญของโปรแกรม Nessus ส่วนแม่ข่าย) จากที่ทำงานบนระบบยูนิกส์ มาให้ทำงานในระบบที่ใช้วินโดวส์ (Win32)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Implementation The NASL-Interpreter for win-nessus

Mr. Apisak Srihamat
Mr. Nattawut Thinthaworn
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor
Mr. Thanunchai Threepak Advisor

ABSTACT

This Project to emphasize leanning about vulnerability scanner called Nessus, this program have responsibility detect velnerability in network, which it work be client-server environment. Because of new vulnerabilities are being discovered and disseminated all the time. So that should be overtake bad guy, nessus to allow user write script vary much like virus signatures in a common virus scanner application, that script can be written in most any language but usually are written in the Nessus Attack Scripting Language (NASL). NASL is Nessus' own language, specifically designed for vulnerability test writing. Nessus is GPL a lot of people in the world use it, because efficiency. but disadvantage of this program to be path Server work with UNIX environment only, it make people not be UNIX user can hardly use it, especially Thai people. At the beginning research discover Mr. Ahmed El Deeb, he has the initiative to develop Nessus work in windows environment, His project name is Win-nessus. But to lack somebody have skillfull about develop NASL-Interpreter. So this project offer port program NASL-Interpreter (Important path of Nessus-server) from UNIX environment to work with Win32 environment

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้าที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้าที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ

หน้าที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ (ต่อ)

หน้าที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หน้าที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมตัวแปลภาษาเอนเอเอสแอลสำหรับวินเนสซัส

Implementation The NASL-Interpreter for win-nessus

ผู้จัดทำ

1. นาย อภิศักดิ์ สีหามาตย์
2. นาย ณัฐวุฒิ ฐินถาวร

_____ อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ ธนัญชัย ตรีภาค)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี เนื่องจากได้รับการแนะนำ สนับสนุน และให้คำปรึกษาเป็นอย่างดีจาก อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ อัครเดช วัชระภูพงษ์ และ อาจารย์ ธนัญชัย ตรีภาค อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งต้องขอขอบคุณเป็นอย่างสูง รวมทั้งอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ให้การอบรมสั่งสอนวิชาความรู้แก่คณะผู้จัดทำมาโดยตลอด

และขอขอบพระคุณเป็นอย่างสูงสำหรับบุคคลที่สำคัญที่สุดที่ทำให้คณะผู้จัดทำมีวันนี้ คือ บิดา มารดา ผู้เป็นที่เคารพรักยิ่งของคณะผู้จัดทำ ซึ่งท่านให้การอบรมสั่งสอน เลี้ยงดู และให้โอกาสในการศึกษาอย่างเต็มที่ จึงขอกราบขอบพระคุณมา ณ ที่นี้

สุดท้ายนี้ขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่อำนวยความสะดวกในการใช้งานเครือข่าย และขอขอบคุณเพื่อนๆ ที่ให้ข้อคิดและเป็นกำลังใจให้เสมอมา

อภิศักดิ์ สีหามาตย์

ณัฐวุฒิ ฐินถาวร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันการตรวจสอบระบบความปลอดภัยของระบบเครือข่ายคอมพิวเตอร์ ภายในองค์กรมาความสำคัญและจำเป็นอย่างยิ่ง เพื่อป้องกัน และระวังภัยที่เกิดจากความเสียหายจากการบุกรุก ของผู้ประสงค์ร้ายจากทั้งภายในและภายนอกองค์กร รวมทั้งช่วยลดความสูญเสียความลับ และข้อมูลสำคัญขององค์กร การใช้ซอร์ฟแวร์ Vulnerability Scanners เป็นแนวทางหนึ่งของการสร้างความปลอดภัยให้กับระบบเครือข่ายคอมพิวเตอร์ อีกทั้งยังช่วยเหลือผู้ดู แลระบบในการตรวจสอบช่องโหว่ต่างๆ ของเครื่องคอมพิวเตอร์ในระบบเครือข่าย ซอฟต์แวร์ดังกล่าวจะทำหน้าที่ในการทดสอบ และการค้นหาพอร์ต หรือโปรแกรมที่มีช่องโหว่ที่เป็นอันตรายแก่ระบบเครือข่าย เช่น ช่องโหว่ที่เกิดจากโปรแกรมม้าโทรจัน (Trojan Horse) เป็นต้น

NESSUS (www.nessus.org) เป็นโปรแกรมในการตรวจสอบระบบคอมพิวเตอร์เพื่อค้นหาจุดอ่อน (Vulnerability) หรือข้อบกพร่องต่างๆ ที่อาจมีอยู่ในระบบซึ่งอาจจะนำไปสู่การบุกรุกเข้าสู่ระบบได้ ซึ่ง อาจจะกระทำโดยผู้ไม่ประสงค์ดีอื่นๆก็ตาม โปรแกรมนี้ได้รับการพัฒนาโดย Renaud Deraison ซึ่งเป็นโปรแกรมที่แจกให้ใช้งานฟรีตามข้อตกลงของ GUN License

ปริญญานิพนธ์นี้จึงมุ่งเน้นการศึกษาโปรแกรมที่ทำหน้าที่สแกนระบบที่ชื่อว่า Nessus ซึ่งโปรแกรมนี้มีหน้าที่ตรวจจับหาจุดบกพร่องในเครือข่าย ซึ่งมีการทำงานลักษณะไคลเอนต์-เซิร์ฟเวอร์ และเนื่องจากมีจุดอ่อนเกิดขึ้นใหม่ได้ทุกๆ วัน เพื่อให้สามารถตรวจสอบช่องโหว่ใหม่ๆ ได้ทันการพัฒนาของผู้ประสงค์ร้ายที่ต้องการโจมตีจุดอ่อนที่เกิดขึ้นใหม่ได้ โปรแกรมนี้ได้อนุญาตให้ผู้ใช้เขียนสคริปที่คล้ายกับลักษณะของไวรัสในโปรแกรมสแกนหาไวรัส ซึ่งก็สามารถเขียนได้ในหลายภาษาแต่ที่นิยมเขียนกันมากคือ ภาษาเอนเอเอสแอล (Nessus Attack Script Language : NASL) ภาษาเอนเอเอสแอลนี้เป็นภาษาที่ Nessus ออกแบบมาสำหรับให้เขียนทำการทดสอบหาช่องโหว่ โดย Nessus เป็นโปรแกรมที่เป็นแบบ GPL ทำให้มีการนำไปใช้กันมาก และมีประสิทธิภาพสูง แต่จุดอ่อนของระบบนี้คือส่วนเซิร์ฟเวอร์จะต้องทำงานในยูนิกส์เท่านั้น ทำให้ผู้ใช้ที่ไม่คุ้นเคยกับยูนิกส์มีความลำบากในการนำไปใช้ โดยเฉพาะคนไทย ซึ่งจากการศึกษาเบื้องต้นพบว่ามีนาย Ahmed El Deeb เป็นผู้ที่มีแนวคิดที่จะพัฒนาโปรแกรม Nessus ให้สามารถใช้งานได้กับระบบที่ใช้วินโดวส์เช่นเดียวกัน ชื่อโครงงาน Win-nessus หากแต่ยังขาดผู้มีความถนัดในการสร้างตัวแปลภาษา ดังนั้นโครงงานนี้จึงนำเสนอที่จะพอร์ตโปรแกรมแปลภาษาเอนเอเอสเอล (ซึ่งเป็นส่วนหนึ่งที่สำคัญของโปรแกรม Nessus ส่วนเซิร์ฟเวอร์) จากที่ทำงานบนระบบยูนิกส์มาให้ทำงานในระบบที่ใช้วินโดวส์ (Win32)เพื่อเป็นประโยชน์กับการสร้าง Nessus ส่วนเซิร์ฟเวอร์ให้ทำงานบนระบบที่ใช้วินโดวส์ได้ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์ที่จัดทำนี้ จัดทำขึ้นภายใต้วัตถุประสงค์หลัก 2 ประการ ได้แก่

(1) เพื่อศึกษาการทำงานและโครงสร้างของโปรแกรม Nessus

(2) เพื่อพอร์ตตัวแปรภาษาเอนเอเอสแอลจากยูนิกส์ให้มาทำงานในวินโดวส์

1.3 ขอบเขตของปริญญานิพนธ์

ขอบเขตการทำงานของปริญญานิพนธ์นี้ได้แก่

(1) วิเคราะห์ภาษาเอนเอเอสแอลและจัดทำกฎที่ใช้ระบุโทเคนของภาษาเอนเอเอสแอล

(2) นำเฟลคและไบสันมาใช้เป็นเครื่องมือสร้างตัวแปลภาษาเอนเอเอสแอล

(3)สร้างตัวแปลภาษาเอนเอเอสแอลจากเฟลคและไบสันให้สามารถนำไปใช้ประโยชน์กับวินเนสซัส ซึ่งเป็นเนสซัสที่ทำงานบนระบบวินโดวส์ได้

1.4 ขั้นตอนการดำเนินงาน

(1) ศึกษารายละเอียดการทำงานของ Nessus

(2) ศึกษาเกี่ยวกับการพอร์ตโปรแกรมจากระบบยูนิกส์ให้มาทำงานในวินโดวส์

(3) ศึกษาวิธีการสร้างตัวแปลภาษา

(4) ศึกษาภาษาเอนเอเอสแอล

(5)ออกแบบขั้นตอนการพอร์ตโปรแกรมตัวแปลภาษาจากระบบยูนิกส์ให้มาทำงานในวินโดวส์

(6) พัฒนาโปรแกรมตัวแปลภาษาเอนเอเอสแอลบนแพลตฟอร์มวินโดวส์

(7) ทดสอบและปรับปรุงตัวแปลภาษาเอนเอเอสแอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

Nessus

2.1 Nessus คืออะไร

Nessus เป็นโปรแกรมที่ใช้ในการตรวจสอบระบบคอมพิวเตอร์เพื่อค้นหาจุดอ่อน(Vulnerability) หรือข้อบกพร่องต่างๆ ที่อาจมีอยู่ในระบบซึ่งอาจจะนำไปสู่การบุกรุกเข้าสู่ระบบได้ ซึ่งอาจกระทำโดยผู้ไม่ประสงค์ดีอื่นๆก็ตาม โปรแกรมนี้ได้รับการพัฒนาโดย Renaud Deraison ซึ่งเป็นโปรแกรมที่แจกให้ใช้งานฟรีตามข้อตกลงของ GNU License โดยโปรแกรม Nessus แบ่งการทำงานออกเป็น 2 ส่วนคือ ส่วนแม่ข่าย (Server) และส่วนลูกข่าย (Client) โดยการทำงานของส่วนแม่ข่ายนั้นทำงานอยู่บนแพลตฟอร์มระบบปฏิบัติการยูนิกส์เท่านั้น และส่วนลูกข่ายสามารถทำงานได้ทั้งบนแพลตฟอร์มระบบปฏิบัติการอื่นๆ เช่น วินโดว์

2.2 วิธีการติดตั้งโปรแกรม Nessus

ดาวน์โหลดโปรแกรม Nessus นี้ได้จากเว็บไซต์ http://www.nessus.org ภายใต้หัวข้อ Download การติดตั้ง Nessus สามารถทำได้ 3 วิธี (อย่างใดอย่างหนึ่ง) ดังนี้

วิธีแรก: ติดตั้งผ่านทาง Internet โดยผ่านโปรแกรม lynx ซึ่งเป็นโปรแกรมเว็บเบราเซอร์โปรแกรมหนึ่ง (ดาวน์โหลดได้จาก http://lynx.browser.org) การติดตั้งทำโดยใช้คำสั่งดังนี้

```
#lynx -source http://install.nessus.org | sh
```

วิธีที่สอง: ดาวน์โหลดสคริปต์ไฟล์ที่มีชื่อว่า nessus-installer.sh (ซึ่งอยู่ในหัวข้อ Download และอยู่ภายใต้ไดเรคตอรี่ nessus-installer/) และทำการติดตั้งโดยใช้คำสั่งดังนี้

```
#sh nessus-installer.sh
```

วิธีที่สาม: ดาวน์โหลดชุดไฟล์คอมไพล์แพ็กเกจซึ่งประกอบไปด้วย

- nessus-libraries-x.x.tar.gz
- libnasl-x.x.tar.gz
- nessus-core.x.x.tar.gz
- nessus-plugins.x.x.tar.gz

(x แทนด้วยตัวเลขเวอร์ชั่นของซอฟต์แวร์ขณะนั้น)

ทำการขยายทุกไฟล์ออกมาโดยใช้คำสั่งดังนี้

```
#tar xvfz nessus-libraries-x.x.tar.gz
#tar xvfz libnasl-x.x.tar.gz
#tar xvfz nessus-core.x.x.tar.gz
#tar xvfz nessus-plugins.x.x.tar.gz
```

ทำการคอมไพล์ nessus-libraries ตามขั้นตอนดังนี้

```
#cd nessus-libraries
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

#./configure

#make

#make install (คำสั่งติดตั้งบรรทัดนี้ต้องใช้ login root เท่านั้น)

ทำการคอมไพล์ libnasl ตามขั้นตอนดังนี้

#cd libnasl

#./configure

#make

#make install (คำสั่งติดตั้งบรรทัดนี้ต้องใช้ login root เท่านั้น)

ทำการคอมไพล์ nessus-core ตามขั้นตอนดังนี้

#cd nessus-core

#./configure

#make

#make install (คำสั่งติดตั้งบรรทัดนี้ต้องใช้ login root เท่านั้น)

ทำการคอมไพล์ nessus-plugins ตามขั้นตอนดังนี้

#cd nessus-plugins

#./configure

#make

#make install (คำสั่งติดตั้งบรรทัดนี้ต้องใช้ login root เท่านั้น)

ภายหลังการคอมไพล์ทั้งหมดเสร็จสิ้นแล้ว จะได้ไฟล์ 2 ไฟล์ที่สำคัญออกมาคือไฟล์ nessusd ซึ่งทำหน้าที่เป็นเซิร์ฟเวอร์ของ Nessus และได้ไฟล์ nessus ซึ่งทำหน้าที่เป็นลูกข่ายของ Nessus โดยทำงานร่วมกับเซิร์ฟเวอร์ nessusd กรณีที่ใช้ลีนุกซ์ ต้องเพิ่ม /usr/local/lib เข้าไปในไฟล์ /etc/ld.so.conf เพื่อเพิ่มไดเรกเทอรีซึ่งมีไลบรารี่ของ Nessus เข้าไปในระบบ และใช้คำสั่ง ldconfig เพื่อทำการอัพเดทไดเรกทอรี่ดังกล่าว

#echo "/usr/local/lib" >> /etc/ld.so.conf

#ldconfig

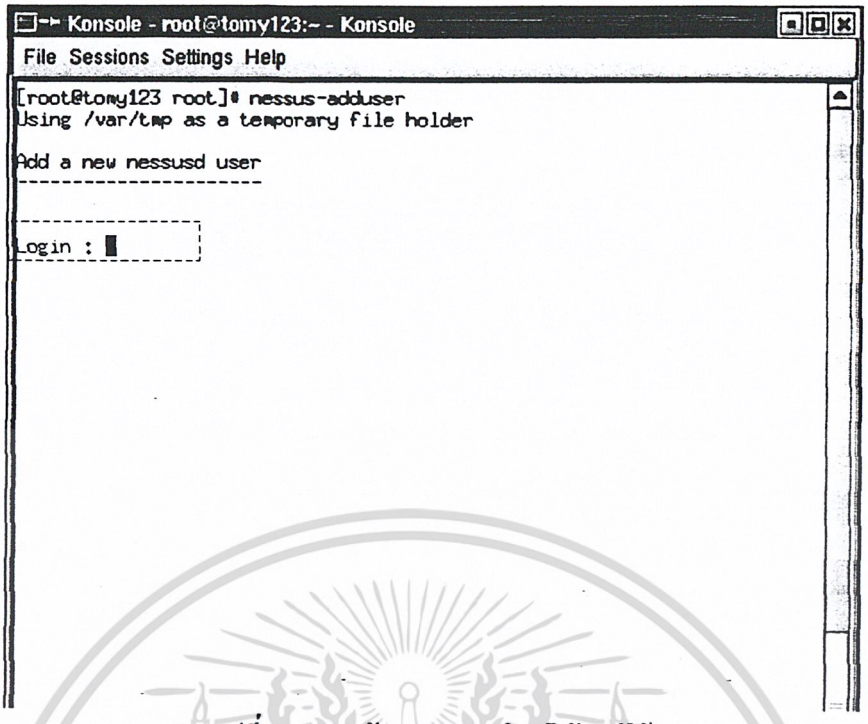
2.3 การปรับแต่งค่าของโปรแกรม Nessus

การปรับแต่งค่าของโปรแกรม Nessus ก่อนการใช้งานแบ่งออกเป็น 2 ส่วนคือ การสร้าง Account ใหม่ให้กับผู้ใช้พร้อมกำหนดสิทธิ์การใช้งานและการปรับแต่งค่าของลูกข่าย nessus

2.3.1. การสร้าง Account ใหม่ให้กับผู้ใช้และการกำหนดสิทธิ์การใช้งาน

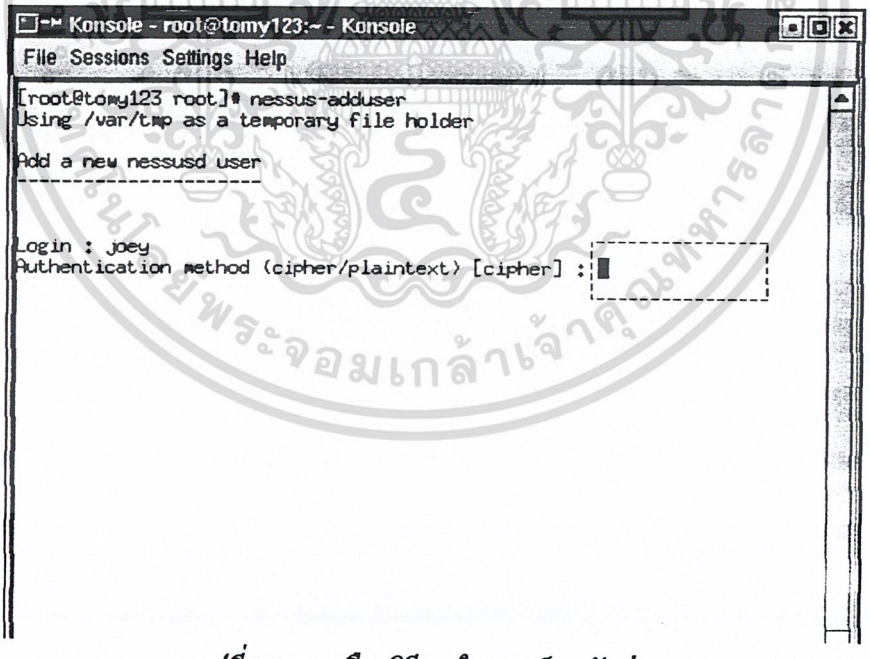
ใช้สคริปต์ที่ชื่อว่า nessus-adduser ซึ่งจะอยู่ในไดเรกทอรี /usr/local/sbin เพื่อสร้าง account ใหม่ให้กับผู้ใช้เพื่อใช้ในการติดต่อไปยัง nessusd เซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.1 การสร้าง Account ใหม่ให้กับผู้ใช้

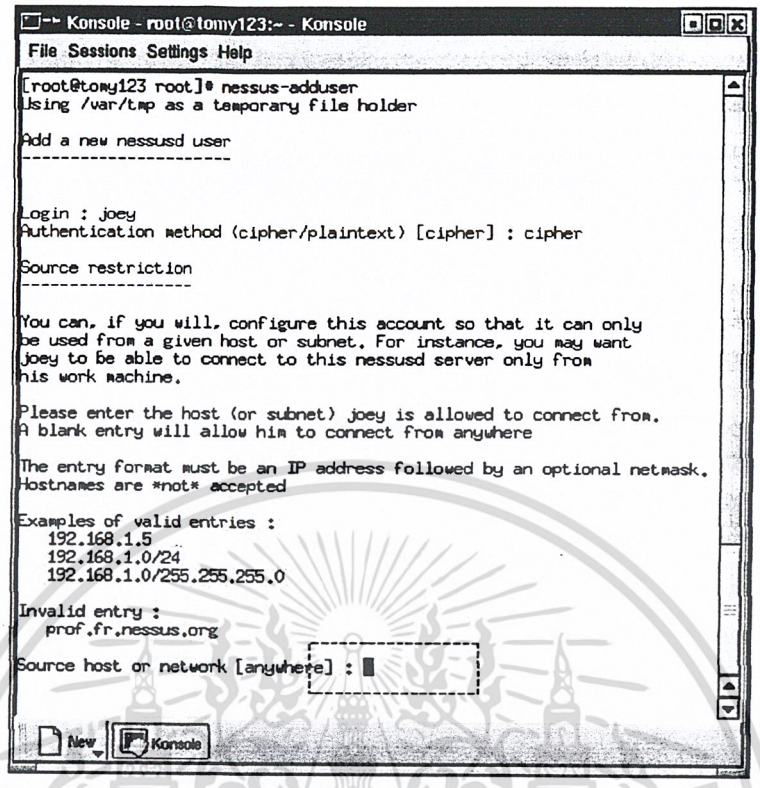
จากรูปที่ 2.1 ให้ใส่ชื่อ Account ที่จะใช้ในการล๊อกอินเพื่อติดต่อกับเซิร์ฟเวอร์ nessusd



รูปที่ 2.2 การเลือกวิธีการในการเก็บรหัสผ่าน

จากรูปที่ 2.2 นี้ให้ใส่วิธีการเก็บรหัสผ่านบนเซิร์ฟเวอร์ของ Account นี้ว่าต้องการให้เป็นแบบเข้ารหัสหรือเป็นแบบข้อความธรรมดาโดยใส่คำว่า plaintext ถ้าไม่ต้องการเข้ารหัส หรือใส่คำว่า cipher ถ้าต้องการเข้ารหัสของรหัสผ่านขอเรียกชื่อ Account joey นี้ว่าเป็น Login name ของ Nessus และขอเรียกชื่อ Account root บน ลีนุกซ์พร้อมท์ (ดูในรูปที่ 2.2 ด้วย) ว่าเป็น User name บนลีนุกซ์ ทั้งนี้เพื่อความแตกต่าง

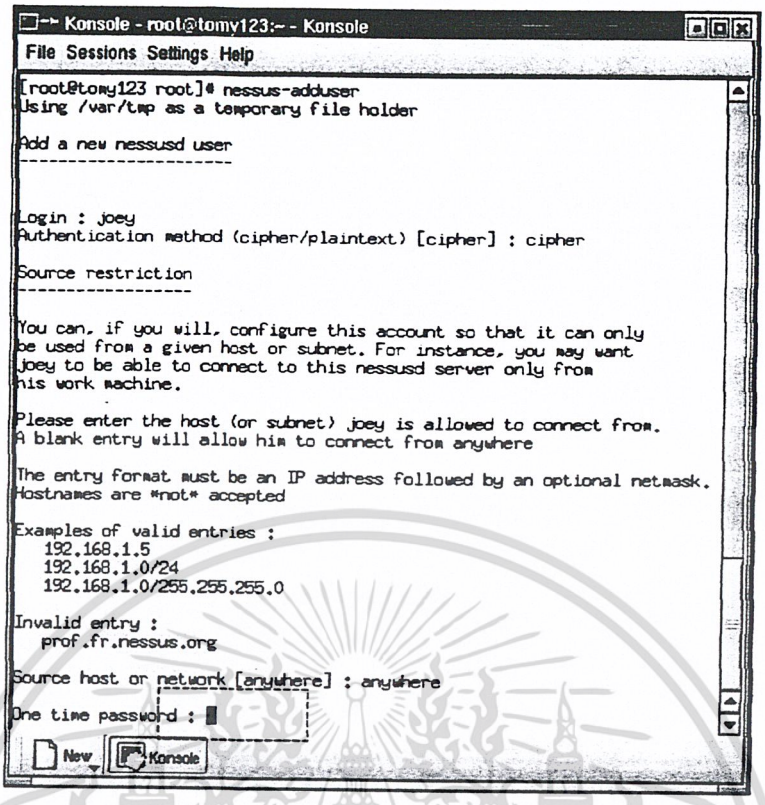
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.3 การกำหนดสิทธิ์ให้ผู้ใช้สามารถทำการล็อกอินเข้ามาใช้งาน Nessus ได้จากที่ไหนบ้าง

จากรูปที่ 2.3 ผู้สร้าง Account ให้กับ joey จะสามารถกำหนดให้ joey ทำการล็อกอินเข้ามาใช้ Nessus จากส่วนใดของเน็ตเวิร์คได้บ้าง ผู้สร้างสามารถกำหนดได้ เช่น เฉพาะเบอร์ IP Address ซึ่งเป็นเครื่องใช้งานของ joey หรืออาจกำหนดเป็นซับเน็ตเช่น 192.168.1.0/24 เป็นต้น จากรูปค่าดีฟอลท์คือ anywhere ซึ่งหมายถึง joey สามารถล็อกอินจากที่ไหนเข้ามาก็ได้

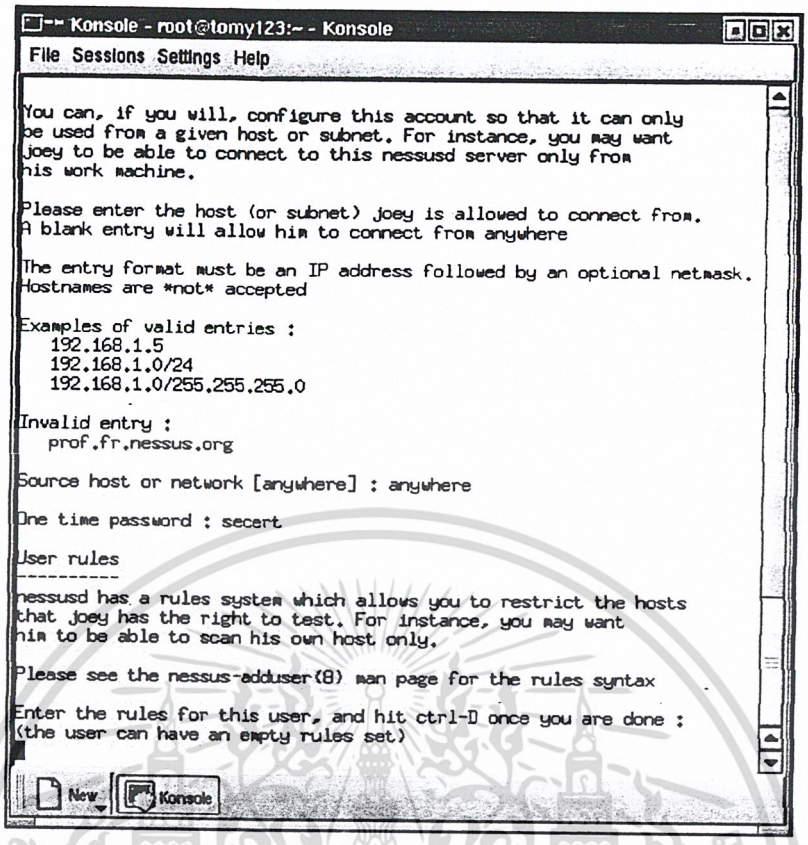
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.4 การกำหนดรหัสผ่าน

รูปที่ 2.4 เป็นการกำหนดรหัสผ่านให้กับ Account joey รหัสผ่านนี้จะถูกสอบถามโดยเซิร์ฟเวอร์ nessusd เพียงครั้งเดียว กล่าวคือในครั้งแรกที่ผู้ ใช้ joey ทำการล็อกอินเข้ามาโดยโปรแกรมลูกข่าย nessus และในครั้งต่อๆไปเมื่อ joey ทำการล็อกอินเข้ามาอีก เซิร์ฟเวอร์ nessusd จะไม่มีการถามถึงรหัสผ่านนี้อีกเลย ดังนั้นจึงเป็นสาเหตุของการเรียกรหัสผ่านนี้ว่า one-time password อนึ่งในการล็อกอินเข้ามาใน Nessus จำเป็นต้องมีการใส่ passphrase ซึ่งเป็นคนละตัวกับ onetime password ณ ที่นี้ โดยในการล็อกอินเข้ามาใช้ Nessus นั้นผู้ใช้จำเป็นต้องมีการใส่ passphrase เสมอ และหลังจากนั้น ถ้าเป็นการล็อกอินครั้งแรก ระบบจะขอให้ผู้ใช้ใส่ one-time password ด้วย จะได้มีการกล่าวถึงเรื่อง passphrase นี้เพิ่มเติมต่อไป

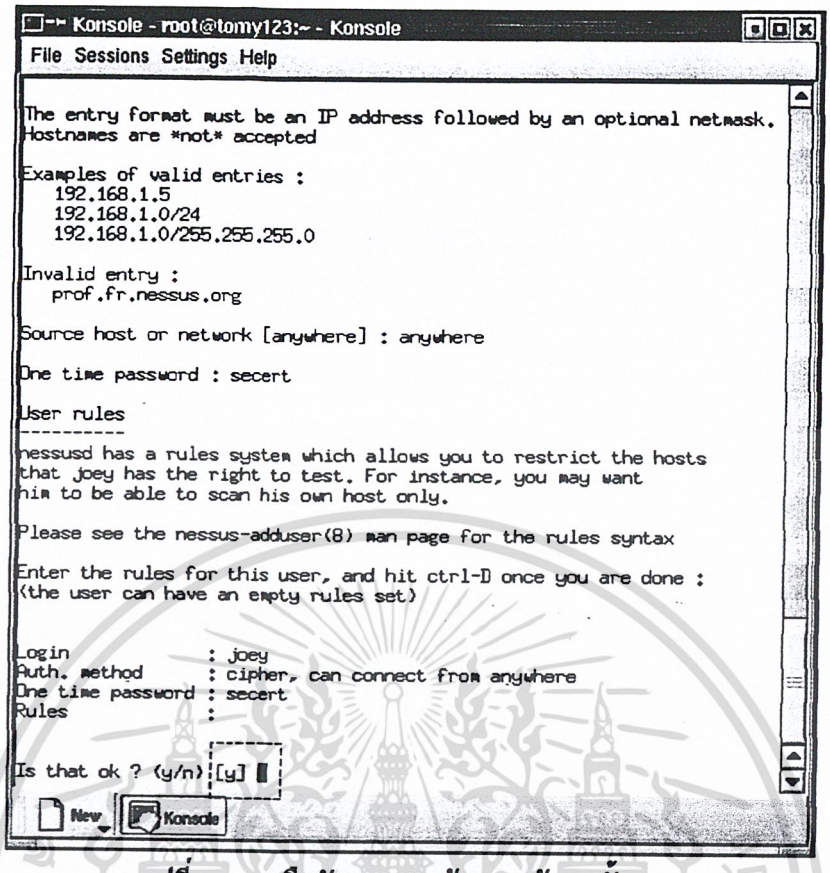
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.5 การกำหนดสิทธิ์ว่า joey สามารถทำการสแกนเน็ตเวิร์คส่วนไหนได้บ้าง

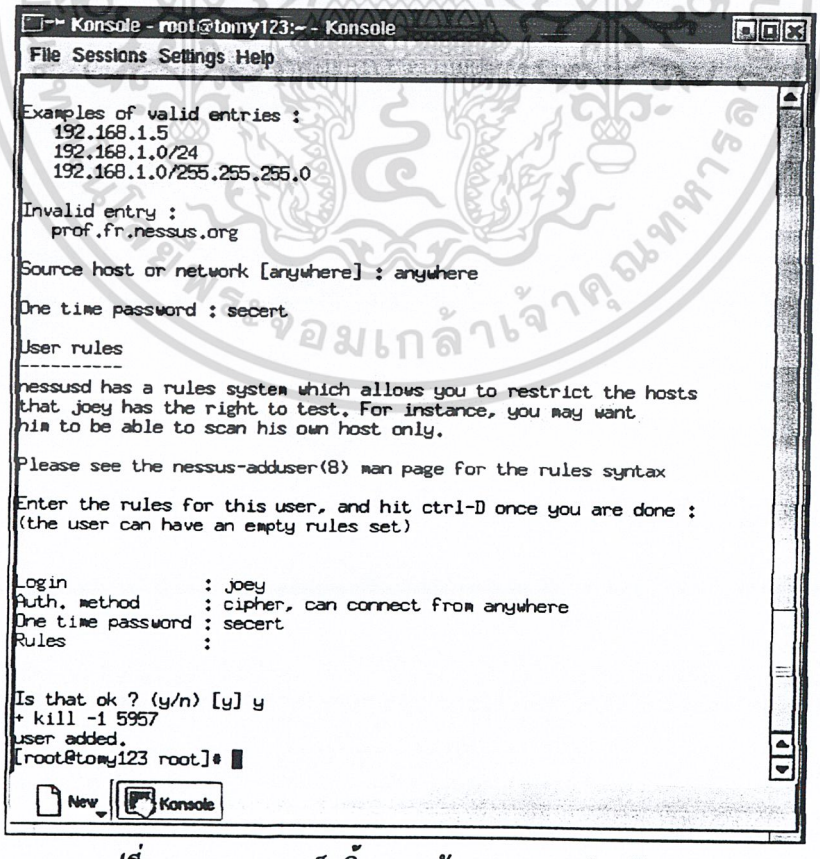
รูปที่ 2.5 นี้เป็นการกำหนดสิทธิ์การใช้งานของ joey ว่าให้สามารถตรวจสอบช่องโหว่ของเครื่องในเน็ตเวิร์คโดยมีขอบเขตการสแกนทำได้ถึงไหน เป็นต้นว่า joey นี้สามารถตรวจสอบช่องโหว่ได้เฉพาะที่ IP Address 192.168.1.53 เท่านั้นเป็นต้นเมื่อกำหนดสิทธิ์เสร็จแล้วให้กดปุ่ม Ctrl–D เพื่อเสร็จสิ้นการกำหนดสิทธิ์ ถ้าเราไม่ใส่กำหนดสิทธ์ใดๆเลย joey จะได้รับสิทธิ์ให้สามารถสแกนเครือข่ายได้ทั้งหมด โดยรายละเอียดเพิ่มเติมที่เกี่ยวข้องกับการกำหนดสิทธิ์นี้สามารถดูได้จากคู่มือแมนนวลเพจ nessus-adduser บนลีนุกซ์พร้อมท์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.6 การยืนยันความถูกต้องของข้อมูลทั้งหมด

ในขั้นนี้ระบบจะถามเพื่อยืนยันความถูกต้องของข้อมูลทั้งหมดที่ให้มาอีกครั้ง

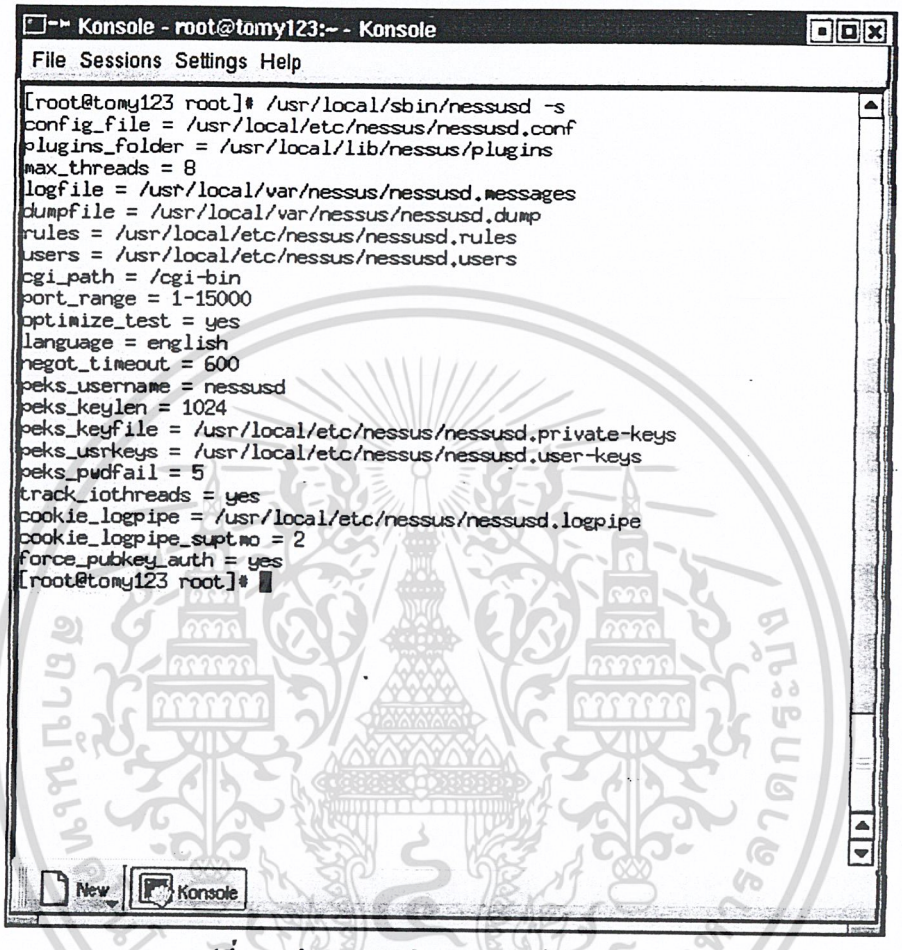


รูปที่ 2.7 แสดงการเสร็จสิ้นการสร้าง Account สำหรับ Joey

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

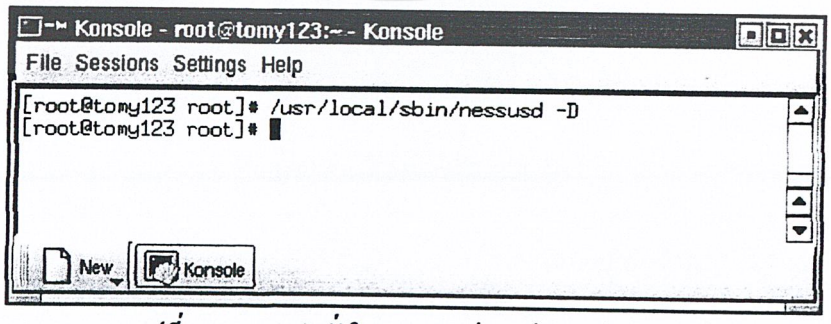
ถ้าตอบ y ผู้ใช้ใหม่ joey จะถูกเพิ่มเข้าไปในระบบและหน้าจอจะขึ้น user added. ซึ่งแสดงว่า Account joey นี้ได้รับการสร้างเสร็จเรียบร้อยแล้ว

nessusd จะมีไฟล์ชื่อ /usr/local/etc/nessus/nessusd.conf ซึ่งเก็บค่าคอนฟิกต่างๆของ Nessus เอาไว้ผู้อ่านสามารถใช้คำสั่ง nessusd –s เพื่อแสดงค่าคอนฟิกต่างๆออกมาทางหน้าจอได้ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 ค่าคอนฟิกต่างๆ ของเซิร์ฟเวอร์ nessusd

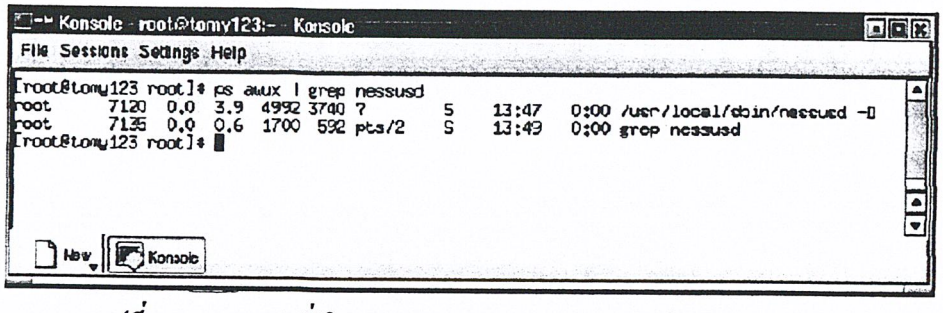
ภายหลังจากตรวจสอบค่าต่างๆในไฟล์ คอนฟิกเรียบร้อยแล้วผู้อ่านก็สามารถเริ่มต้นใช้งานเซิร์ฟเวอร์ nessusd ได้ แล้วโดยต้องทำการล็อกอินบนลินุกซ์เป็น root เท่านั้นจึงจะสามารถสตาร์ทเซิร์ฟเวอร์ได้โดยใช้คำสั่งดังรูป



รูปที่ 2.9 แสดงคำสั่งในการสตาร์ทเซิร์ฟเวอร์ nessusd

และสามารถตรวจสอบการทำงานของเซิร์ฟเวอร์ nessusd โดยใช้คำสั่งดังรูปที่ 2.10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

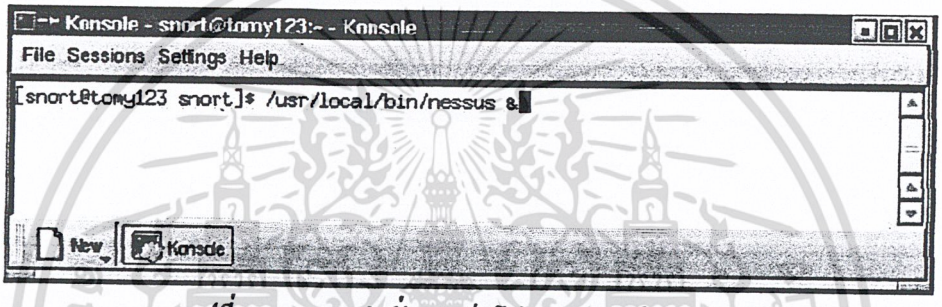


รูปที่ 2.10 แสดงคำสั่งในการตรวจสอบการทำงานของโปรเซส nessusd

2.4 การใช้งาน Nessus Client

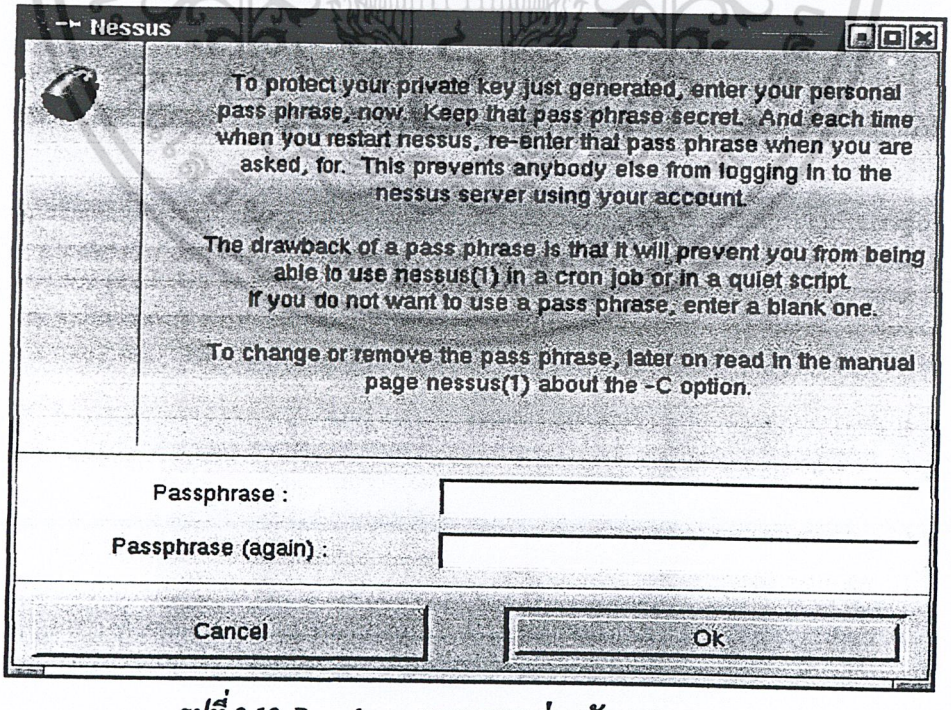
การปรับแต่งค่าของโปรแกรมลูกข่าย nessus

โปรแกรมลูกข่าย nessus จะอยู่ในไดเรคตอรี่ /usr/local/bin/nessus ผู้อ่านสามารถสตาร์ทโปรแกรมลูกข่าย nessus เพื่อใช้งานดังนี้



รูปที่ 2.11 แสดงคำสั่งสตาร์ทโปรแกรมลูกข่าย nessus

เครื่องหมาย & ในรูป เป็นการสั่งสตาร์ทโปรแกรมลูกข่ายในแบบแบ็คกราวด์ ให้สังเกตว่าขณะนี้ผู้ใช้ซึ่งอยู่บนลีนุกซ์ใช้ User name เป็น snort

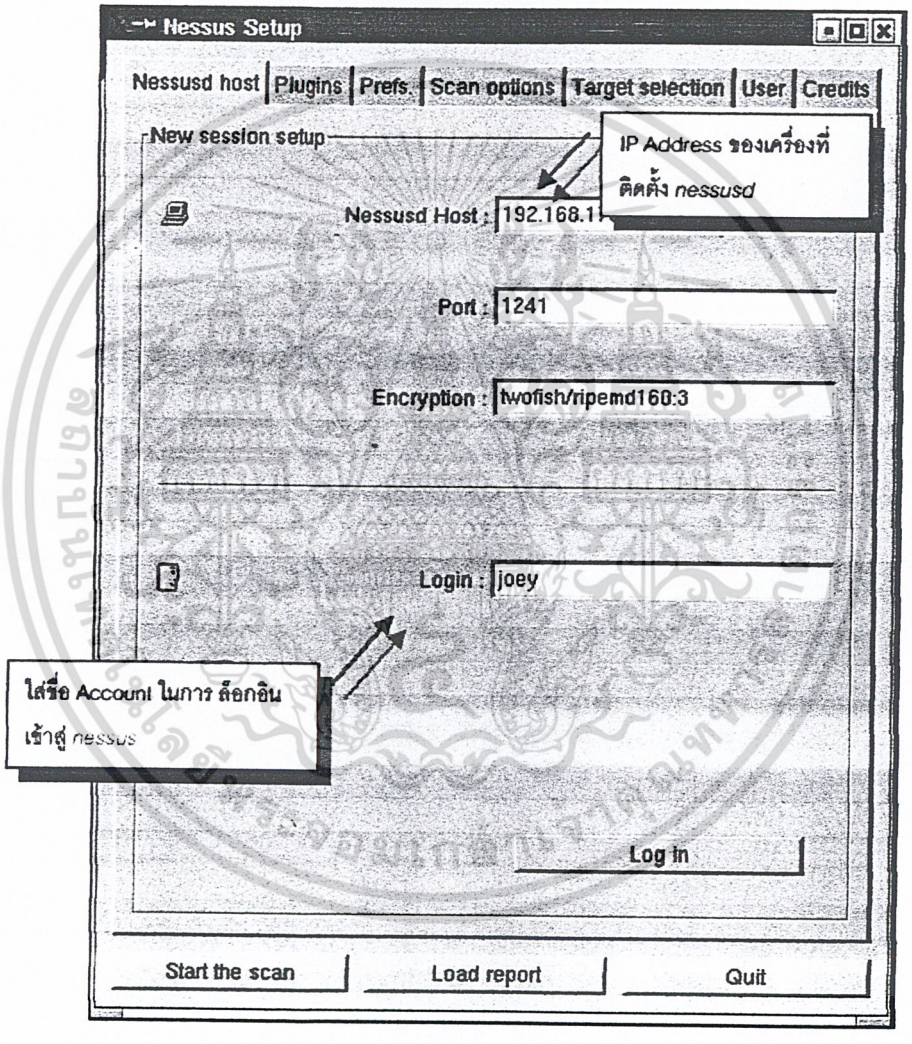


รูปที่ 2.12 Passphrase และกุญแจส่วนตัว (private key)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปเมื่อเริ่มต้นการใช้งานโปรแกรมลูกข่าย nessus เป็นครั้งแรก โปรแกรมจะทำการ สร้างกุญแจส่วนตัว (private key) ให้กับผู้ใช้ตาม User name (ซึ่งในที่นี้คือ snort ดูในรูปที่ 2.11 ด้วย) บนลินุกซ์กล่าวคือ 1 User name บนลินุกซ์เมื่อทำการสตาร์ท nessus เป็นครั้งแรกโปรแกรมจะทำการสร้างกุญแจส่วน ตัวให้กับ User name ผู้นั้นเพราะฉะนั้น 1 User name บนลินุกซ์จะคู่กับ 1 กุญแจส่วนตัวเสมอซึ่งเป็นความสัมพันธ์แบบ one-to-one

เมื่อสร้างกุญแจส่วนตัวเสร็จแล้วโปรแกรมจะขอให้ผู้ใช้ใส่ passphrase (ซึ่งเทียบได้กับรหัสผ่าน) ซึ่งเป็น passphrase ของกุญแจส่วนตัวที่เพิ่งสร้างขึ้นมานั่นเอง และผู้ใช้จะต้องเก็บ passphrase นี้ไว้เป็นความลับ บรรทัดที่สองในรูปจะเป็นการยืนยันความถูกต้องของ passphrase อีกครั้ง



รูปที่ 2.13 หน้าล็อกอินของโปรแกรมลูกข่าย nessus

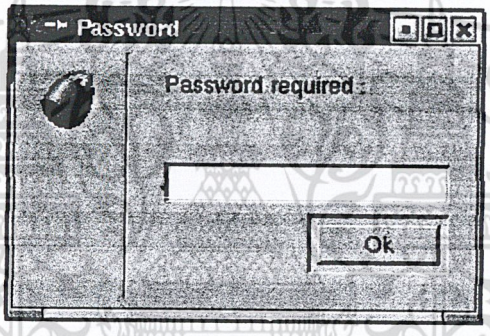
หน้าแรกของโปรแกรมลูกข่าย nessus จะปรากฏดังรูปที่ 2.13 ก่อนจะทำการล็อกอินด้วย Account ของผู้ใช้ (ในรูปคือ joey) ผู้ใช้จะต้องทำการใส่ IP Address ของเครื่องที่รันเซิร์ฟเวอร์ nessusd อยู่ ค่าพอร์ตที่เซิร์ฟเวอร์ ทำ งานอยู่ และวิธีการเข้ารหัสที่ใช้ในการติดต่อระหว่างลูกข่ายและเซิร์ฟเวอร์ในรูปเซิร์ฟเวอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะอยู่ที่ IP Address 192.168.176.210 ที่พอร์ต 1241 (ซึ่งเป็นค่าดีฟอลท์พอร์ตของ nessusd)และใช้วิธีการเข้ารหัสแบบ twofish/ripemod160:3

ให้สังเกตว่า Login name ของ nessus ซึ่งในรูปที่ 2.13 คือ joey ซึ่งอาจจะเป็นคนละชื่อกับ Username บนลีนุกซ์ซึ่งในขณะนี้คือ snort เมื่อ joey ทำการล็อกอินเพื่อติดต่อกับเซิร์ฟเวอร์ nessusd เป็นครั้งแรกโปรแกรมจะขอให้ผู้ใช้ใส่ one-time password ที่ joey ได้กำหนดเอาไว้ในรูปที่ 4 เมื่อใส่อย่างถูกต้อง

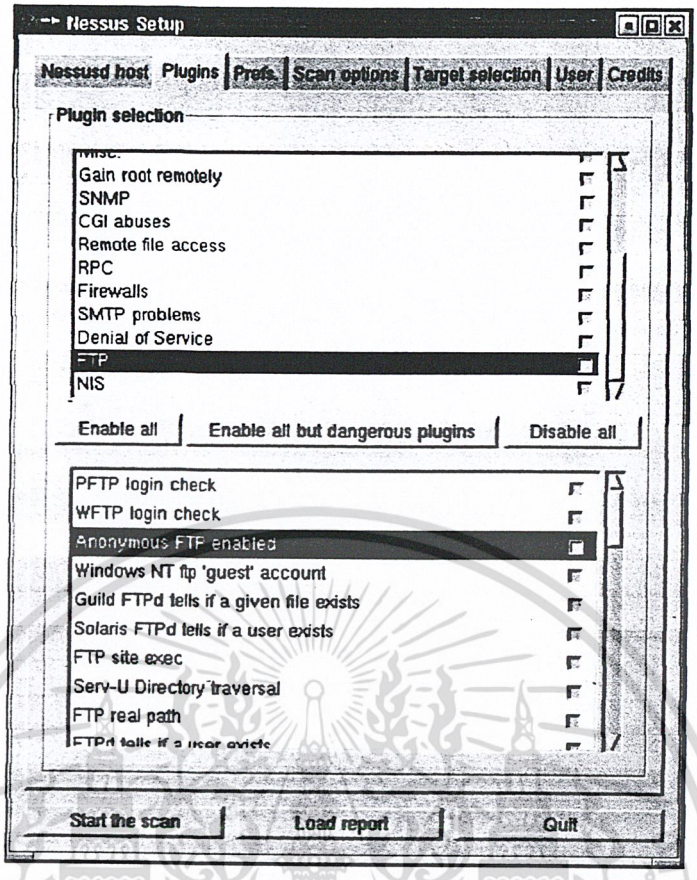
แล้ว โปรแกรมจะเอา Login name joey นี้ไปผูกติดกับกุญแจส่วนตัวของ User name snort บนลีนุกซ์ ดังนั้น Login name joey นี้จะไม่สามารถใช้ได้ในการล็อกอินเพื่อเข้า Nessus โดยใช้ User name อื่นๆนอกจาก snort และเมื่อ joey ทำการล็อกอินเข้า Nessus ในครั้งต่อๆไประบบจะสอบถามเฉพาะ passphraseของกุญแจส่วนตัวแต่จะไม่สอบถาม one-time password นี้อีกต่อไปอย่างไรก็ตาม User name หนึ่งชุดบนลีนุกซ์สามารถมี Login name ใน nessus ได้หลายชุด เช่นนอกจาก joey แล้วอาจจะมี Login name อื่นๆได้อีก โดยแต่ละ Login name อาจจะมีสิทธิ์ในการสแกนเน็ตเวิร์ค แตกต่างกันไป เป็นต้น



รูปที่ 2.14 แสดงหน้าจอให้ใส่ one-time password ของผู้ใช้

จากรูปหลังจากที่ใส่ Login name joey แล้ว ระบบจะขอให้ใส่รหัสผ่านของ Account นี้ตามที่ได้กำหนดไว้ในตอนที่สร้าง Account ใหม่นี้โดย nessus-adduser

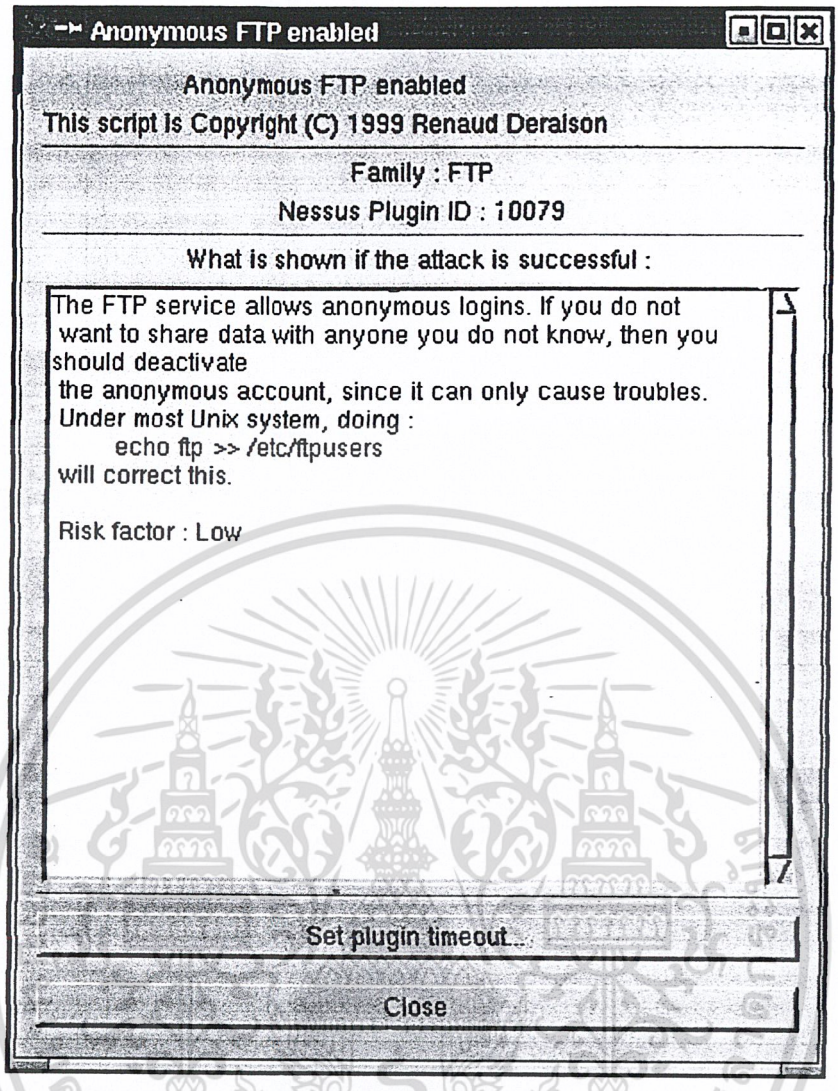
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.15 แสดงหมวด plugin หรือหมวดใหญ่ๆของจุดอ่อน

เมื่อเข้าสู่ระบบได้แล้วโปรแกรมจะอยู่ในหน้า plugin หรือหมวดใหญ่ๆของจุดอ่อนนั่นเอง เพื่อให้ผู้ใช้เลือกหมวดตามต้องการ โดยสามารถติ๊กเลือกเพื่อ Enable หรือติ๊กเอาออกเพื่อ Disable ในสี่เหลี่ยมเล็กๆทางขวามือ ส่วนหน้าต่างด้านล่างจะแสดงหมวดย่อยของ plugin หนึ่งๆให้ผู้ใช้สามารถทำการปรับแต่งค่าของโปรแกรมเพิ่มเติมได้ ในรูปจะประกอบไปด้วยจุดอ่อนของ FTP ในรูปแบบต่างๆ ที่สามารถเลือกเอาออกได้

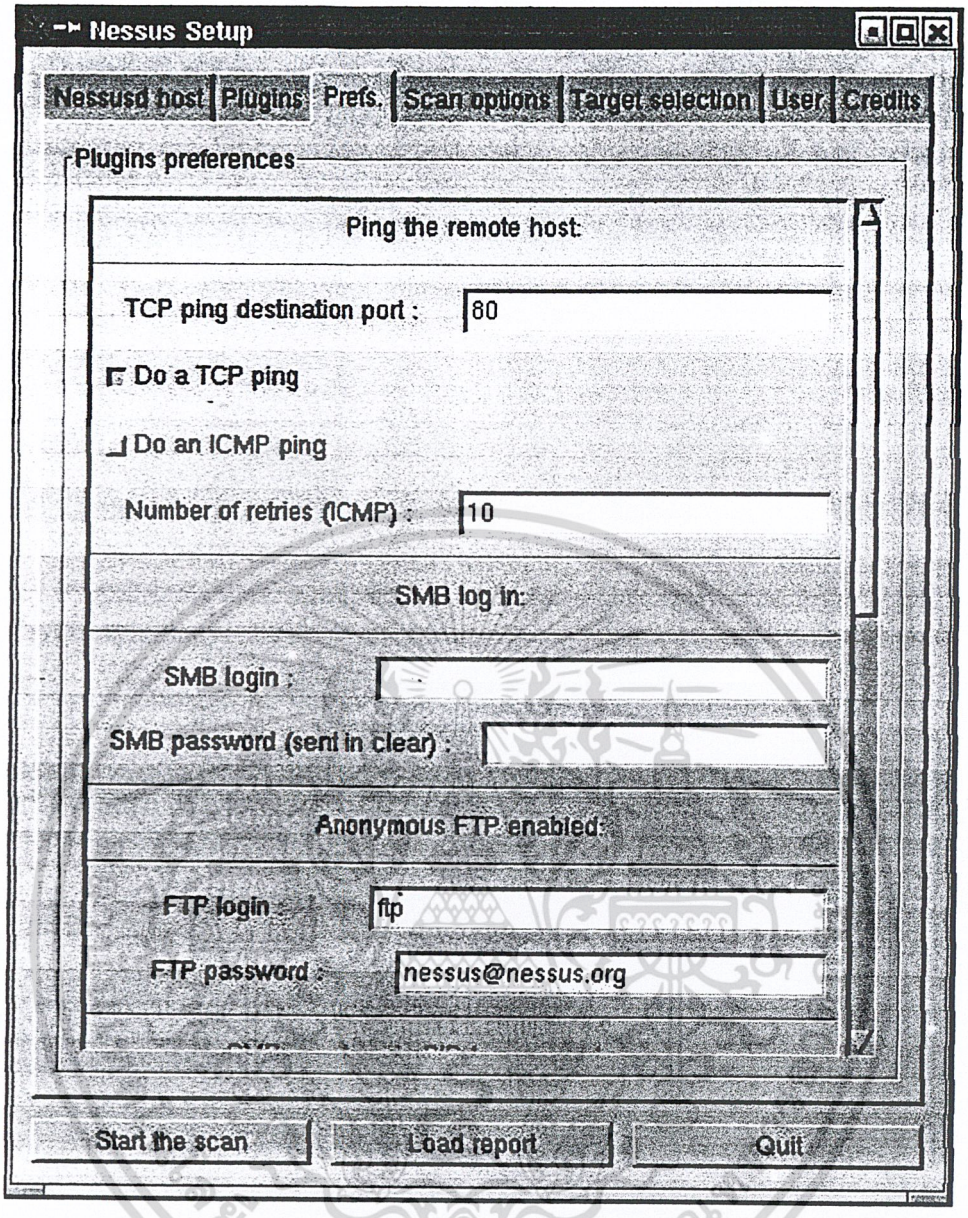
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.16 แสดงรายละเอียดของ plugin ที่ได้เลือกไว้

จากรูปที่ 2.15 เมื่อผู้ใช้ทำการคลิกไปที่ Anonymous FTP enable โปรแกรมจะแสดงรายละเอียดเพิ่มเติมของทางเลือกนี้ ดังแสดงในรูปที่ 2.16 ซึ่งอธิบายไว้ ว่า หากองค์กรใดไม่มีความจำเป็นต้องใช้ Anonymous FTP เพื่อแชร์ข้อมูลกับผู้อื่นที่มาจากภายนอกองค์กร ก็ควรปิดขีดความสามารถนี้ทิ้งไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

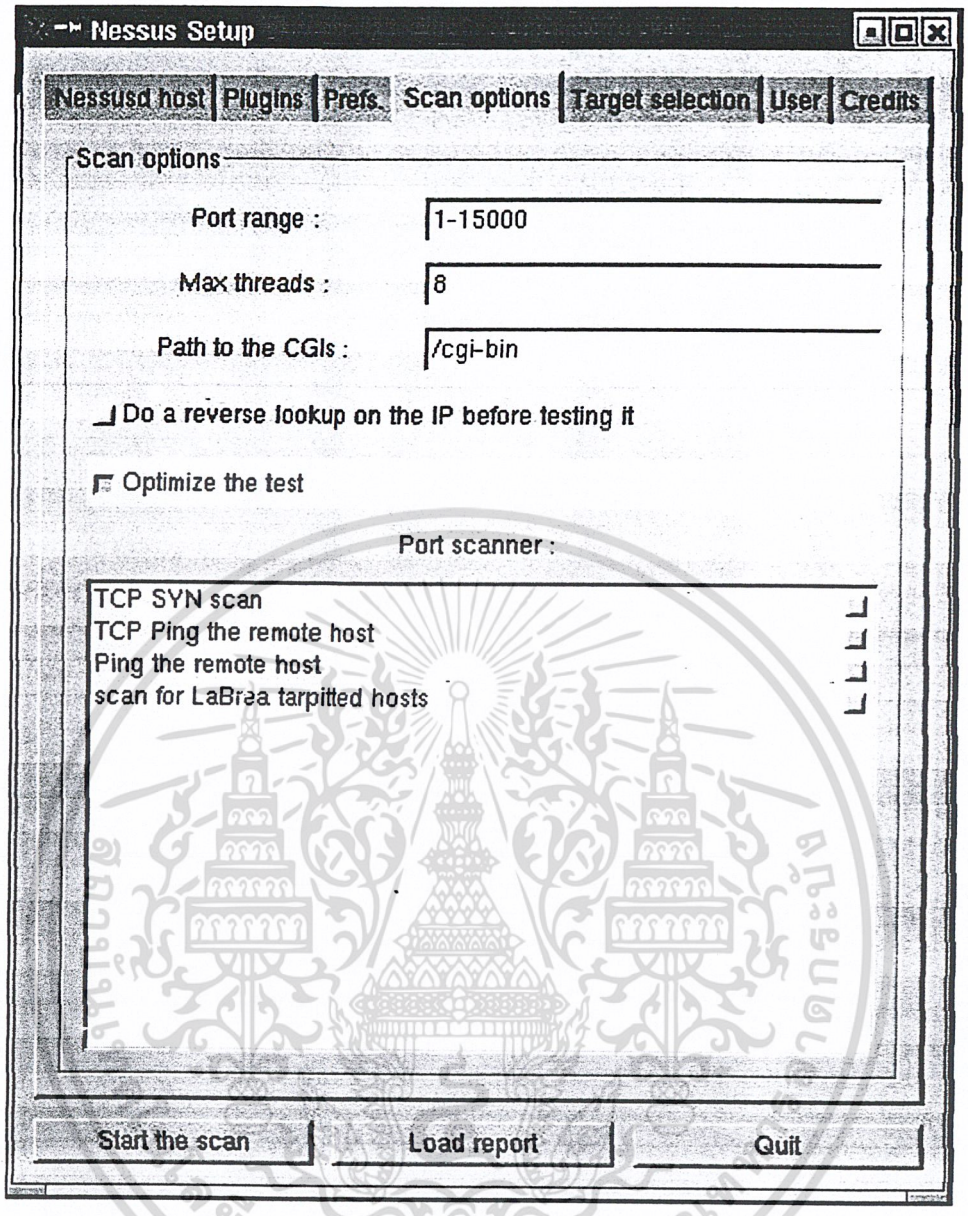


รูปที่ 2.17 แสดงหมวด Plugin Preference ของโปรแกรม

จากรูปที่ 2.17 ในหมวด Plugin Preference ผู้ใช้สามารถกำหนดค่า option ต่างๆของ plugin ที่ได้เลือกไว้ (ในหมวด plugin) เช่น ในการ ping เครื่องในเน็ตเวิร์คผู้ใช้อาจเลือกวิธี ping โดยใช้โปรโตคอล TCP หรือ โปรโตคอล ICMP ก็ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

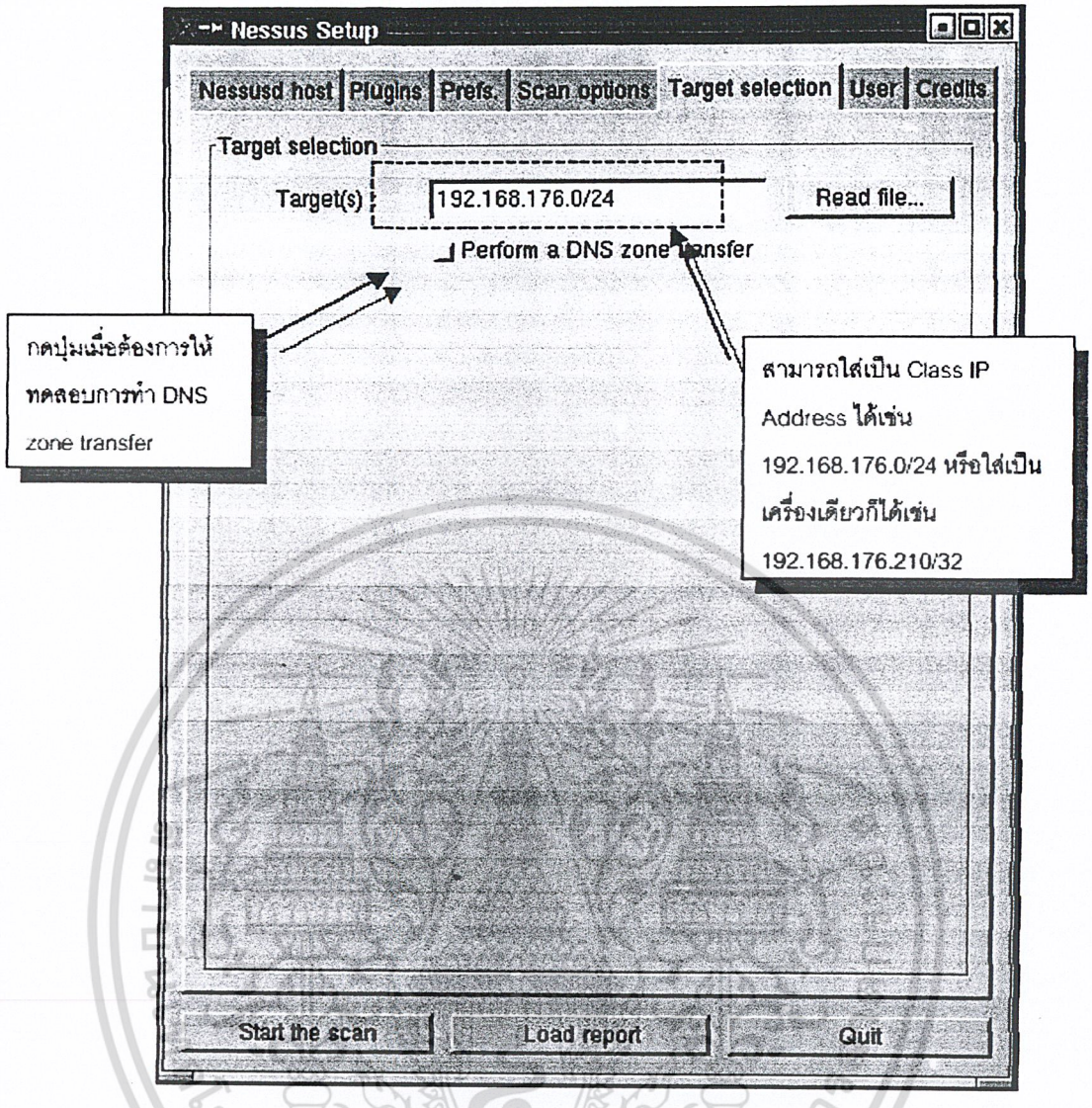


รูปที่ 2.18 แสดงหมวด Scan options ของโปรแกรม

จากรูปที่ 2.18 ในหมวด Scan option นี้ผู้ใช้สามารถกำหนดรายละเอียดของการสแกน เช่น ค่าของพอร์ตที่จะทำการสแกน (ในรูปพอร์ตที่จะถูกสแกนมีค่าระหว่าง 1 - 15,000) การสแกนสามารถทำพร้อมๆกันได้ 8 เครื่อง(หรือ 8 โปรเซส) ไดเรคตอรี่ซึ่งเก็บซีจีไอสคริปต์ เป็นต้น โดยปกติการสแกนพอร์ตกระทำโดยโปรแกรม nmap ซึ่งเป็นโปแกรมที่ทำงานควบคู่เป็นอย่างดีกับ Nessus

61409

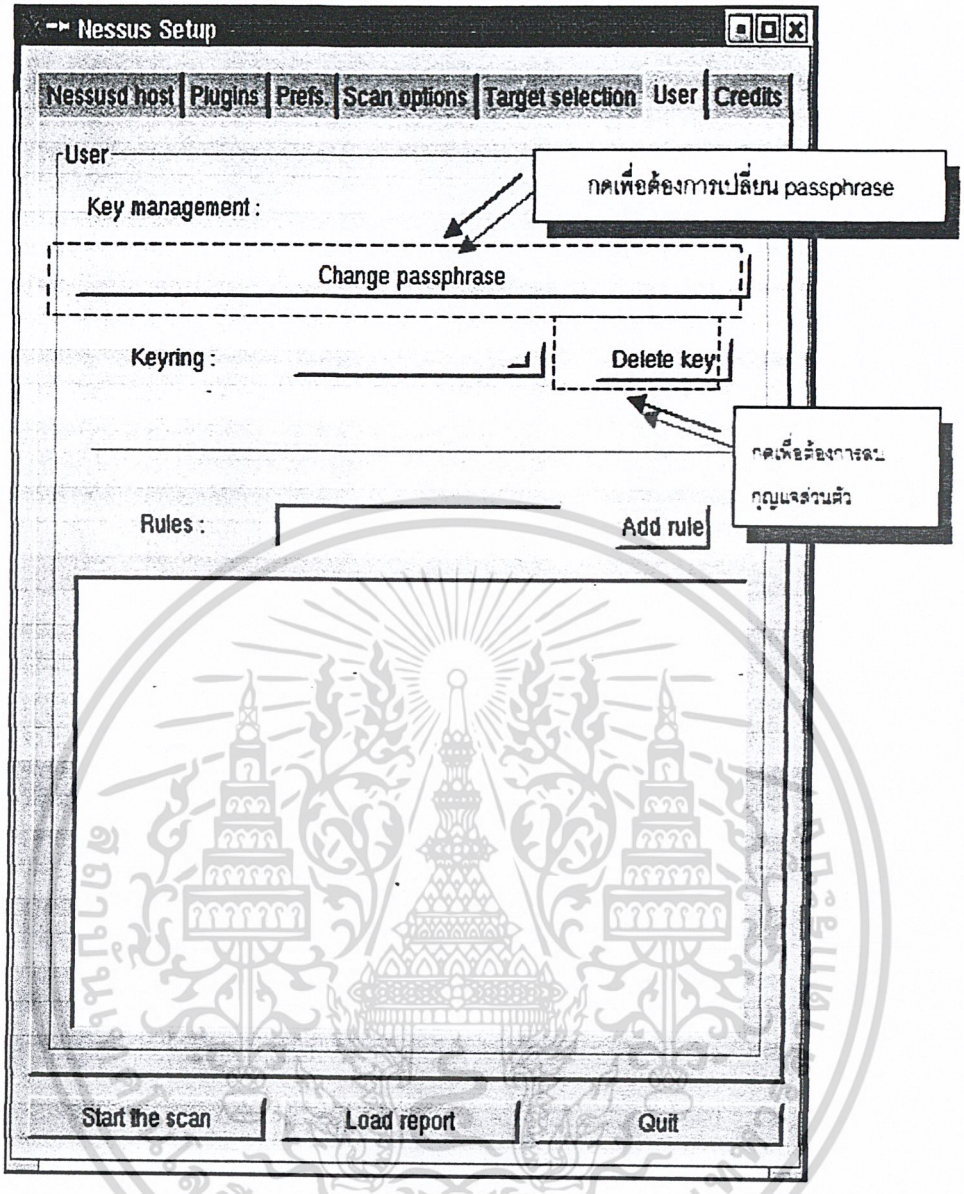
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.19 แสดงหมวด Target selection ของโปรแกรม

จากรูปที่ 2.19 ในหมวด Target selection ผู้ใช้สามารถเลือกเครื่องเป้าหมายที่จะได้รับการสแกนเพื่อตรวจสอบหาจุดอ่อน โดยสามารถเลือกเป็นเครื่องๆหนึ่ง หรือ เลือกทั้งซับเน็ตก็สามารถทำได้ เช่นในรูป 192.168.176.0/24 โดยอาจจะใส่เป็นหลายๆค่า และแต่ละค่าคั่นจากกันด้วยเครื่องหมาย ',' และหากต้องการตรวจสอบว่า DNS ที่จะถูกสแกนด้วยนั้นผู้อื่นสามารถทำ Zone transfer ได้หรือไม่ ก็ให้ติ๊กเลือกปุ่ม Preform a DNS zone transfer ด้วย (โดยปกติแล้วการทำ Zone transfer ถ้าจะให้มีความปลอดภัยสูงควรจะสามารถกระทำได้จาก เฉพาะเครื่องที่มีสิทธิ์เท่านั้น)

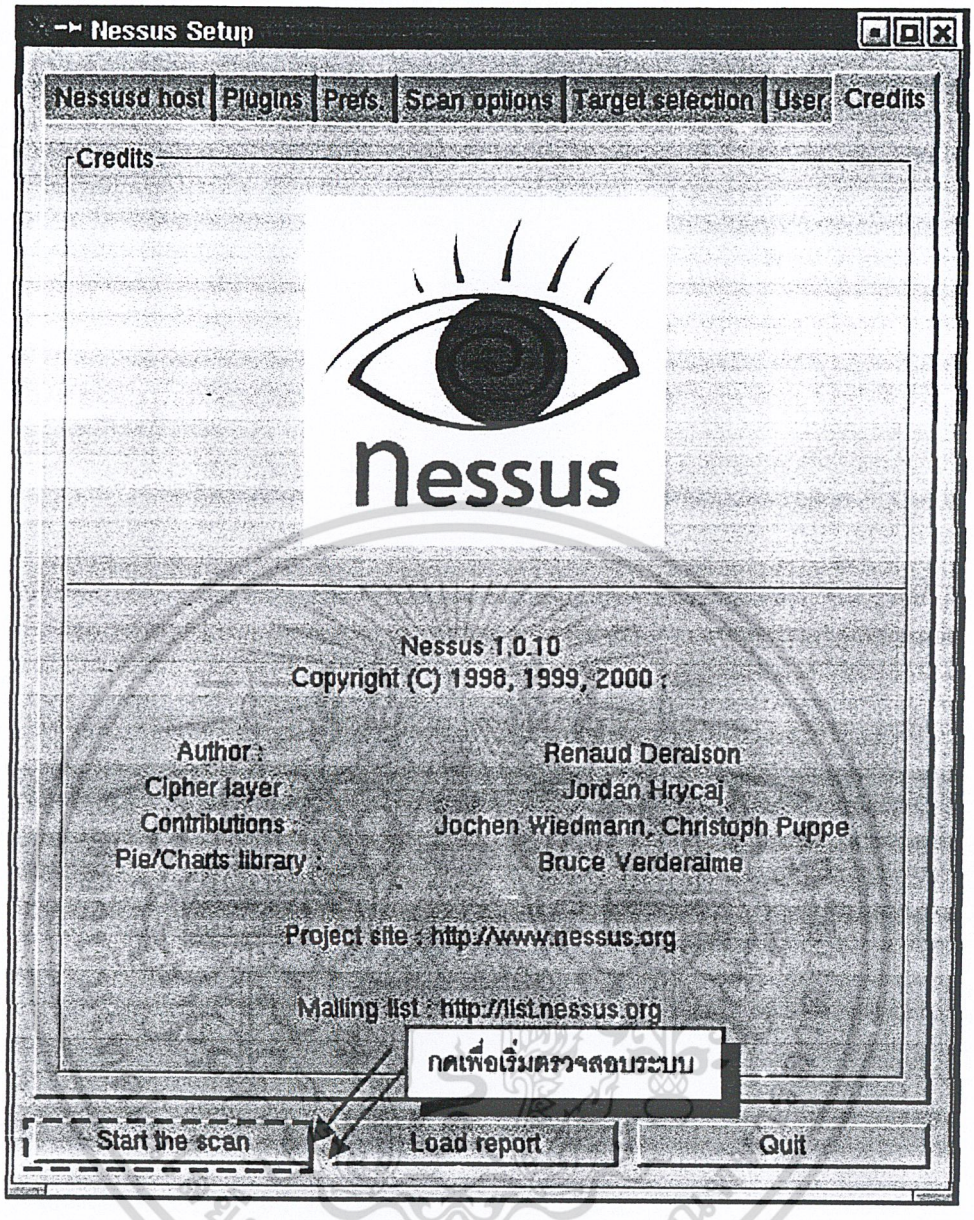
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.20 แสดงหมวด User ของโปรแกรม

จากรูปที่ 2.20 ในหมวด User ผู้ใช้สามารถเปลี่ยน passphrase ที่ใช้ในการเข้าโปรแกรมลูกข่าย nessus ได้ หรือแม้กระทั่งลบกุญแจส่วนตัวของตนเองทิ้งก็สามารถทำได้ รวมถึงการกำหนดสิทธิ์ในการสแกนเน็ตเวิร์คต่างๆเพิ่มเติมก็สามารถทำได้ (ตรงปุ่ม Add rule)

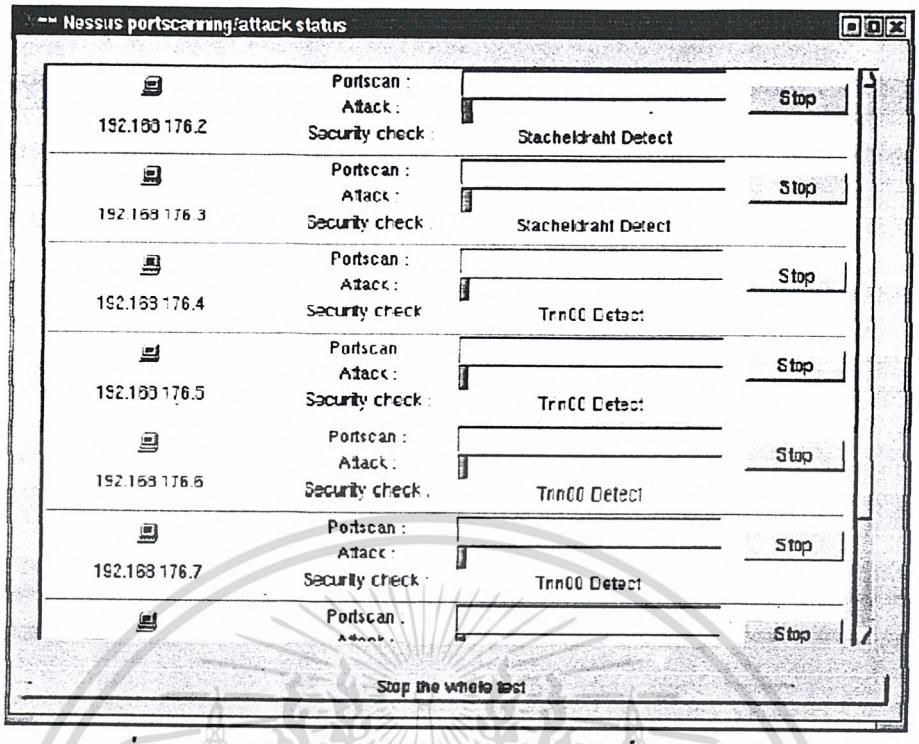
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



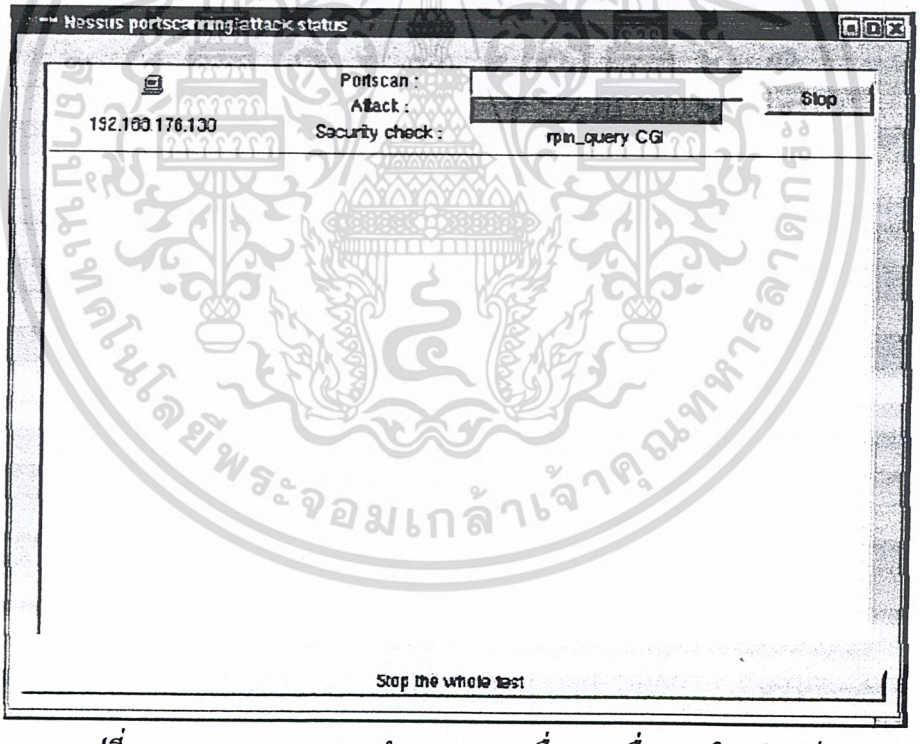
รูปที่ 2.21 แสดงหมวด Credits ของโปรแกรม

จากรูปที่ 2.21 ในหมวด Credits นี้ให้รายละเอียดเกี่ยวกับคณะผู้พัฒนาโปรแกรม Nessus เวอร์ชั่น รวมทั้งเว็ปไซต์ที่สามารถเข้าไปดูข้อมูลต่างๆเกี่ยวกับ Nessus ได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.22 แสดงสถานะขณะทำการสแกนหลายๆเครื่องภายในเน็ตเวิร์ค

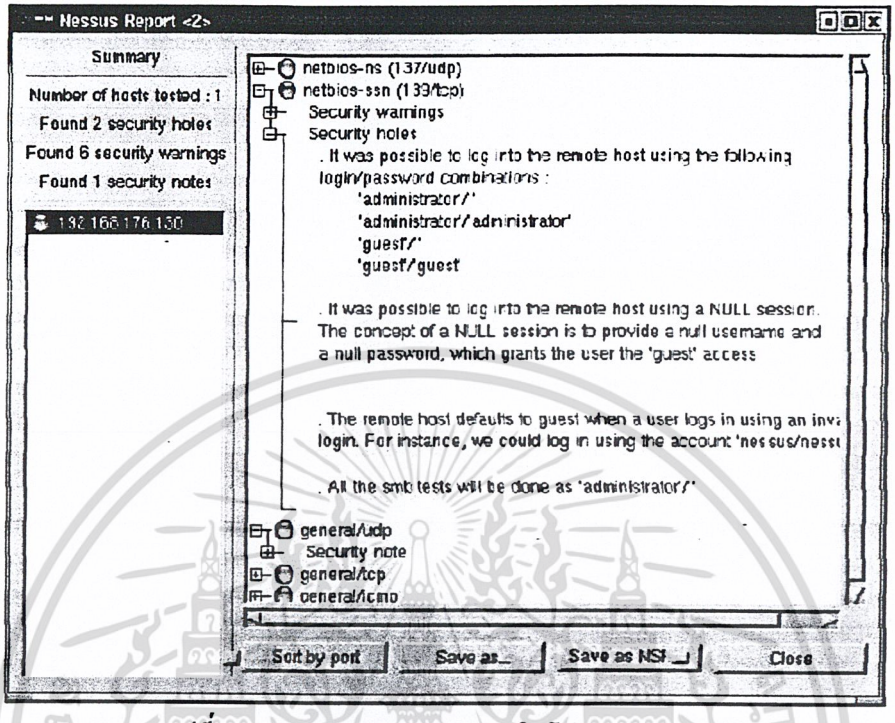


รูปที่ 2.23 แสดงสถานะขณะทำการสแกนเครื่อง 1 เครื่องภายในเน็ตเวิร์ค

รูปที่ 2.21 เป็นการสแกนเน็ตเวิร์คหลายๆเครื่องพร้อมๆกันตามค่า IP Address ที่ได้ตั้งค่าเอาไว้ในหมวด Target selection ส่วนรูปที่ 2.22 เป็นการสแกนเครื่องเพียงเครื่องเดียวที่ IP Address 192.168.176.130 ณ ที่เวลาใดเวลาหนึ่งในระหว่างที่ทำการสแกนผู้ใช้สามารถหยุดการสแกนลงได้โดยการ

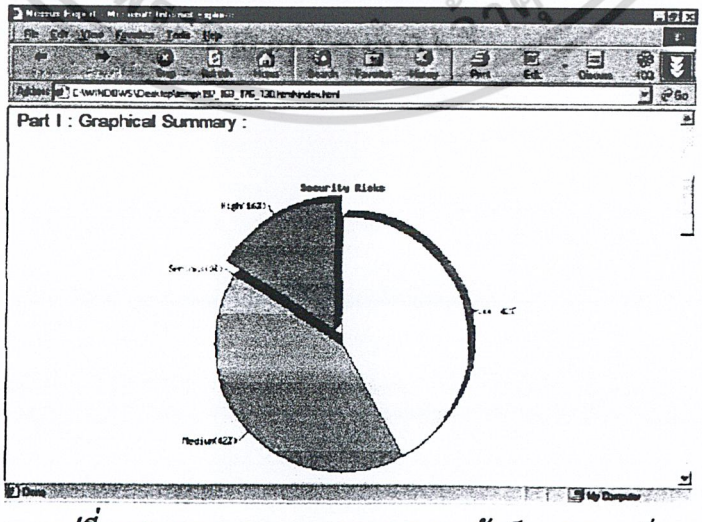
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กดปุ่ม Stop ทางด้านขวามือ หรือแม้กระทั่งหยุดการสแกนทั้งหมดเลยก็สามารถทำได้โดยการกดปุ่มStop the whole test ทางด้านล่างของหน้าต่าง



รูปที่ 2.24 แสดงผลจากการสแกนในโปรแกรม nessus

รูปที่ 2.23 แสดงผลที่ได้จากการสแกนเครื่อง 192.168.176.130 หน้าต่างทางด้านซ้ายแสดงประเภทของปัญหาความปลอดภัยที่พบ (Security holes,Security warnings และ Security note) ส่วนหน้าต่างทางด้านขวามือ เมื่อคลิกลงไปที่แต่ละวงกลม ผู้อ่านก็จะได้รายละเอียดและคำแนะนำเพิ่มเติมเกี่ยวกับปัญหานั้นๆ



รูปที่ 2.25 แสดงผลการสแกนออกทางหน้าเว็บบราวเซอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

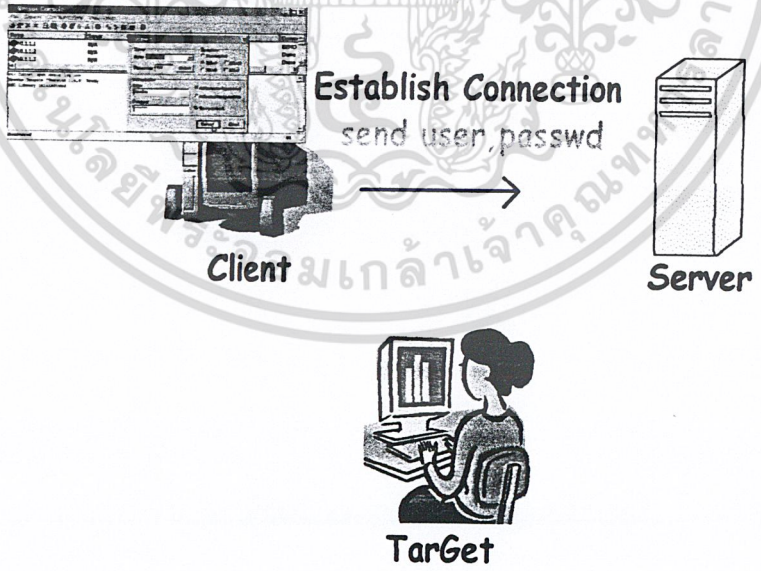
รูปที่ 2.24 เป็นการนำผลที่ได้จากการสแกนเครื่อง 192.168.176.130 โดย Nessus ซึ่งจะสร้างผลลัพธ์ที่ได้ออกมาเป็นไฟล์ index.html มาแสดงบนเว็บบราวเซอร์ กราฟพายชาร์ตแสดงให้เห็นถึงประเภทของความเสี่ยงแบบ Low, Medium, High และ Serious โดยคำนวณเป็นเปอร์เซ็นต์ ซึ่งจะมีประโยชน์ของการเห็นรูปรวมของปัญหาทั้งหมดในเน็ตเวิร์คของเรา อันจะนำไปสู่การแก้ไขปัญหาในภายภาคหน้าต่อไป

2.5 Plugin

Plugin เป็นหัวใจสำคัญของ Nessus เนื่องจากเป็นที่เก็บชุดโปรแกรมสำหรับตรวจสอบข้อบกพร่องต่างๆไม่ว่าจะเป็น การกระทำ backdoor, Denial Of Service (DOS), ขโมยสิทธิ์ root, และอื่นๆ ชุดโปรแกรมตรวจสอบเหล่านี้จะถูกเก็บเป็นหมวดหมู่ปกติแล้วจะอยู่ในไดเรกทอรี /usr/local/lib/nessus/plugin โดยชุดโปรแกรมเหล่านี้ถูกพัฒนาขึ้นมาด้วยภาษา NASL (Nessus Attack ScriptingLanguage)ผู้อ่านสามารถอ่านวิธีการเขียนโปรแกรมภาษานี้ได้จาก http://www.nessus.org/doc/nasl.html และสามารถหาชุดโปรแกรม Plugin ใหม่ๆ เพื่อใช้ทดสอบระบบของเราได้ที่ http://www.nessus.org ในหัวข้อ Plugins หรือที่ http://cgi.nessus.org/plugins/

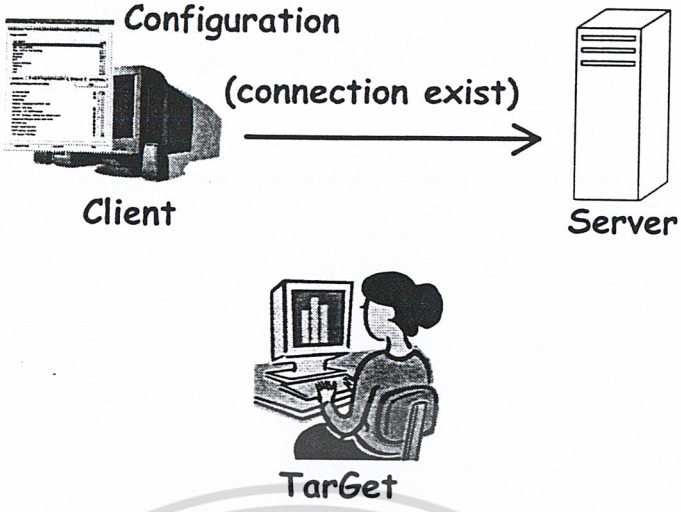
2.6 โครงสร้างการทำงานของ Nessus

เมื่อพิจารณาที่การทำงานของ Nessus ซึ่งมีลักษณะการทำงานร่วมกันของ ส่วนแม่ข่าย (Server) และส่วนลูกข่าย (Client) จะเห็นว่ามีโครงสร้างการทำงานเป็นดังรูปที่ 2.25

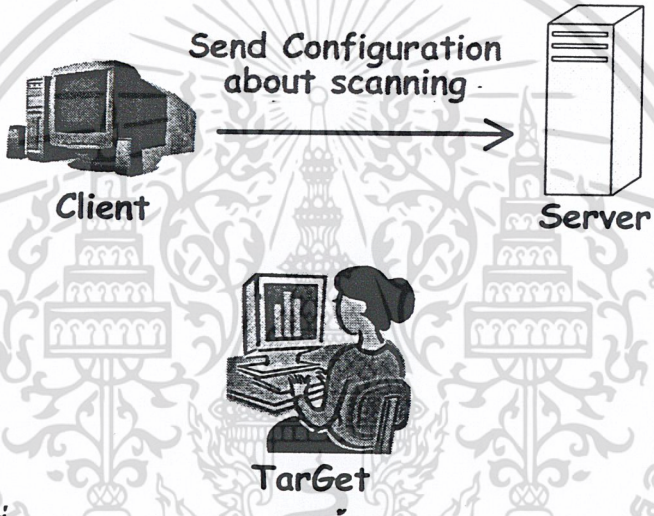


รูปที่ 2.26(ก.) แสดงการสถาปนาการเชื่อมต่อระหว่างส่วนลูกข่ายกับแม่ข่าย

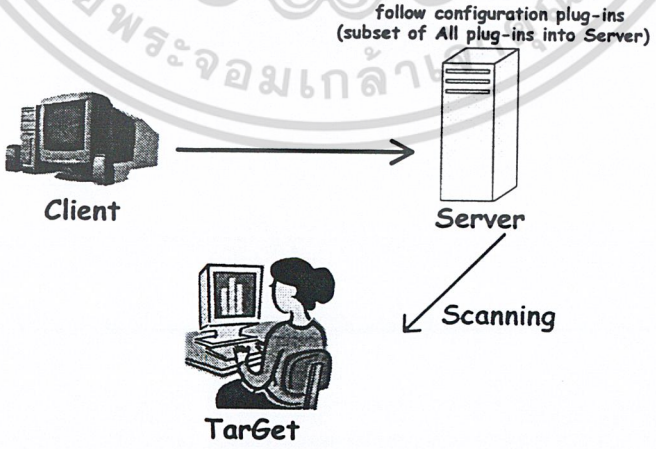
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.26(ข.) แสดงการปรับตั้งรูปแบบการสแกนโดยส่วนลูกข่ายเพื่อส่งให้แม่ข่าย

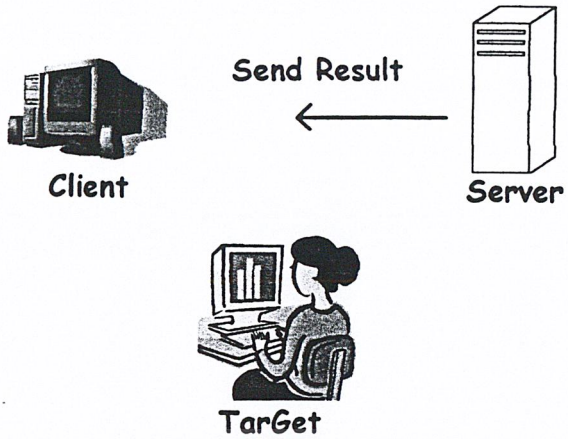


รูปที่ 2.26(ค.) แสดงการส่งการปรับตั้งรูปแบบการสแกนให้ส่วนแม่ข่าย



รูปที่ 2.26(ง.) แสดงรูปส่วนแม่ข่ายจะสแกนตามที่ได้ปรับตั้งรูปแบบการสแกนไว้โดยส่วนลูกข่าย

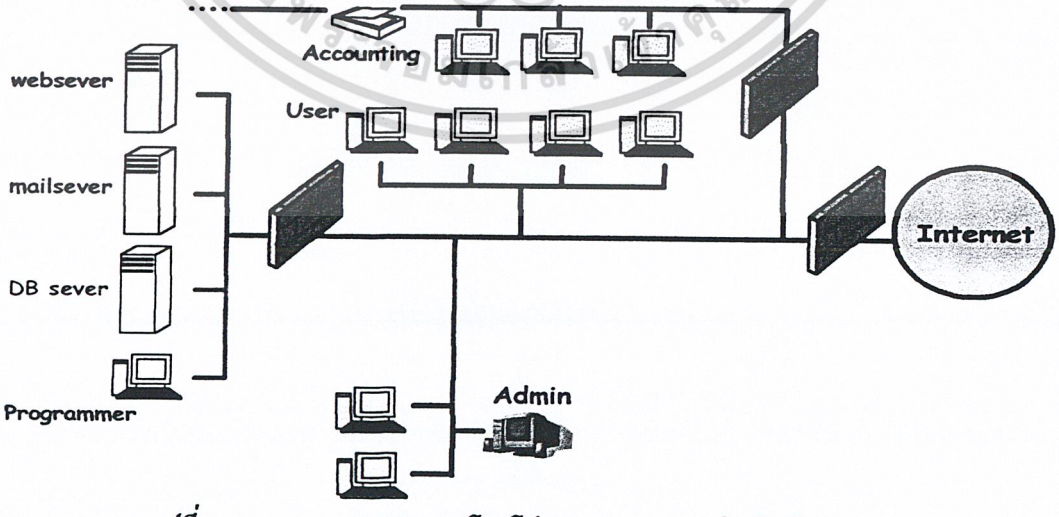
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.26(ฉ.) เมื่อส่วนแม่ข่ายได้ผลการสแกนแล้วก็จะส่งผลลัพธ์ให้กับส่วนลูกข่าย

รูปที่ 2.26(ฉ.) แสดงส่วนลูกข่ายจะนำผลลัพธ์ที่ได้รับมาแสดงผลในรูปแบบต่างๆ

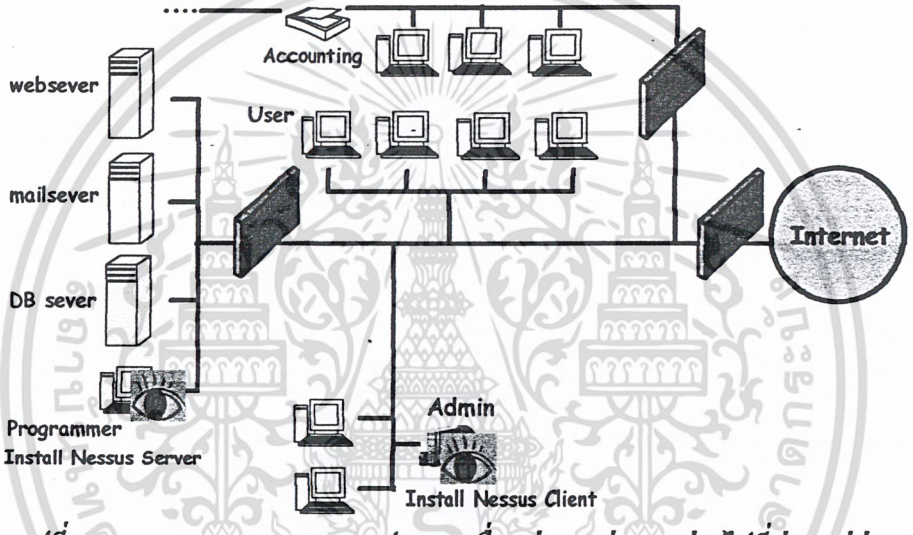
ด้วยโครงสร้างการทำงานร่วมกันของ ส่วนแม่ข่าย และส่วนลูกข่ายนี้ทำให้เกิดข้อได้เปรียบในเรื่องของมุมมองการสแกนหาช่องโหว่ของระบบโดยสามารถแสดงให้เห็นได้ดังรูปที่ 2.26



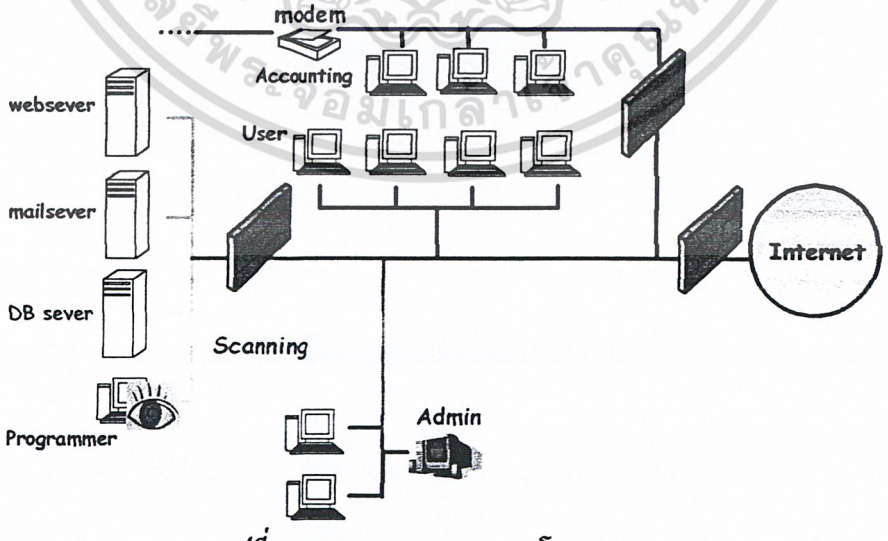
รูปที่ 2.27 (ก.) แสดงการสแกนโดยโปรแกรมสแกนหาช่องโหว่แบบปกติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 2.26 (ก.) จะเห็นว่าการสแกนจะทำการสแกนจากเครื่องผู้ดูแลระบบ (Admin) ได้ที่เครือข่ายที่ผู้ดูแลได้ดูแลอยู่ ซึ่งจะเห็นได้ว่าการสแกนจะต้องผ่านไฟล์วอร์ล ซึ่งอาจทำให้การสแกนบางรูปแบบไม่สามารถได้ผลรับตามต้องการ อาจเนื่องมาจากการปรับตั้งค่ากำหนดของไฟล์วอร์ล แม้จะสามารถปรับตั้งค่ากำหนดให้สามารถสแกนได้ในครั้งนั้นๆ แต่หากเครือข่ายมีขนาดใหญ่และมีส่วนย่อยซับซ้อนก็อาจทำให้เกิดความยุ่งยากได้ซึ่งรูปที่ 2.26 (ข.) และรูปที่ 2.26 (ค.) เป็นการแสดงการทำงานของ Nessus ซึ่งจะทำให้ได้มุมมองที่ต่างออกไปได้อย่างไรก็ตามจากโครงสร้างการทำงานที่แสดงในรูปทั้งสองก็จะเห็นข้อด้อยของ Nessus ที่ส่วนแม่ข่ายสามารถทำงานได้เฉพาะในระบบยูนิกส์เท่านั้นทำให้เกิดข้อจำกัดในการสแกนขึ้นเนื่องจากหากบางเครือข่ายย่อยมีแต่ระบบปฏิบัติการวินโดว์ซึ่งแสดงดังรูปที่ 2.26(ง.)

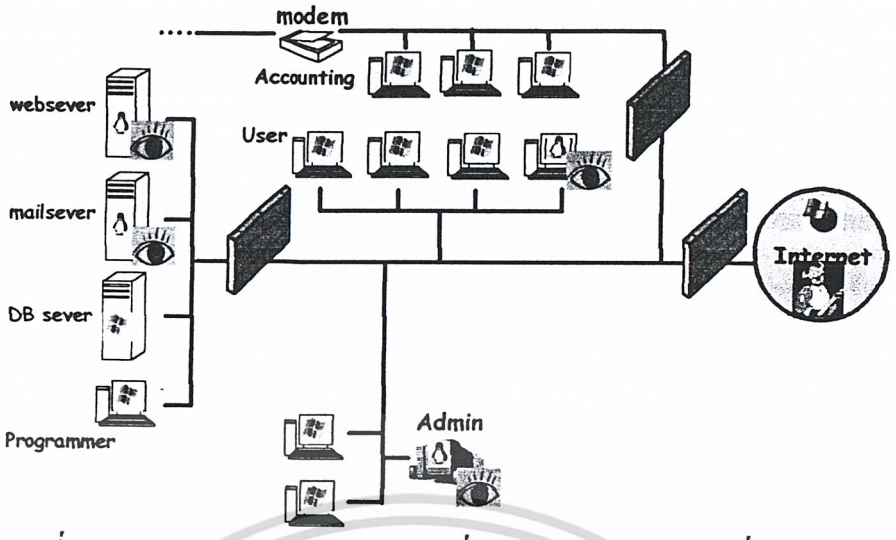


รูปที่ 2.27 (ข.) แสดงการสแกนสถาปนาการเชื่อมต่อจากส่วนลูกข่ายไปที่ส่วนแม่ข่าย



รูปที่ 2.27 (ค.) แสดงการสแกนโดย Nessus

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.27 (ง) เป็นการสแกนสถาปนาการเชื่อมต่อจากส่วนลูกข่ายไปที่ส่วนแม่ข่าย

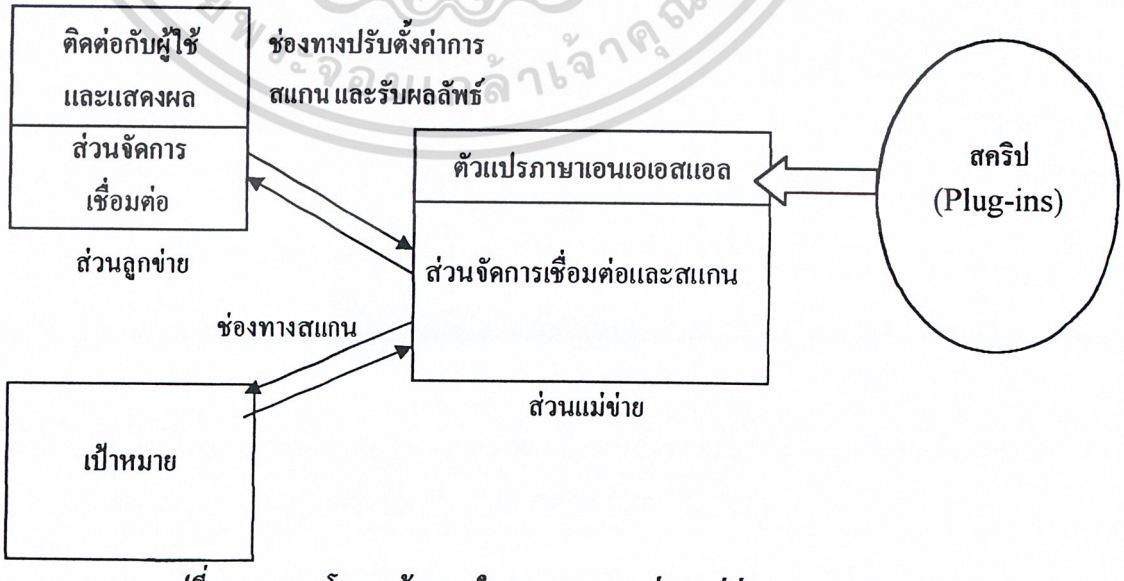
2.7 โครงสร้างของ Nessus

โครงสร้างส่วนลูกข่ายของ Nessus แบ่งย่อยได้เป็น 2 ส่วนดังนี้

- ส่วนติดต่อกับผู้ใช้เพื่อปรับตั้งค่าการทำงานและแสดงผลการทำงาน
- ส่วนที่จัดการเชื่อมต่อกับส่วนแม่ข่าย

โครงสร้างส่วนแม่ข่ายของ Nessus ซึ่งเป็นส่วนที่เราสนใจนั้นจะแบ่งได้เป็น 2 ส่วนดังนี้

- ส่วนที่จัดการเชื่อมต่อกับส่วนลูกข่ายและทำการสแกน
- ส่วนแปลภาษาเอนเอเอสแอลที่แปลภาษาเอนเอเอสแอลที่เลือกปรับตั้งค่ามาจากส่วนลูกข่าย



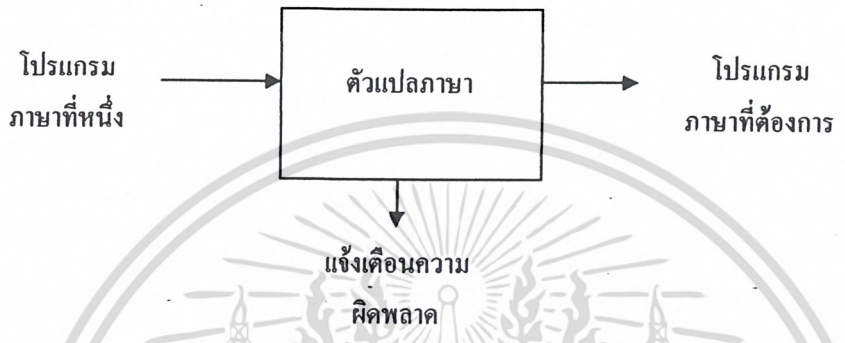
รูปที่ 2.28 แสดงโครงสร้างภายในของ Nessus ส่วนแม่ข่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
เครื่องมือสร้างตัวแปลภาษา

3.1 ตัวแปลภาษา คืออะไร

ตัวแปลภาษาเป็นโปรแกรมที่อ่านโปรแกรมที่เขียนในภาษาหนึ่ง (source language) และแปลงไปเป็นโปรแกรมที่เหมือนกันในภาษาอื่น (target language) ดังแสดงในรูป 3.1



รูปที่ 3.1 แสดงลักษณะการแปลภาษา

3.2 การสร้างตัวแปลภาษา

การสร้างตัวแปรภาษานั้นคล้ายกับการเขียนโปรแกรมทั่วไป สามารถใช้ซอฟต์แวร์เป็นเครื่องมือช่วยซึ่งเลคและเยค (Lex&Yacc)เป็นเครื่องมือที่ช่วยในการเขียนโปรแกรมแปลภาษา โดยจะมีงานหลักอยู่สองอย่างที่ต้องทำกับ ภาษาที่ป้อนเข้ามาเพื่อทำการแปล คือ ตัดแบ่งภาษาที่ป้อนเข้ามาเป็นหน่วยคำ และค้นหาความสัมพันธ์ระหว่างหน่วยคำ สำหรับโปรแกรมค้นหาข้อความในเอกสาร หน่วยคำน่าจะเป็นบรรทัดของเอกสาร และต้องการแยกความแตกต่างของบรรทัดที่มีข้อความที่ต้องการค้นหากับบรรทัดที่ไม่มีข้อความ สำหรับในโปรแกรมภาษาซี หน่วยคำเป็น ชื่อตัวแปร, สตริง, คอนสแทรน, โอเปอเรเตอร์ เป็นต้น ซึ่งจะถูกแยกเป็นหน่วยคำ ซึ่งจะถูกเรียกว่า 'โทเคน' ซึ่งกระบวนการดังกล่าวเรียนว่า เลคซิเคล อานาไลเซอร์ หรือ เลคซิ่ง ซึ่งเลคจะช่วยในการทำการอธิบายว่าโทเคนที่เป็นไปได้มีอะไรบ้างและสร้างออกมาเป็นภาษาซีให้ ซึ่งสิ่งที่ได้ออกมาจะถูกเรียกว่า เลคเซอร์ หรือสแกนเนอร์ ซึ่งสิ่งนี้สามารถระบุโทเคนได้ คำอธิบายที่เราให้กับเลคนั้นเรียกว่า เลคสเป็กสิฟิก ซึ่งก็คือ regular expression

ดังนั้นสิ่งที่นำเข้ามาจะถูกแบ่งเป็นโทเคนนั้น โปรแกรมมักจะต้องการสร้างความสัมพันธ์ระหว่างโทเคน ใน ตัวแปรภาษาซีนั้นจะค้นหา เอ็กซ์เพรสชั่น, สเตทเมนท์, การประกาศตัวแปร, บล็อก และฟังก์ชั่นในโปรแกรม ซึ่งการกระทำนี้เรียนว่า พาซซิ่ง และกฎที่ใช้กำหนดความสัมพันธ์ให้โปรแกรมเข้าใจได้คือ แกรมม่า ซึ่งเยคจะรับแกรมม่าที่รัดกุมและสร้างโปรแกรมภาษาซีที่สามารถ พาซ แกรมม่าได้ เยคพาเซอร์จะค้นหาว่าลำดับของโทเคนตรงกับกฎที่ให้ไว้ในแกรมม่าหรือไม่ และนอกจากนี้ยังตรวจสอบการสร้างประโยคที่ไม่ถูกต้องถ้าภาษาที่นำเข้ามาไม่ตรงกับกฎใดๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 ตัวอย่างการแยกคำด้วยเลค

ตัวอย่างนี้เป็นโปรแกรมที่ใช้แยกความแตกต่างชนิดของคำในภาษาอังกฤษ โดยจะเริ่มจากการระบุ parts of speech (noun, verb, etc.)

โดยจะเริ่มจากกลุ่มของคำกิริยาที่ต้องการแยกคำดังนี้

is am are were was be being been do does did

will would should can could has have had go

ตัวอย่างที่ 3.1 แสดงการค้นหาคำกิริยา

```
%{
        //นี่เป็นตัวอย่างการค้นหาคำกิริยา
%}
%%
[/t]+   /*      ไม่สนในช่องว่าง      */;
is |
am |
are |
were |
was |
be |
being |
been |
do |
does |
did |
will |
would |
should |
can |
could |
has |
have |
had |
go      { printf("%s: is a verb\n",yytext); }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
[a-zA-Z]+        { printf("%s: is a verb\n",yytext); }
.|\n       { ECHO; /* แม้ไม่ใส่บรรทัดนี้หากไม่พบ regular expression ใดๆ ก็จะ ECHO */     }
%%
main()
{
        yylex();
}
```

ด้านล่างจะแสดงว่าหากมีการคอมไฟล์และสั่งทำงาน

```
C:\>example3_1
did I have fun?
Did: is a verb
I: is not a verb
have: is a verb
fun: is not a verb
?
^C
C:\>
```

โดยสามารถอธิบายเป็นส่วนได้ดังนี้

```
%{
        //นี่เป็นตัวอย่างการค้นหาคำกิริยา
%}
```

ส่วนแรกเป็น ส่วนที่ใช้กำหนด (definition section) โดยเลคจะคัดลอกทุกสิ่งที่อยู่ระหว่าง "%{" และ "%}" ไปตรงๆ ที่ไฟล์ภาษาซีที่จะถูกสร้าง ดังนั้นก็น่าจะเขียนภาษาซีระหว่างส่วนนี้

%% เป็นเครื่องหมายที่แสดงการจบส่วนนี้

ส่วนต่อไปเป็นส่วนของกฎ ซึ่งกฎนั้นจะมีสองส่วนด้วยกันคือส่วนของรูปแบบและส่วนของสิ่งที่ต้องปฏิบัติ

ซึ่งจะแยกระหว่างสองส่วนนี้ด้วยช่องว่าง

```
[/t]+     /*        ไม่สนในช่องว่าง          */ ;
```

ในวงเล็บนั้นจะระบุว่าให้ตรงกับตัวอักษรทุกตัวที่อยู่ในวงเล็บนี้ ซึ่ง "\t" เป็นตัวอักษรแท็บ หรือ " " ใช้แสดงช่องว่าง "+" หมายถึงให้ตรงกับรูปแบบที่อยู่ด้านซ้ายมือของเครื่องหมายบวกอย่างน้อยให้มีหนึ่งตัวอักษร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อไปเป็นกลุ่มของกฎที่ใช้เครื่องหมาย "|" หมายถึงให้ใช้การปฏิบัติอย่างเดียวกันในรูปแบบต่อจากเครื่องหมายนี้

```
is |
am |
are |
were |
was |
be |
being |
been |
do |
does |
did |
will |
would |
should |
can |
could |
has |
have |
had |
go      { printf("%s: is a verb\n",yytext); }
```

หมายความว่าหากภาษาที่นำเข้ามาตรงกับคำกิริยาใดๆ ในนี้ก็ให้ปฏิบัติคำสั่ง printf โดยอาร์เรย์ yytext จะเก็บข้อความที่ตรงกับรูปแบบคำกิริยาที่กำหนด

ส่วนกฎสุดท้ายเป็น

```
[a-zA-Z]+       { printf("%s: is a verb\n",yytext); }
.|\n        { ECHO; /* แม้ไม่ใส่บรรทัดนี้หากไม่พบ regular expression ใดๆ ก็จะ ECHO */ }
```

โดย [a-zA-Z]+ เป็นการให้ตรงกับตัวอักษรอย่างน้อยหนึ่งตัวอักษร "-" จะมีความหมายพิเศษเมื่ออยู่ในวงเล็บ โดยเหมือนให้ a และ z เป็นขอบเขตของอักษรที่ต้องการ ดังนั้นก็จะหมายความว่าต้องการให้ตรงกับอักษร a ถึง z หรือ A ถึง Z อย่างน้อยหนึ่งตัวอักษร ส่วน.|\n นั้น "." คืออักษรใดๆหรืออักษรจบบรรทัด (Enter)

ส่วนจบส่วนของกฎนี้จะจำกัดโดยเครื่องหมาย %%

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในส่วนสุดท้าย เป็นส่วนโปรแกรมย่อย (user subroutines section)ซึ่งจะคัดลอกทุกสิ่งที่อยู่ระหว่าง "%{" และ "%}" ไปตรงๆ ที่ไฟล์ภาษาซีที่จะถูกสร้าง ดังนั้นก็น่าจะเขียนภาษาซีระหว่างส่วนนี้

```
main()
{
        yylex();
}
```

เลคเซอร์จะทำงานโดยเรียกโปรแกรมภาษาซีให้ทำงานโดยเรียก yylex()

ซึ่งโปรแกรมตัวอย่างนี้สามารถใช้งานได้กับ แฟลกซ์ (flex) ซึ่งเป็นโปรแกรมที่เป็น GPL ที่มีการใช้กันมากในระบบปฏิบัติการวินโดวส์โดยสามารถสั่งให้ทำงานได้โดยคำสั่งดังนี้

```
C:\>flex exemple3_1.l
C:\>cc lex.yy.c -o first.exe
```

ซึ่ง cc เป็นตัวแปรภาษาซี โดย "-o first.exe" คือให้ผลลัพธ์การทำงานได้เป็นชื่อไฟล์ first.exe หลังจากทำตามคำสั่งดังกล่าวก็สามารถสั่งทำงานได้ที่ไฟล์ first.exe

3.4 แกรมม่า

เยค(yacc)รับแกรมม่าที่ผู้ต้องการสร้างตัวแปลภาษาระบุและสร้างพาเซอร์ที่ตรวจหา "ประโยค" ที่มีอยู่ในแกรมม่า ซึ่งได้กล่าวไปแล้วว่าแกรมม่าเป็นลำดับของกฎที่พาเซอร์ใช้ในการหาความสัมพันธ์ในประโยคจากภาษาที่นำเข้ามา สำหรับตัวอย่างเป็นแกรมม่าที่อาจนำไปใช้ในการสร้างเครื่องคิดเลขได้

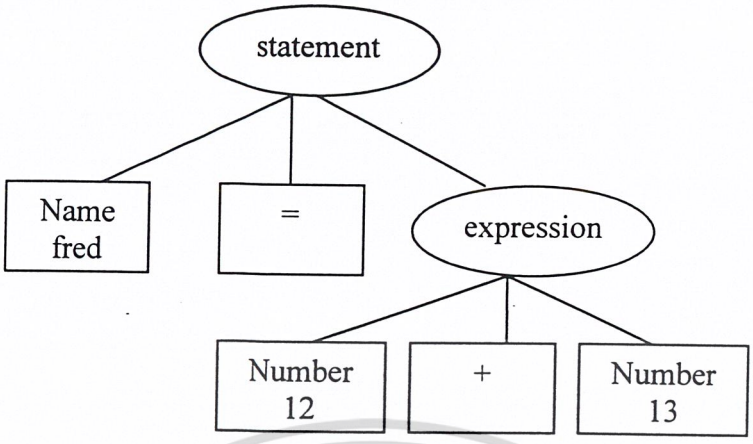
Statement -> NAME = expression

Expression -> NUMBER + NUMBER | NUMBER – NUMBER

เครื่องหมาย "|" หมายถึงเป็นไปได้ทั้งสองทางที่จะเป็นซีมบัล(symbol) เดียวกัน เช่นจากตัวอย่างจะหมายความว่า expression สามารถเป็นได้ทั้งการบวกและการลบ ซีมบัลที่อยู่ทางซ้ายของเครื่องหมาย -> จะถือว่าเป็น left-hand side ของกฎนั้นๆ ใช้คำย่อยว่า LHS, และซีมบัลที่อยู่ทางขวามือก็จะเป็น right-hand side ซึ่งก็คือ RHS ซึ่งการที่จะแสดงการพาซประโยคมักจะนำเสนอเป็นโครงสร้างที่มีรูปแบบเป็นต้นไม้ (tree) เช่นถ้าเราต้องการพาซโดยนำ ."fred = 12 + 13" เข้ามาทำการพาซด้วยแกรมม่านี้จะได้ผลดังรูปที่ 3.2 ซึ่งจะถือว่า "12 + 13" เป็น expression, และ "fred = expression" เป็น statement ซึ่งในความเป็นจริงแล้วเยคพาเซอร์ไม่สามารถสร้างโครงสร้างต้นไม้ได้เองผู้สร้างตัวแปรภาษาจะต้องสร้างมันด้วยตนเอง ซิลบัลนั้นในความเป็นจริงแล้วก็คือค่าที่นำเข้ามาจากเลคเซอร์ซึ่งเลคเซอร์จะรีเทอร์นเป็นเทอร์มินัล (terminal)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หรือโทเคนมาให้ซึ่งนำมารวมเป็น LHS ของกฎซึ่งก็คือนอนเทอร์มินัล (non-terminal)



รูปที่ 3.2 แสดงการพาซในรูปแบบโครงสร้างต้นไม้ (parse tree)

3.5 ค่าของซิมบัลและการปฏิบัติงาน

ทุกๆซิมบัลในเยคพาเซอร์จะมีค่าของตัวมันเอง ค่าเหล่านี้จะเพิ่มข้อมูลที่เป็นรายละเอียดให้กับซิมบัล ถ้าซิลบัลนั้นๆแสดงตัวเลข ค่าของมันก็ควรจะเป็นตัวเลข ถ้าซิมบัลแสดงตัวอักษรสตริงค่าของมันก็ควรจะเป็น pointer ที่ชี้ไปยังที่เก็บสตริงที่คัดลอกไว้ ซึ่งนอนเทอร์มินัลเองก็สามารถเก็บค่าเป็นค่าอะไรก็ได้ตามที่ต้องการซึ่งสร้างได้ตามหลักการของพาเซอร์ โดยการกำหนดค่า union typedef ที่เรียกว่า YYTYPE ซึ่งหากไม่มีการกำหนดค่าให้พาเซอร์จะมองว่าทุกๆค่าเป็นชนิด int เสมอดังตัวอย่างที่ 3.2 ที่แสดงการสร้างเครื่องคิดเลขจากพาเซอร์

อย่างไรก็ตามพาเซอร์จะสรุปกฎใดพาเซอร์จะกระทำคำสั่งภาษาซีที่ผู้ใช้กำหนดให้มีความสัมพันธ์กับกฎนั้นๆ ซึ่งที่การปฏิบัติงานนั้นสามารถอ้างถึงค่า RHS ได้โดยซิมบัล $1, $2, ..., และสามารถเปลี่ยนแปลงค่า LHS ได้โดยผ่าน $$ ได้

ตัวอย่างที่ 3.2 แสดงการสร้างเครื่องคิดเลขจากพาเซอร์

```
%token NAME NUMBER
%%
statement: NAME '=' expression
        | expression { printf("= %d\n", $1); }
        ;
expression: expression '+' NUMBER { $$ = $1 + $3; }
        | expression '-' NUMBER { $$ = $1 - $3; }
        | NUMBER        { $$ = $1; }
        ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากตัวอย่างนี้ $$ ก็จะเป็นการอ้างถึงซิมบัล expression ส่วน $1 เป็นซิมบัลตัวแรกของกฎ และ $2 เป็นเครื่องหมายแสดงว่าบวกหรือลบ ส่วน $3 คือ NUMBER ที่จะนำมากระทำด้วย ซึ่งจะเห็นว่าในตัวอย่างไม่ได้มีการอ้างถึง $2 เนื่องจากค่าที่จะนำไปปรับเปลี่ยนค่าของ $$ และมีการเปลี่ยนแปลงได้นั้นเป็น $1 และ $3

โดยหากต้องการให้พาเซอร์ทำงานได้นั้นเราจะต้องสร้างเลคเซอร์ที่จะแยกโทเคนออกมานำเข้าพาเซอร์ได้ ซึ่งพาเซอร์จะถือเป็นโปรแกรมในระดับสูงกว่าเลคเซอร์ซึ่งจะมีการเรียกโปรแกรม yylex() เมื่อต้องการโทเคน เพื่อให้เลคเซอร์ค้นหาโทเคนที่พาเซอร์สนใจ โดยเยคจะมีการระบุชื่อของโทเคนไว้ให้ในไฟล์ y.tab.h แล้วให้เลคเซอร์นำไปใช้ระบุโทเคนที่พาเซอร์ต้องการได้ ซึ่งลักษณะโปรแกรมเลคเซอร์จะเป็นดังนี้

```
%{
#include "y.tab.h"
extern int yylval;
%}
%%
[0-9]+ { yylval = atoi(yytext); return NUMBER; }
[\t] ; /* ไม่กระทำการใดๆกับช่องว่าง */
\n return 0;
. return yytext[0];
%%
```

จะเห็นว่าเลคเซอร์สามารถรีเทอร์นโทเคนให้พาเซอร์ ถ้าโทเคนไหนมีค่าที่สัมพันธ์กับโทเคนก็สามารถเก็บใน yylval ก่อนที่จะทำงานนั้นๆ ได้

การสั่งงานโปรแกรมเครื่องคิดเลขจากพาเซอร์ทำได้โดย

```
C:\>yacc -d example3_2.y        #จะทำการสร้าง y.tab.c และ y.tab.h
C:\>lex example3_2.l            #สร้าง lex.yy.c
C:\>cc -o example3_2.exe y.tab.c lex.yy.c
C:\>example3_2.exe
99+12
= 111
C:\>example3_2.exe
2+3-14+33
= 24
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
C:\>example3_2.exe
100 + -50
syntax error
```

ตัวอย่างนี้เป็นการทดสอบ กรณีถ้าแกรมม่าไม่ยอมรับภาษาที่นำเข้ามาทำการพาซ ก็จะแสดงข้อความ "syntax error" ให้ทราบ

ในการพัฒนาตัวแปรภาษาเอนเอเอสแอลนั้นใช้เครื่องมือ flex และ bison ซึ่งเป็น GNU เวอร์ชั่นของเลคและเยค ซึ่งสามารถใช้งานได้โดยไม่เสียค่าใช้จ่ายใดๆ

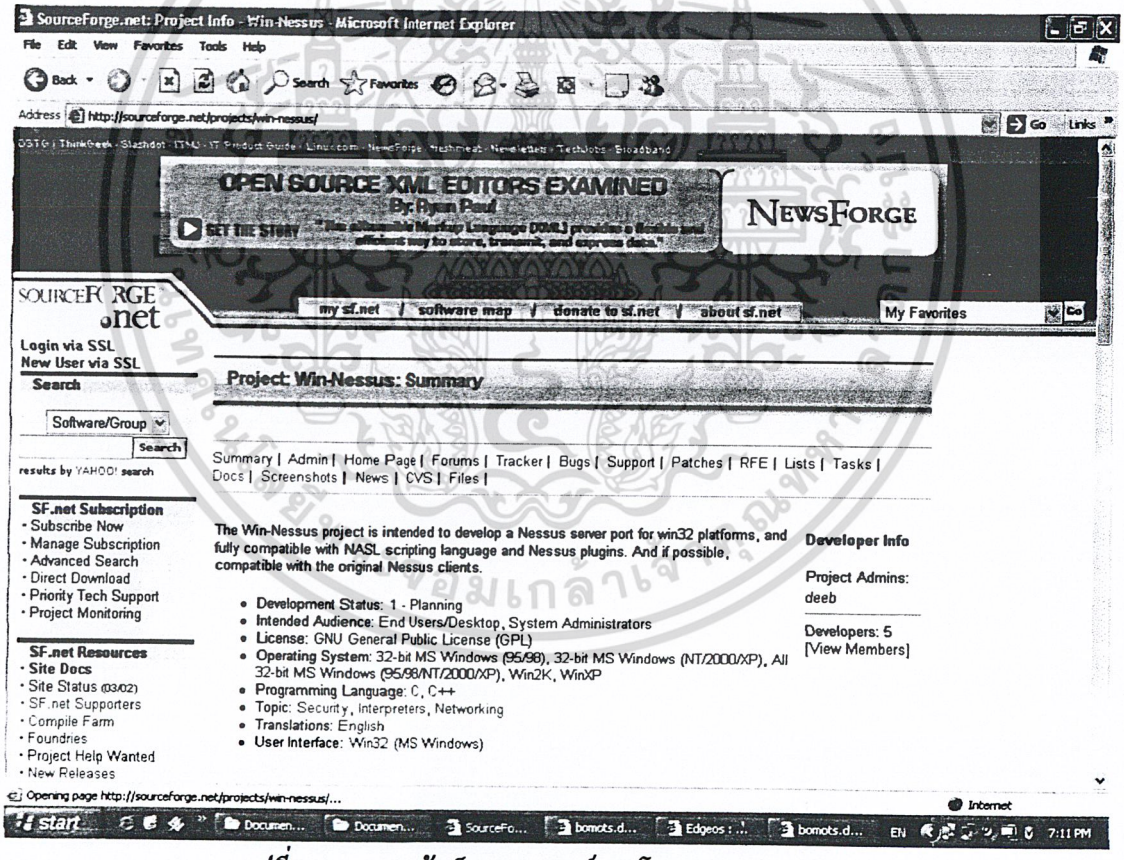
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4

การออกแบบและแนวทางการพัฒนา

4.1 การออกแบบและแนวทางการพัฒนาโปรแกรมตัวแปลภาษาเอนเอเอสแอล

การพัฒนาตัวแปลภาษาเอนเอเอสแอลนี้เป็นการพอร์ตตัวแปรภาษาที่ทำงานได้บนระบบยูนิกส์ให้สามารถทำงานบนระบบปฏิบัติการวินโดว์ได้ ซึ่งการพัฒนาตัวแปลภาษาเอนเอเอสแอลเป็นการพัฒนาเพื่อนำไปใช้ประโยชน์ต่อไปสำหรับการสร้าง Nessus ส่วนเซิร์ฟเวอร์ ซึ่งมีผู้ที่สนใจพัฒนาโปรแกรม Nessus ส่วนเซิร์ฟเวอร์อยู่แล้ว และได้ตั้งเป็นโครงการขึ้นที่ http://www.sourceforge.net โดยมีชื่อโครงการว่า "Win-nessus" เปิดให้ผู้ที่มีความสามารถและสนใจพัฒนาโปรแกรมนี้ทั่วโลกร่วมกันพัฒนา โดยผู้ควบคุมโครงงานนี้มีชื่อว่า นาย Ahmed El Deeb เป็นชาวอียิปต์ ในเบื้องต้นได้พยายามทำการติดต่อทางโทรศัพท์และอีเมล์กับนาย Ahmed El Deeb เพื่อเข้าร่วมทำการพัฒนาโปรแกรม Nessus ส่วนเซิร์ฟเวอร์

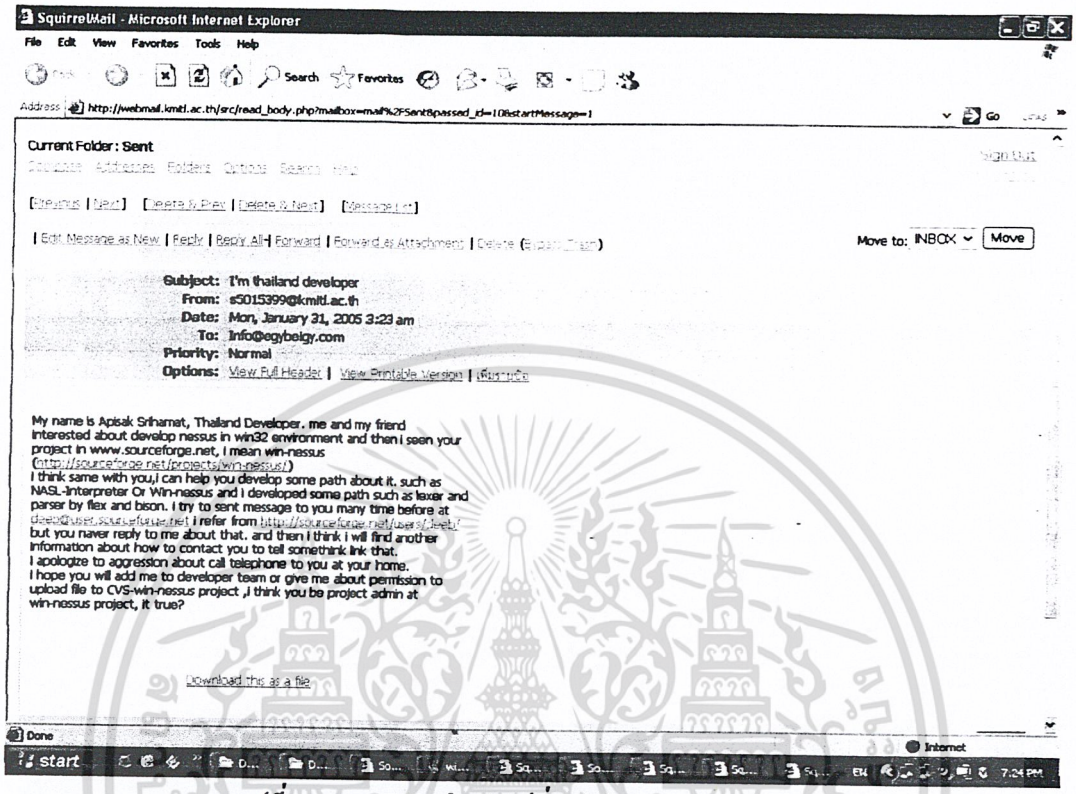


รูปที่ 4.1 แสดงหน้าเว็บบราวเซอร์ของโครงการ Win-nessus

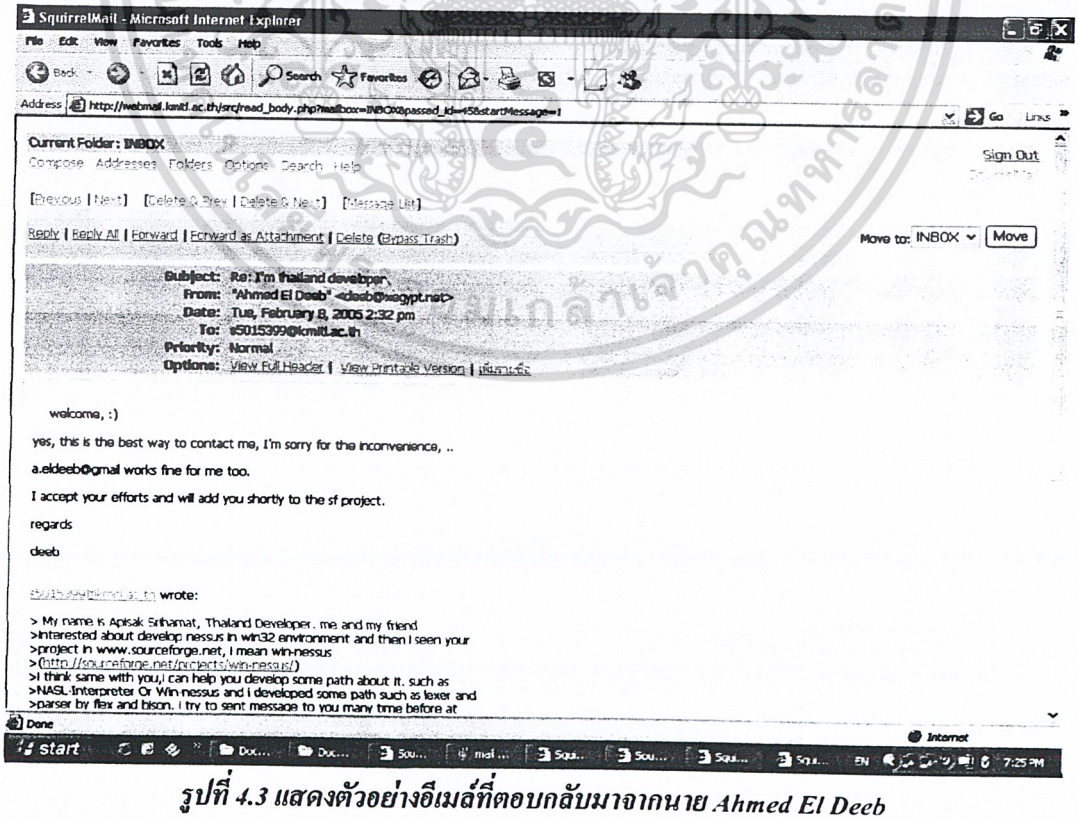
โดยในรูปที่ 4.2 แสดงตัวอย่างอีเมล์ที่ส่งให้นาย Ahmed El Deeb เพื่อแจ้งความต้องการเข้าร่วมพัฒนาโปรแกรมกับทีมงาน Win-nessus ซึ่งในเวลาต่อมาก็ได้รับการติดต่อกลับมาจากนาย Ahmed El Deeb ทางอีเมล์พร้อมกับได้รับสิทธิเป็นทีมพัฒนาและอัพโหลดไฟล์ตัวแปลภาษาเอนเอเอสแอลเข้าสู่ CVS

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เซิร์ฟเวอร์ ของ sourceforge.net ในชื่อโครงงานว่า Win-nessus ซึ่งอีเมล์ตอบกลับและหน้าเว็บบราวเซอร์แสดงผู้ร่วมพัฒนาก็จะแสดงดังรูปที่ 4.3 และรูปที่ 4.4 ตามลำดับ

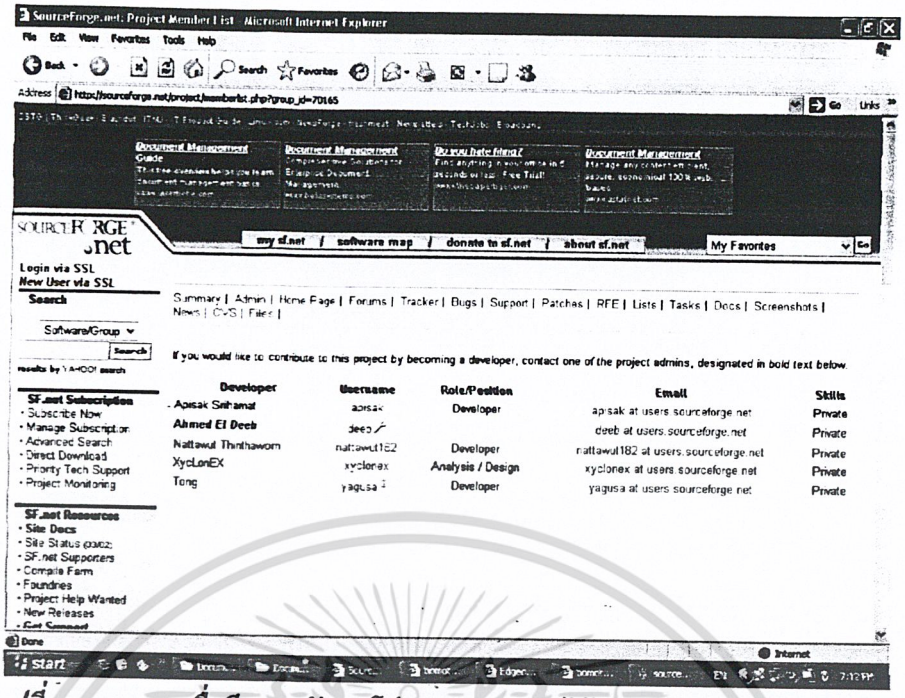


รูปที่ 4.2 แสดงตัวอย่างอีเมล์ที่ส่งให้นาย Ahmed El Deeb



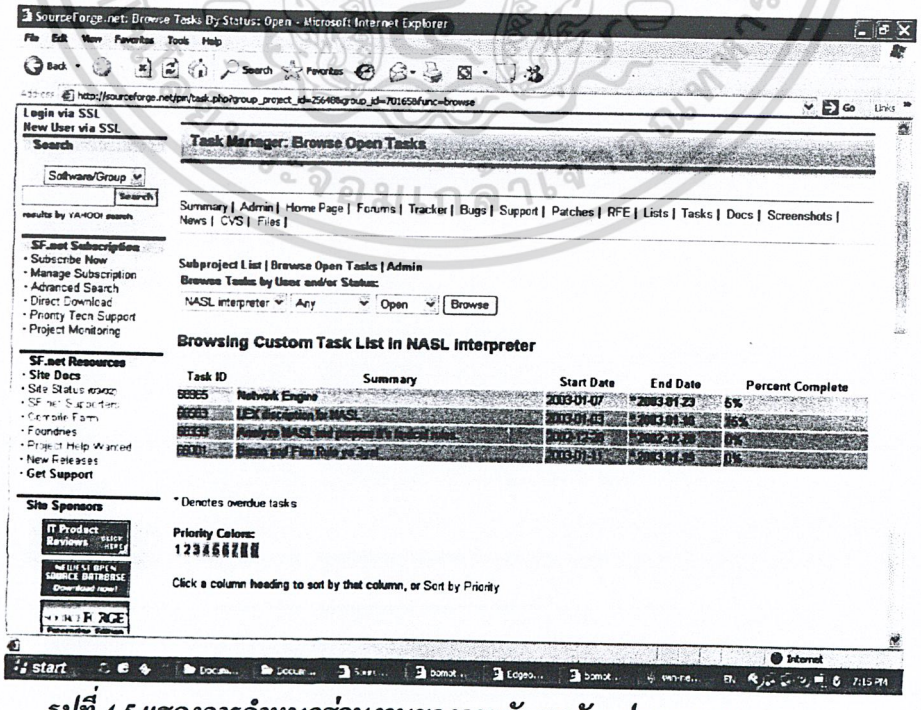
รูปที่ 4.3 แสดงตัวอย่างอีเมล์ที่ตอบกลับมาจากนาย Ahmed El Deeb

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.4 แสดงรายชื่อทีมงานพัฒนาโปรแกรมหลังจากได้ทำการติดต่อเข้าร่วมงานแล้ว

เนื่องจากการพัฒนาตัวแปลภาษาเอนเอเอสแอลนี้เป็นการพัฒนาร่วมกับโครงการ Win-nessus ที่จัดตั้งไว้โดยนาย Ahmed El Deeb ดังที่ได้กล่าวไปแล้ว ซึ่งแนวทางการพัฒนาตัวแปลภาษาเอนเอเอสแอลนี้ก็ได้ถูกออกแบบไว้แล้ว ดังนั้นในส่วนของการออกแบบการพัฒนาตัวแปลภาษาเอนเอเอสแอลนี้จะยึดหลักที่ถูกวางแนวทางไว้แล้วรูปที่4.5 แสดงการกำหนดส่วนงานของการพัฒนาตัวแปลภาษาเอนเอเอสแอลที่ออกแบบไว้แล้ว



รูปที่ 4.5 แสดงการกำหนดส่วนงานของการพัฒนาตัวแปลภาษาเอนเอเอสแอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 4.5 จะเห็นว่าส่วนงานของตัวแปลภาษาเอนเอเอสแอลนั้น ได้ถูกออกแบบไว้ให้มีส่วนงานทั้งหมด 4 ส่วนดังนี้

- วิเคราะห์ภาษาเอนเอเอสแอลและสร้างเลคสิเคอร์รูล (Lexical Rule)
- นำเลคมาบรรยายภาษาเอนเอเอสแอล
- นำเฟลกและไบสันมาสร้างตัวแปลภาษาเอนเอเอสแอล
- ส่วนงานเครือข่าย

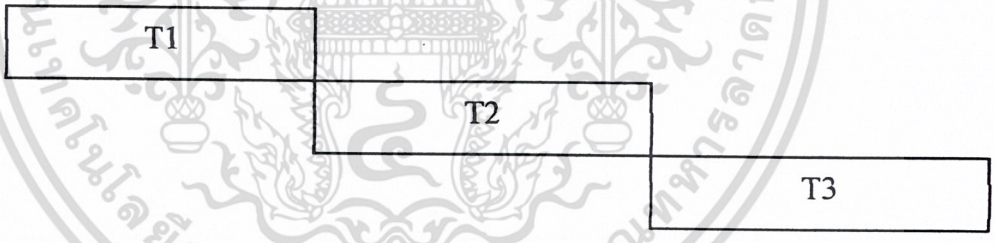
ซึ่งจากการพิจารณาแล้วส่วนงานที่ใช้ความรู้ความสามารถทางการสร้างตัวแปลภาษาคือ 3 ส่วนงานแรก ในส่วนงานที่ 4 คือส่วนงานเครือข่ายนั้นจะอาศัยความสามารถทางด้านการเขียนโปรแกรมผ่านเครือข่ายซึ่งจากการติดต่อกับนาย Ahmed El Deeb แล้วพบว่ามีผู้ที่ถนัดทางการเขียนโปรแกรมผ่านเครือข่ายอยู่แล้วจึงมุ่งหมายการพัฒนาไปที่ส่วนงานทั้ง 3 ส่วน คือ การวิเคราะห์ภาษาเอนเอเอสแอลและสร้างเลคสิเคอร์รูล, นำเลคมาบรรยายภาษาเอนเอเอสแอล และ นำแฟลกและไบสันมาสร้างตัวแปลภาษาเอนเอเอสแอล

โดยงานทั้ง 3 ส่วนจะมีความสัมพันธ์ที่ขึ้นต่อกันเป็นดังรูปที่ 4.6 และสามารถนำไปเป็นงานหลักของการพัฒนาแบบวนรอบซึ่งแสดงในรูปที่ 4.7 ได้

T1 คือขั้นตอนการวิเคราะห์ภาษาเอนเอเอสแอลและสร้างเลคสิเคอร์รูล

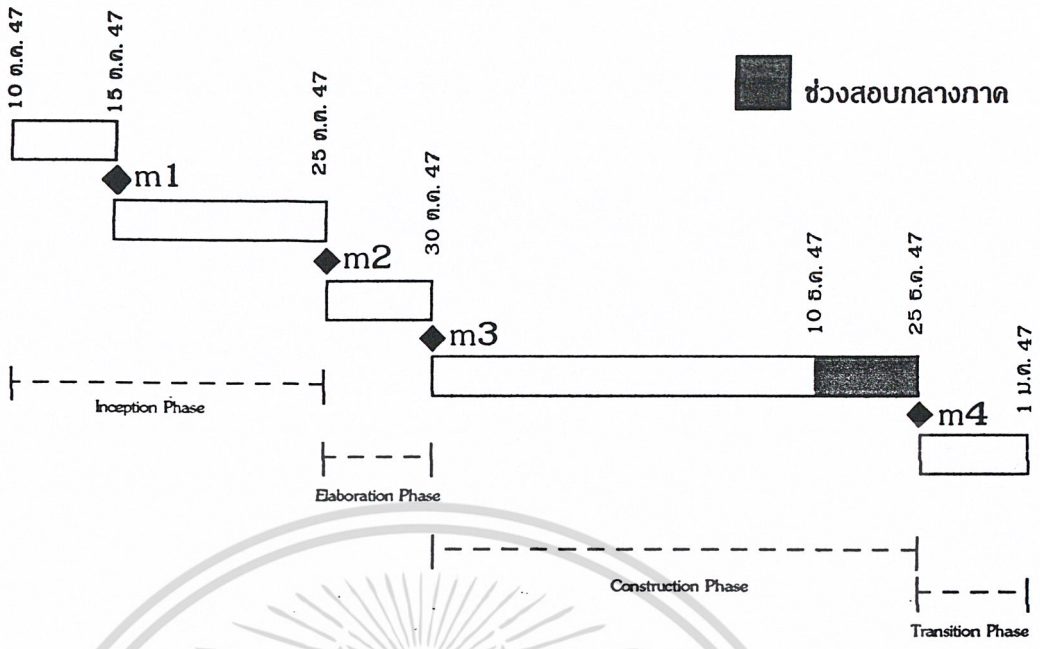
T2 คือขั้นตอนการนำเลคมาบรรยายภาษาเอนเอเอสแอล

T3 คือขั้นตอนการนำเฟลกและไบสันมาสร้างตัวแปลภาษาเอนเอเอสแอล



รูปที่ 4.6 แสดงความสัมพันธ์การขึ้นต่อกันของส่วนงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.7 แสดงแผนงานการพัฒนาแบบวนรอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
ขั้นตอนการพัฒนา

ตามที่ได้ออกแบบแนวทางการพัฒนาโปรแกรมตัวแปลภาษาเอนเอเอสแอลไว้แล้ว มีส่วนงานหลักที่ต้องทำทั้งหมด 3 ส่วนงานคือ

- วิเคราะห์ภาษาเอนเอเอสแอลและสร้างเลคสิเคอร์รูล (Lexical Rule)
- นำเลคมาบรรยายภาษาเอนเอเอสแอล
- นำเฟลกและไบสันมาสร้างตัวแปลภาษาเอนเอเอสแอล

5.1 ขั้นตอนการวิเคราะห์ภาษาเอนเอเอสแอลและสร้างเลคสิเคอร์รูล

ในขั้นตอนนี้เริ่มจากการศึกษารายละเอียดของภาษาเอนเอเอสแอลจาก NASL2 Reference manual พบว่าชนิดตัวแปรที่ภาษาเอนเอเอสแอลสนับสนุนมีดังนี้

- Integer
- String
- Array
- NULL value
- Booleans

และสามารถทำงานร่วมกับตัวดำเนินการ(Operators) ดังต่อไปนี้

- ตัวดำเนินการทั่วไป
  - = เป็นตัวดำเนินการให้ค่า (assignment operator)
  - [] เป็นตัวดำเนินการบอกตัวชี้อาร์เรย์ (Array index operator)
- ตัวดำเนินการที่เกี่ยวกับการคำนวณ
  - + เป็นตัวดำเนินการบวก addition operator
  - – เป็นตัวดำเนินการลบ subtraction operator
  - * เป็นตัวดำเนินการคูณ multiplication operator
  - / เป็นตัวดำเนินการหาร integer division operator
  - % เป็นตัวดำเนินการหารเอาเศษ modulo
  - ** เป็นตัวดำเนินการยกกำลัง exponentiation หรือ power function
- ตัวดำเนินการคล้ายในภาษาซีที่เป็นที่นิยม
  - ++ เป็น pre-incrementation (++x) หรือ post-incrementation (x++)
  - - เป็น pre-decrementation (–x) หรือ post-decrementation (x–)
  - += -= *= /= %= มีความหมายเหมือนกับในภาษา C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- <<= และ >>= ก็มีอยู่ด้วย; แต่มี >>>= เพิ่มเข้ามาอีกด้วย

- ตัวดำเนินการกับสตริง
  - + เป็นตัวดำเนินการต่อสตริง string concatenation
  - – เป็นตัวดำเนินการตัดสตริง "string subtraction"
  - [] ตัวดำเนินการดึงสตริงที่ถูกชี้extracts one character from the string
  - >< เป็นตัวดำเนินการเปรียบเทียบความเท่ากันของสตริง "string match"
  - >!< เป็นตัวดำเนินการเปรียบเทียบความไม่เท่ากัน "string don't match"
  - =~ เป็นตัวดำเนินการเปรียบเทียบความเท่ากันแบบเรคกูล่าร์เอ็กซ์เพรสชั่น "regex match" operator
  - !~ ตัวดำเนินการเปรียบเทียบความไม่เท่ากันแบบเรคกูล่าร์เอ็กซ์เพรสชั่น "regex don't match" operator
- ตัวดำเนินการเปรียบเทียบ
  - == เป็น TRUE ถ้า arguments ทั้งคู่เท่ากัน, FALSE ในทางตรงกันข้าม
  - != เป็น TRUE ถ้า arguments ทั้งคู่ต่างกัน, TRUE ในทางตรงกันข้าม
  - > เป็นตัวดำเนินการเปรียบเทียบความมากกว่า "greater than" operator
  - >=เป็นเป็นตัวดำเนินการเปรียบเทียบความมากกว่าหรือเท่ากัน "greater than or equal" operator
  - < เป็นตัวดำเนินการเปรียบเทียบความน้อยกว่า "lesser than" operator
  - <= เป็นเป็นตัวดำเนินการเปรียบเทียบความน้อยกว่าหรือเท่ากัน "lesser than or equal" operator
- ตัวดำเนินการทางตรรกะ
  - ! เป็น logical "not". TRUE ถ้า argument เป็น FALSE, FALSE ในทางตรงกันข้าม
  - && เป็น logical "and". ถ้า argument ตัวแรกเป็นFALSE, ตัวที่สองก็ไม่ต้อง evaluated
  - || เป็น logical "or". ถ้า argument ตัวแรกเป็น TRUE, ตัวที่สองก็ไม่ต้อง evaluated
- ตัวดำเนินการในระดับบิต
  - ~ เป็นตัวดำเนินการทางคณิตศาสตร์ที่เรียกว่า "not", the 1-complement
  - & เป็นตัวดำเนินการทางคณิตศาสตร์ที่เรียกว่า "and"
  - | เป็นตัวดำเนินการทางคณิตศาสตร์ที่เรียกว่า "or"
  - ^ เป็นตัวดำเนินการทางคณิตศาสตร์ที่เรียกว่า "xor" (exclusive or)
  - << เป็นตัวดำเนินการเลื่อนซ้าย logical bit shift to the left
  - >> เป็นตัวดำเนินการเลื่อนขวา arithmetic / signed shift to the right10

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

○ >>> เป็น logical / unsigned shift to the right11

ลำดับความสำคัญ

ตัวดำเนินการเหล่านั้นมีลำดับความสำคัญโดยเรียงจากสูงไปต่ำดังตารางที่ 5.1

Operators	Associativity
++ --	None
**	Right
~ -(unary minus)	Left
!	Left
* / %	Left
+ -	Left
<< >> >>>	Left
&	Left
^	Left
|	Left
< <= > >= == != < > =~ !~ >!< ><	None
&&	Left
||	Left
= += -= *= /= %= <<= >>= >>>=	Right

ตารางที่ 5.1 แสดงลำดับความสำคัญพร้อมความสัมพันธ์กับผลที่ได้

จากนั้นวิเคราะห์รูปแบบของภาษาเอนเอเอสแอลสามารถกำหนดโทเคนเป็นดังตารางที่ 5.2

โทเคน	ตัวอย่างอินพุต	ข้อมูลอธิบายโทเคน
IF	if	'if'
ELSE	else	'else'
EQ	==	'=='
NEQ	!=	'!='
SUPEQ	>=	'>='
INFEQ	<=	'<='
OR	||	'||'
AND	&&	'&&'
MATCH	><	'><'
NOMATCH	>!<	'>!<'
REP	rep	'rep'
FOR	for	'for'
REPEAT	repeat	'repeat'
UNTIL	until	'until'
FOREACH	foreach	'foreach'

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

WHILE	while	'while'
BREAK	break	'break'
FUNCTION	function	'function'
RETURN	return	'return'
INCLUDE	include	'include'
LOCAL	local_var	'local_var'
GLOBAL	global_var	'global_var'
PLUS_PLUS	++	'++'
MINUS_MINUS	--	'__'
L_SHIFT	<<	'<<'
R_SHIFT	>>	'>>'
R_USHIFT	>>>	'>>>'
EXPO	**	'**'
PLUS_EQ	+=	'+='
MINUS_EQ	-=	'-='
MULT_EQ	*=	'*='
DIV_EQ	/=	'/='
MODULO_EQ	%=	'%='
L_SHIFT_EQ	<<=	'<<='
R_SHIFT_EQ	>>=	'>>='
R_USHIFT_EQ	>>>=	'>>>='
RE_MATCH	=~	'=~'
RE_NOMATCH	!~	'!~'
IDENT	script_id, Name, port9	เป็นอักษรที่เรียงต่อกันไม่ว่าจะเป็นตัวเล็กหรือใหญ่ สามารถใส่เครื่องหมาย underscore ไว้หลังจากอักษร
STRING1	'This love'	string ระหว่าง simple quotes
STRING2	"www/apache", "General"	string ระหว่าง double quotes
INTEGER	399, 0365, 0x2AB4	เป็นตัวเลขฐานสิบที่เรียงต่อกันหรือนำหน้าด้วยเลขศูนย์แล้วตามด้วยเลขฐานแปด หรือนำหน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

		ด้วย 0x แล้วตามด้วยเลขฐานสิบหก

ตารางที่ 5.2 แสดงโทเคนและตัวอย่างอินพุตพร้อมคำอธิบายรูปแบบ

5.2 ขั้นตอนการนำเลคมาบรรยายภาษาเอนเอเอสแอล

ในขั้นตอนนี้จะเป็นการนำเลคมาบรรยายภาษาเอนเอเอสแอลซึ่งใช้เครื่องมือเฟรกในการทดสอบการบรรยายภาษาเอนเอเอสแอลโดยใช้ข้อมูลที่ได้วิเคราะห์ภาษาไว้เบื้องต้นในตารางที่ 5.2 จัดทำเรคกูลาร์เอ็กซ์เพรสชั่น (regular expression) ที่ครอบคลุมตามคำอธิบายรูปแบบของแต่ละโทเคน ซึ่งนอกเหนือจากโทเคนที่แสดงในตารางที่ 5.2 แล้วหากสแกนพบอักขระใดๆ จะส่งเป็นอักขระหนึ่งตัวอักษรไปให้พาเซอร์ (parser) ซึ่งจะครอบคลุมอักขระอย่างเช่น {,}, [,], +,-,*, /, <,>เป็นต้นซึ่งเหตุผลที่ทำแบบนี้เพราะว่าจะมีส่วนช่วยในการตรวจสอบความผิดพลาดในขั้นตอนการพาซ์ ต่อไปได้ เมื่อพิจารณาได้ดังกล่าวแล้วก็เริ่มทำการพัฒนาในส่วนเลคเซอร์ โดยใช้เครื่องมือเฟรก (flex) และใช้โปรแกรมจีวีไอเอ็ม (gvim) เป็นเครื่องมือในการทำงาน ซึ่งในการพัฒนานั้นก็จะสร้างเรคกูลาร์เอ็กซ์เพรสชั่น ที่ทำให้เฟรกสามารถระบุโทเคนได้ โดยผู้จัดทำโครงงานพิจารณาจากคำอธิบายรูปแบบ ของโทเคน นั้นๆ ประกอบการสร้างเรคกูลาร์เอ็กซ์เพรสชั่น ซึ่งในระหว่างการเรคกูลาร์เอ็กซ์เพรสชั่น พบว่า เกิดกรณีคาบเกี่ยวกันของการระบุเรคกูลาร์เอ็กซ์เพรสชั่น ขึ้น ยกตัวอย่างกรณีของ integer ที่เป็นเลขฐานสิบหก หากข้อมูลที่รับเข้ามาเป็น '0x0052' และเรามีการเรคกูลาร์เอ็กซ์เพรสชั่น ไว้เป็น [0x][0-9a-zA-Z] ให้เป็น INTEGER ที่เป็นเลขฐานสิบหก และมีการกำหนด IDENTIFIER ไว้เป็น [a-zA-Z][a-zA-Z0-9_] ไม่ว่าจะกำหนด IDENTIFIER ไว้ก่อนหรือหลังการกำหนดเรคกูลาร์เอ็กซ์เพรสชั่น ของ INTEGER ตัวเลคเซอร์ที่สร้างขึ้นมาจากเฟรกจะระบุว่าพบ โทเคน '0' เป็น INTEGER และ 'x' ที่เป็น IDENTIFIER และ '0052' เป็น INTEGER ซึ่งผิดวัตถุประสงค์ของผู้สร้างที่ต้องการให้ระบุเพียงโทเคนเดียว ซึ่งกรณีเช่นเดียวกันนี้เกิดกับการระบุ STRING1, STRING2 และ COMMENT

จากการศึกษาพบว่า ในกรณีนี้สามารถแก้ไขได้โดยการใช้ Start condition ซึ่งเป็นกลไกที่เฟรกจัดเตรียมไว้ให้ในกรณีเช่นนี้ โดยลักษณะการใช้ในการระบุ INTEGER ของเราจะเสมือนระบุให้มีพิจารณา เป็นกรณีพิเศษไว้ คือถ้าเรคกูลาร์เอ็กซ์เพรสชั่นเป็นเลขศูนย์ '0' ก็ให้ BEGIN(ST_ZERO) หมายถึงให้เข้าสู่ state ST_ZERO แล้วให้พิจารณาเฉพาะเรคกูลาร์เอ็กซ์เพรสชั่นที่เราได้ระบุไว้เท่านั้น ถ้าดูจากรูปแบบ ที่เรากำหนดไว้เรคกูลาร์เอ็กซ์เพรสชั่นที่จะพิจารณาใน state ST_ZERO ก็จะได้เป็น

[xX][0-9a-zA-Z]+ เป็น INTEGER ที่เป็นเลขฐานสิบหก

[0-7]+ เป็น INTEGER ที่เป็นเลขฐานแปด

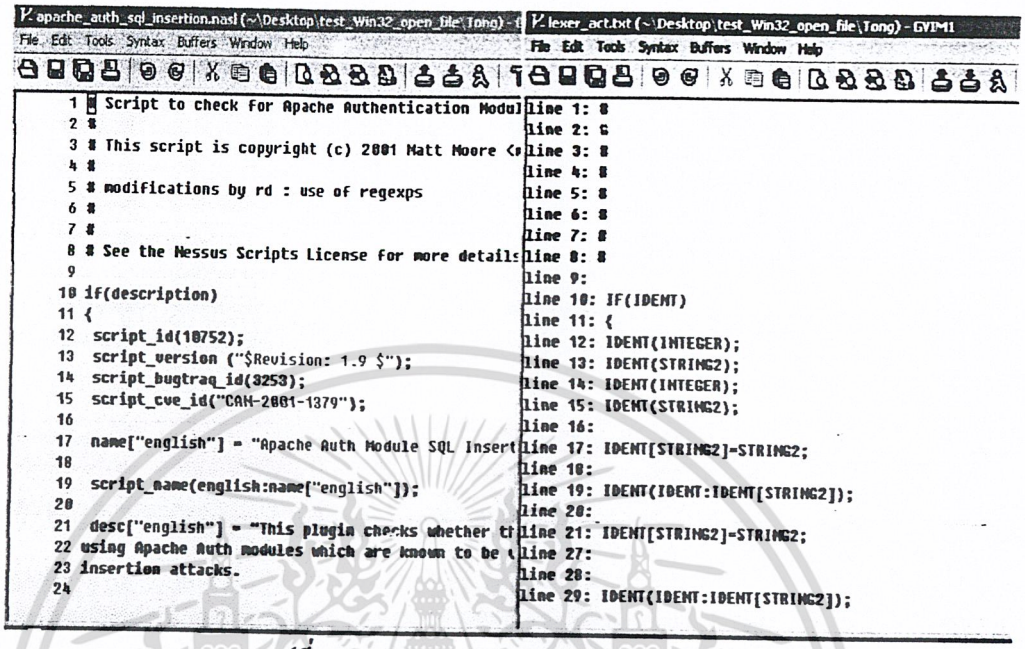
[0-9]+ เป็น INTEGER ที่เป็นเลขฐานสิบ

[^0-90-7xX\n] ให้กลับไปพิจารณา state initial นั่นคือ state ปรกตินั่นเอง

ซึ่ง STRING1, STRING2 และ COMMENT ก็ใช้กลไกเช่นเดียวกันนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังที่ได้กล่าวไปแล้วว่าอินพุต ของขั้นตอนนี้เป็นเอนเอเอสแอลสคริปและเอาท์พุต เป็น โทเคนที่ได้จากการสแกน พร้อมส่งให้พาเซอร์ ต่อไป ซึ่งในขั้นตอนนี้ได้ทำการเก็บผลการตรวจจับ โทเคนไว้ในไฟล์ ดังตัวอย่างในรูปที่ 5.1



รูปที่ 5.1 แสดงผลการตรวจจับโทเคน

จากรูปซ้ายมือเป็นสคริปที่ต้องการจะทำการสแกนขวามือเป็นผลการตรวจจับโทเคน ซึ่งการทำงานในส่วนนี้จะแจ้งความผิดพลาดเมื่อพบ Invalid identifier คือเราจะต้องเรคกูลาร์เอ็กซ์เพรสชั่นให้สามารถระบุว่าอินพุตที่รับเข้ามานั้นจำแนกเป็นโทเคนตามที่ภาษาเอนเอเอสแอล ต้องการได้

5.3 ขั้นตอนการนำเฟลกและไบสันมาสร้างตัวแปลภาษาเอนเอเอสแอล

จากขั้นตอนที่ผ่านมาจะได้กฎที่พร้อมสำหรับการสร้างเลคเซอร์แล้ว จึงใช้เครื่องเฟรกสร้างเลคเซอร์ขึ้นจากข้อมูลที่มีอยู่และเริ่มการสร้างพาเซอร์ด้วยเครื่องไบสัน โดยทุกๆภาษาทางโปรแกรมจะมีกฎ ที่กำหนดขึ้นมาเป็นโครงสร้างในการสร้างประโยคที่เป็นรูปแบบที่รู้ว่าเป็นโปรแกรมนั้นๆ เช่นในโปรแกรมภาษาปาซคาลจะเกิดมาจากบล๊อกหลายๆ บล๊อกและบล๊อกเกิดมาจากหลายๆ สเตทเมนท์ และสเตทเมนท์ เกิดมาจากเอ็กซ์เพรสชั่นหลายๆเอ็กซ์เพรสชั่นและ เอ็กซ์เพรสชั่นก็จะต้องมาจากโทเคนหลายๆ โทเคนเช่นเดียวกัน ดังนั้น การสร้างประโยคของภาษาทางโปรแกรมจะสามารถสร้างได้โดยอธิบายจาก context-free grammar หรือ BNF (Backus-Naur Form) ดังนั้นแกรมม่า มีความสำคัญมากกับทั้งการออกแบบภาษาและสร้างตัวแปลภาษาโดย

- แกรมม่าที่ใช้จะต้องให้ความแม่นยำ, ง่ายในการเข้าใจ และมีการระบุ syntactic ของภาษาทางโปรแกรมนั้นๆ
- จากความแม่นยำของแกรมม่า เราจะสามารถสร้างพาเซอร์ ที่มีประสิทธิภาพในการตัดสินว่าโปรแกรมที่เขียนนั้นมีการสร้างประโยคที่ถูกต้องตามรูปแบบ ดังนั้นใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นตอนการสร้างพาเซอร์การที่สามารถทราบว่าประโยคที่สร้างนั้นกำกวม และ สิ่งใดก็ตามที่จะทำให้ยากต่อการพาซ์ ก็จะช่วยให้มันทำงานได้ง่ายขึ้น

- การออกแบบแกรมม่า ที่เหมาะสมจะทำให้โครงสร้างของภาษาทางโปรแกรมนั้น ช่วยให้การแปลงโปรแกรมที่ถูกเขียนไปเป็นภาษาเครื่องได้อย่างถูกต้องรวมทั้งการตรวจสอบความผิดพลาดของโปรแกรมที่เขียนซึ่งก็จะมีเครื่องมือสำหรับการแปลงการอธิบายแกรมม่าไปเป็นการทำงานของโปรแกรม
- เมื่อภาษามีการพัฒนาอย่างต่อเนื่องมาเป็นระยะเวลาหนึ่ง ก็ได้มีการเรียนรู้วิธีการสร้างแบบใหม่และการเพิ่มประสิทธิภาพในการทำงาน ดังนั้นวิธีการสร้างแบบใหม่นี้สามารถทำให้การสร้างภาษาใหม่ๆง่ายขึ้นเมื่อเราสามารถสร้างโดยอาศัยรายละเอียดแกรมม่าที่มีอยู่แล้วของภาษาได้

ดังนั้นในขั้นตอนนี้จึงเน้นในส่วนของการพิจารณา Context-free Grammar ของภาษาเอนเอเอสแอลเพื่อให้ได้มาซึ่งแกรมม่าที่ถูกต้องสมบูรณ์ โดย Context-free Grammar นั้นจะประกอบไปด้วย เทอร์มินัล (terminals), นอนเทอร์มินัล (nonterminals), สัญลักษณ์เริ่มต้น (a start symbol) และ โปรดักชั่น (productions)

เทอร์มินัล เป็นสัญลักษณ์พื้นฐานที่ได้มาจากสตริงรูปแบบต่างๆ ซึ่งก็คือโทเคนนั่นเอง

นอนเทอร์มินัล เป็นตัวแปรที่ใช้แทนประโยคที่แสดงกลุ่มของสตริงมันจะช่วยการวิเคราะห์ประโยคและการแปลงประโยค

สัญลักษณ์เริ่มต้น จะเป็นนอนเทอร์มินัลตัวหนึ่งที่ถูกเลือกขึ้นมา และเป็นกลุ่ม ของสตริงจะเห็นว่าไม่ใช่ถูกๆ นอนเทอร์มินัลที่จะได้เป็นสัญลักษณ์เริ่มต้นซึ่งมันจะเป็นจุดเริ่มที่จะกำหนดว่าภาษานั้นๆ มีการจำกัดความโดยแกรมม่าหรือไม่อย่างไร

โปรดักชั่น ของแกรมม่าจะระบุลักษณะที่เทอร์มินัลและนอนเทอร์มินัลสามารถรวมกันเป็นสตริงได้ แต่ละโปรดักชั่นจะมีนอนเทอร์มินัลตามด้วยเครื่องหมายลูกศร ต่อด้วยสตริงนอนเทอร์มินัลและหรือเทอร์มินัล

ซึ่งเมื่อพิจารณาภาษาเอนเอเอสแอลแล้ว ก็จะมีเทอร์มินัลเป็นโทเคนดังที่ได้กำหนดไว้แล้วในขั้นตอนการสร้างเลคเซอร์ และเมื่อพิจารณาจากไฟล์โปรแกรมที่เขียนขึ้นมาโดย Michel Arboi แล้วเราจะสามารถทราบได้ว่ามีนอนเทอร์มินัล เป็น tiptop, instr_decl_list, instr_decl, func_decl, arg_decl, arg_decl_1, block, instr_list, instr, simple_instr, ret, if_block, loop, for_loop, while_loop, repeat_loop, foreach_loop, aff_func, rep, string, inc, func_call, arg_list, arg_list_1, arg, aff, lvalue, identifier, array_elem, array_index, post_pre_incr, expr, var, ipaddr, loc, glob เหตุที่เราสามารถทราบได้เช่นนี้เนื่องจากว่าไบสัน มีรูปแบบการที่จะสามารถให้กำหนดกฎของแกรมม่าโดยสัญลักษณ์นอนเทอร์มินัล ต่อด้วยเครื่องหมาย ':' ตามด้วยกลุ่มของนอนเทอร์มินัล และหรือ เทอร์มินัลที่ derive มาจากสัญลักษณ์นอนเทอร์มินัล นั้น และตามด้วยเอ็กซ์ชั่นที่ต้องการจะให้ทำหากพบกลุ่มของโทเคน ที่ที่ตรงกับสัญลักษณ์นอนเทอร์มินัล นั้น จะเห็นว่าเมื่อเราทราบรูปแบบการกำหนดกฎของแกรมม่าของไบสันแล้วเราก็จะทราบว่าแกรมม่าของภาษาเอนเอเอสแอลประกอบด้วยสัญลักษณ์นอนเทอร์มินัล ใดบ้าง นอกจากนี้เรายังสามารถ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทราบได้อีกว่าสัญลักษณ์เริ่มต้น เป็น นอนเทอร์มินัล ที่ชื่อว่า tiptop เราสามารถทราบได้เช่นนี้เนื่องจากไบสันเองมีกฎที่บอกว่ากฎแรกที่อยู่ในไฟล์จะเป็นสัญลักษณ์เริ่มต้นหากไม่มีการปรับแต่งใดๆ หรือถ้าไม่อย่างนั้นต้องมีการประกาศออกมาเป็นพิเศษซึ่งก็จะมีรูปแบบการประกาศ โดยใช้ข้อความ '%start' แล้วตามด้วยสัญลักษณ์เริ่มต้น เมื่อพิจารณาดูแล้วพบว่า ในโปรแกรมที่ถูกเขียนพบว่ามีการประกาศ '%start tiptop' ไว้จึงสามารถสรุปได้ว่าสัญลักษณ์เริ่มต้น ของแกรมม่าของภาษาเอนเอเอสแอลเป็น tiptop เมื่อทราบข้อมูลดังที่ได้กล่าวมาแล้วนั้นเราจึงสามารถนำมาเขียนเป็นโครงสร้างประโยคของภาษาเอนเอเอสแอลได้ดังต่อไปนี้

tiptop⟶ instr_decl_list

instr_decl_list⟶ instr_decl

instr_decl⟶ instr | func_decl

func_decl⟶ FUNCTION identifier '(' arg_decl ')' block

arg_decl⟶ | arg_decl_1

arg_decl_1⟶ identifier | identifier ',' arg_decl_1

block⟶ '{' instr_list '}' | '{' '}'

instr_list⟶ instr | instr instr_list

instr⟶ simple_instr ';'| block | if_block | loop

simple_instr ⟶ aff | post_pre_incr | rep | func_call | ret | inc | loc | glob | BREAK

ret⟶ RETURN expr | RETURN

if_block⟶ IF '(' expr ')' instr | IF '(' expr ')' instr ELSE instr

loop ⟶ for_loop | while_loop | repeat_loop | foreach_loop

for_loop ⟶ FOR '(' aff_func ';' expr ';' aff_func ')' instr

while_loop ⟶ WHILE '(' expr ')' instr

repeat_loop ⟶ REPEAT instr UNTIL expr ';'

foreach_loop ⟶ FOREACH identifier '(' expr ')' instr

aff_func⟶ aff | post_pre_incr | func_call |

rep⟶ func_call REP expr

string ⟶ STRING1 | STRING2

inc⟶ INCLUDE '(' string ')'

func_call⟶ identifier '(' arg_list ')'

arg_list ⟶ arg_list_1 |

arg_list_1⟶ arg | arg ',' arg_list_1

arg ⟶ expr | identifier '⟶' expr

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

aff⟶ lvalue '=' expr| lvalue PLUS_EQ expr| lvalue MINUS_EQ expr| lvalue MULT_EQ expr| lvalue DIV_EQ expr| lvalue MODULO_EQ expr| lvalue R_SHIFT_EQ expr| lvalue R_USHIFT_EQ expr| lvalue L_SHIFT_EQ expr

lvalue⟶ identifier| array_elem

identifier⟶ IDENT | REP

array_elem⟶ identifier '[' array_index ']'

array_index⟶ expr

post_pre_incr⟶ PLUS_PLUS lvalue| MINUS_MINUS lvalue| lvalue PLUS_PLUS| lvalue MINUS_MINUS

var⟶ identifier| array_elem | func_call

ipaddr⟶ INTEGER '.' INTEGER '.' INTEGER '.' INTEGER

loc⟶ LOCAL arg_decl

glob⟶ GLOBAL arg_decl

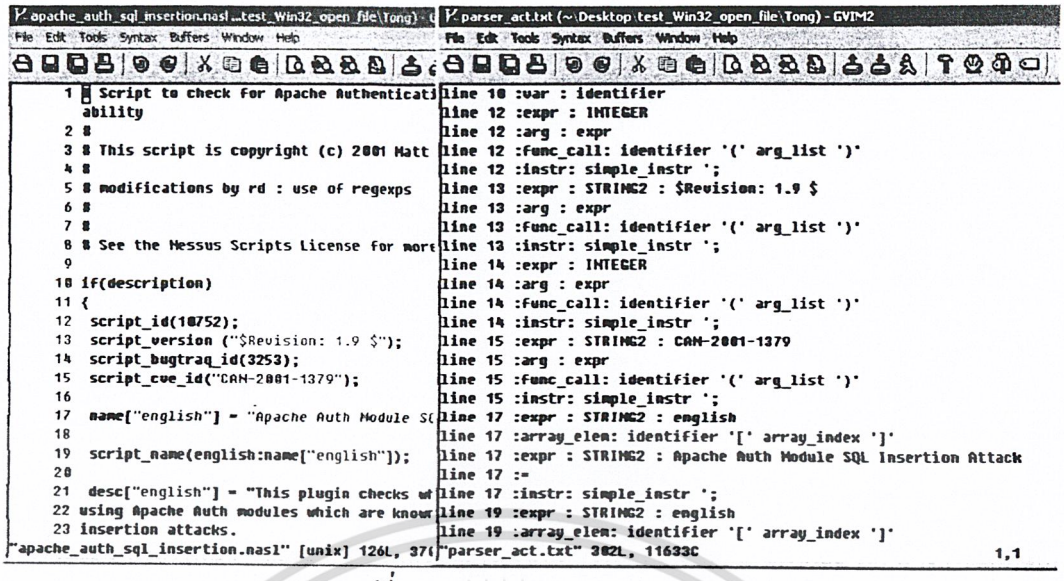
ซึ่งเราสามารถนำไปสร้างพาเซอร์ได้ โดยใช้ เครื่องมือไบสันสำหรับระบบปฏิบัติการวินโดว์ในการสร้าง โดยอินพุต ของขั้นตอนนี้จะเป็นโทเคนที่ถูกส่งมาให้จากเลคเซอร์ที่ได้สร้างไว้แล้วในขั้นตอนแรก โดยการติดต่อกันระหว่างพาเซอร์ และเลคเซอร์จะแตกต่างจากโปรแกรมตัวแปลภาษาของ Michel Arboi ที่ทำงานได้ในระบบยูนิกส์ สืบเนื่องมาจากความแตกต่างในการเลือกวิธีสร้างเลคเซอร์ซึ่งโดยทั่วไปมีอยู่ 3 วิธีที่จะสร้างเลคเซอร์

- ใช้ lexical-analyzer generator, เช่นเฟรก(flex)ที่จะสร้างเลคเซอร์โดยอาศัยการกำหนดเรคกูลาร์เอ็กซ์เพรสชั่นในกรณีนี้เครื่องมือที่ใช้สร้างจะจัดเตรียมชุดคำสั่ง สำหรับการอ่านและการจัดสรรบัฟเฟอร์ของอินพุตซึ่งผู้จัดทำโครงงานเลือกวิธีการนี้
- เขียนเลคเซอร์โดยใช้การเขียนโปรแกรมติดต่อกับระบบที่เกี่ยวข้องกับ I/O ในการที่จะอ่านอินพุต ซึ่ง Renaud Deraison และ Michel Arboi ผู้คิดค้นและสร้างตัวแปรภาษาเอนเอเอสแอลที่ทำงานบนระบบยูนิกส์ เลือกวิธีนี้โดยใช้ภาษาซีในการสร้าง
- ใช้ภาษาแอสแซมบลี้ เขียนเลคเซอร์ และควบคุมการอ่านอินพุต

ในส่วนที่ผู้จัดทำสร้างนั้นจะสามารถปล่อยให้พาเซอร์ เรียกฟังก์ชั่น yylex() ได้อย่างอัตโนมัติได้เลย ขณะที่โปรแกรมเดิมนั้นสร้างโดยภาษาซี จึงต้องมีการส่งผ่านอาร์กิวเมนท์ ที่เป็น โครงสร้างเฉพาะผ่านฟังก์ชั่น ชื่อ nasllex()ไปบอกข้อมูลอย่างเช่นชื่อไฟล์ ด้วย

การพัฒนาในขั้นตอนนี้จะ เขียนผลการตรวจพบนอนเทอร์มินัลไว้เพื่อตรวจสอบความถูกต้องด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.2 แสดงผลการ Parse script

5.4 ขั้นตอนการสร้างสื่อกลางสำหรับการแปลภาษา (Intermediate code generation)

โดยความสามารถของเฟรกและไบสันนั้นสามารถทำได้เพียงแค่ตรวจสอบการสร้างประโยคในโปรแกรมว่าถูกต้องตามโครงสร้างการสร้างประโยคของภาษานั้นๆ หรือไม่ ซึ่งในการสร้างตัวแปลภาษาที่มีภาระหน้าที่แปลงโปรแกรมจากภาษาหนึ่งไปเป็นอีกภาษาหนึ่งนั้นจะต้องมีขั้นตอนการสร้างสื่อกลางสำหรับการแปลภาษาเพื่อนำไปแปลเป็นอีกภาษาหนึ่งได้ต่อไป โดยเมื่อพิจารณาการสร้างสื่อแปลภาษาเอนเอเอสแอลที่ Michel Arboi เป็นผู้สร้างขึ้นจะสามารถทราบได้ว่า ได้มีการสร้างสื่อกลางสำหรับการแปลภาษาชนิดที่เรียกว่า three-address code ที่เป็นลำดับของ สเตทเมนท์ในรูปแบบทั่วไป จึงได้มีการศึกษาจากโปรแกรมเดิมเพื่อให้โครงสร้างต่างๆ ของโปรแกรมตัวแปลภาษาใกล้เคียงจากของเดิมให้มากที่สุดเมื่อศึกษาไประยะเวลาหนึ่งจึงได้ทราบว่า union ที่ใช้เสมือนตัวเชื่อโหนดของทรีจะเก็บค่าต่างๆ ในกรณีเจอนอนเทอร์มินัลนั้นถูกประกาศใช้โดย '%union' ซึ่งถือเป็นกลไกที่ไบสันจัดเตรียมไว้ให้สำหรับช่วยในการสร้าง three-address code ด้วย ซึ่งดูได้จากตารางที่ 5.3 ที่เป็นการแสดงชนิดของข้อมูลและชื่อที่ใช้ในการอ้างถึงซึ่งถูกสร้างโดย '%union' ในโปรแกรมตัวแปรภาษาเอนเอเอสแอลที่ทำงานได้ในระบบยูนิกส์เดิม

ชนิดของข้อมูล	ชื่อที่ใช้ในการอ้างถึง
int	num
char	*str
struct asciiz	data
• char	*val
• int	len
tree_cell	*node

ตารางที่ 5.3 แสดงชนิดของข้อมูลและชื่อที่ใช้ในการอ้างถึง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อพิจารณาจากโปรแกรมตัวแปลภาษาเอนเอเอสแอลเดิมก็จะทราบได้ว่าในแต่ละนอนเทอร์มินัลที่ตรวจพบได้ในขั้นตอนการพาร์ซ ก็จะมีการเก็บค่าไว้เป็นโหนดโดยมีการจองพื้นที่ไว้ เป็นโครงสร้าง tree_cell แล้วเก็บไว้กับลิงค์ที่เป็น union โดยลิงค์นั้นอ้างถึงได้โดย pseudo-variable $$ ซึ่งถือเป็นงานหลักของการปฏิบัติงานในไบสันซึ่งก็จะมีการอ้างถึงตัวแปรโครงสร้างประโยคของของกฎนั้นๆ โดยอ้างถึง $1, $2 และหากมีตัวแปรโครงสร้างประโยคอยู่อีกก็เรียงลำดับต่อไปเป็น $3, $4... เช่น

for_loop ⟶ FOR '(' aff_func ';' expr ';' aff_func ')' instr

ภายในการปฏิบัติงานของกฎนี้จะมีการอ้างถึงนอนเทอร์มินัลที่เป็นตัวแปรโครงสร้างประโยคของ กฎนี้ก็สามารถอ้างถึงนอนเทอร์มินัล aff_func ว่าเป็น $3, expr เป็น $5, aff_func เป็น $7 และ instr ว่าเป็น $9

ต่อจากนั้นจะต้องมีการเรียกใช้ฟังก์ชั่นในการจองพื้นที่สำหรับโหนด ซึ่งฟังก์ชั่นในการจองพื้นที่นั้นมีอยู่หลายฟังก์ชั่น เนื่องจากมีรูปแบบความต้องการในการเก็บข้อมูลที่แตกต่างกันดังตารางที่ 5.4

Function declaration	Description
tree_cell* alloc_tree_cell(int, char*);	ใช้ในการจองพื้นที่สำหรับ tree_cell ของ rule ที่ต้องการเก็บชนิดของ node และ node ต่อไปเท่านั้น
tree_cell* alloc_expr_cell(int, int, tree_cell*, tree_cell*);	ใช้ในการจองพื้นที่สำหรับ tree_cell ของ rule ที่เป็น expression ซึ่งจะต้องเก็บชนิดของ node และ node ที่เกี่ยวข้องมากกว่าหนึ่ง

ตารางที่ 5.4 แสดงฟังก์ชั่นที่ใช้จองพื้นที่ของโหนดทรี

ซึ่งในการจองพื้นที่แต่ละครั้งจะมีการเก็บชนิดของโหนดนั้นๆไว้ด้วยเพื่อสะดวกในการระบุและแปลความหมายต่อไปซึ่งเมื่อตรวจสอบจากโปรแกรมตัวแปลภาษาเอนเอเอสแอลเดิมแล้วจะสามารถทราบได้ว่ามีการประกาศชนิดไว้โดยใช้ enum node_type ซึ่งจะมีการเก็บชนิดของโหนด ไว้ดังตารางที่ 5.5

นอนเทอร์มินัลของแต่ละกฎ	ชนิดของโหนด
instr_decl_list	NODE_INSTR_L
func_decl	NODE_FUN_DEF
arg_decl_1	NODE_DECL

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

instr_list	NODE_INSTR_L
simple_instr	NODE_BREAK
ret	NODE_RETURN
if_block	NODE_IF_ELSE
for_loop	NODE_FOR
while_loop	NODE_WHILE
repeat_loop	NODE_REPEAT_UNTIL
foreach_loop	NODE_FOREACH
rep	NODE_REPEATED
func_call	NODE_FUN_CALL
arg	NODE_ARG
aff	NODE_PLUS_EQ, NODE_MINUS_EQ, NODE_MULT_EQ, NODE_DIV_EQ, NODE_MODULO_EQ, NODE_R_SHIFT_EQ, NODE_R_USHIFT_EQ, NODE_L_SHIFT_EQ
lvalue	NODE_VAR
array_elem	NODE_ARRAY_EL
post_pre_incr	EXPR_INCR
expr	EXPR_AND, EXPR_NOT, EXPR_OR, EXPR_PLUS, EXPR_MINUS, EXPR_U_MINUS, EXPR_BIT_NOT, EXPR_MULT, EXPR_EXPO, EXPR_DIV, EXPR_MODULO, EXPR_BIT_AND, EXPR_BIT_XOR, EXPR_BIT_OR, EXPR_R_SHIFT, EXPR_R_USHIFT, EXPR_L_SHIFT, COMP_MATCH, COMP_NOMATCH, COMP_RE_MATCH, COMP_RE_NOMATCH, COMP_LT, COMP_GT, COMP_EQ, COMP_NE,

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	COMP_GE, COMP_LE, CONST_INT, CONST_STR, CONST_DATA
var	NODE_VAR
ipaddr	CONST_STR
loc	NODE_LOCAL
glob	NODE_GLOBAL

ตารางที่ 5.5 แสดงนอนเทอร์มินัล และชนิดที่กฎนั้นๆ เก็บค่าระบุชนิดของโหนดไว้

เมื่อดูจากตารางแล้วจะสังเกตเห็นว่าไม่ใช่ทุกนอนเทอร์มินัลที่ derive ออกมาเป็นกฎที่จะสร้างโหนดเก็บเป็นชนิดเอาไว้เนื่องจากบางกฎก็สร้างมาเพื่อประโยชน์ในการเชื่อมต่อโครงสร้างประโยคไม่จำเป็นต้องเก็บค่าใส่โหนดไว้เนื่องจากมิได้มีประโยชน์ในการนำไปแปลภาษาจาก สื่อกลางสำหรับการแปลภาษาไปเป็นภาษาที่ต้องการ และบางกฎที่มีการ derive จากนอนเทอร์มินัล เป็นเทอร์มินัลเลยก็จะสามารถเก็บค่าของเทอร์มินัลนั้นได้เลย เช่นกฎ string ⟶ STRING1 ก็สามารถเก็บค่า ของสตริงที่รับมาได้เลย ซึ่งในทางกลับกันบางกฎที่ derive เป็นได้ หลายๆ แบบก็สามารถเก็บโหนดต่างชนิดกันได้เช่นเดียงกับกฎของนอนเทอร์มินัล aff และ expr

ซึ่งจากข้อมูลที่มีอยู่จึงนำมาสร้างเป็น Three-address code ขึ้นโดยจะจองพื้นที่ให้กับกฎที่พบเจอโดยจะมีการเรียกใช้งานฟังก์ชั่นที่เอาไว้จองพื้นที่ของโหนดทรีแล้วส่งพอร์ยเตอร์ต่อกันไปเรื่อยๆ จนถึงกฎที่อยู่บนสุดที่สามารถวิ่งไปได้ โดยในขั้นตอนนี้ผู้จัดทำโครงงานได้ทำการ dump ทรีเก็บไว้เป็นผลของขั้นตอนนี้เพื่อตรวจสอบการทำงานของขั้นตอนนี้โดยฟังก์ชั่น

void nasl_dump_tree(const tree_cell* c)

เป็น Function ที่จะถูกเรียกใช้ในขั้นตอนนี้ โดยมันจะไม่มีการรีเทอร์นค่าใดๆกลับมาแต่จะไปเขียนผลของการทำงานไว้ที่ไฟล์ที่ชื่อ output.txt

ส่วนฟังก์ชั่นอื่นๆที่เกี่ยวข้องนั้นจะดูจากโปรแกรมตัวแปลภาษาที่ทำงานได้บนยูนิกส์ในระดับฟังก์ชั่นการทำงานที่เกี่ยวข้องกับการทำงานที่สนใจว่าสามารถนำมาใช้กับระบบวินโดว์ได้เลยหรือไม่และหากไม่สามารถนำมาใช้งานได้ก็จะพิจารณาว่าจะทำการแก้ไขให้สามารถใช้งานได้อย่างไร โดยเงื่อนไขที่จะใช้พิจารณาเบื้องต้นจะพิจารณาจากฟังก์ชั่นการทำงานนั้นๆ มีการทำงานที่เกี่ยวข้องกับเรื่องดังนี้หรือไม่หากเกี่ยวข้องจะต้องพิจารณาแก้ไขในแง่การเขียนโปรแกรมต่อไป

Process

Signals and signal handling

Threads

Memory management

User, groups, and security

File and data access

Interprocess communication

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

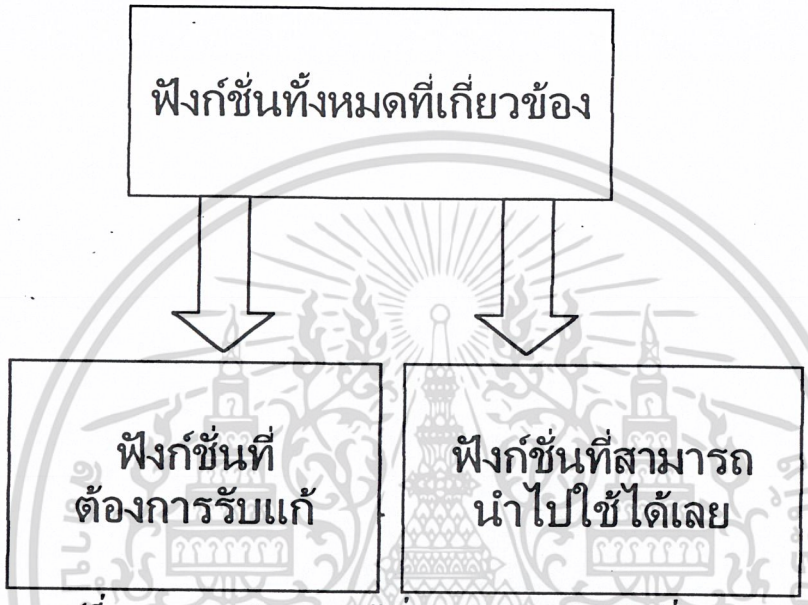
Sockets and networking

Process environment

Multiprocessor considerations

Deamons and services

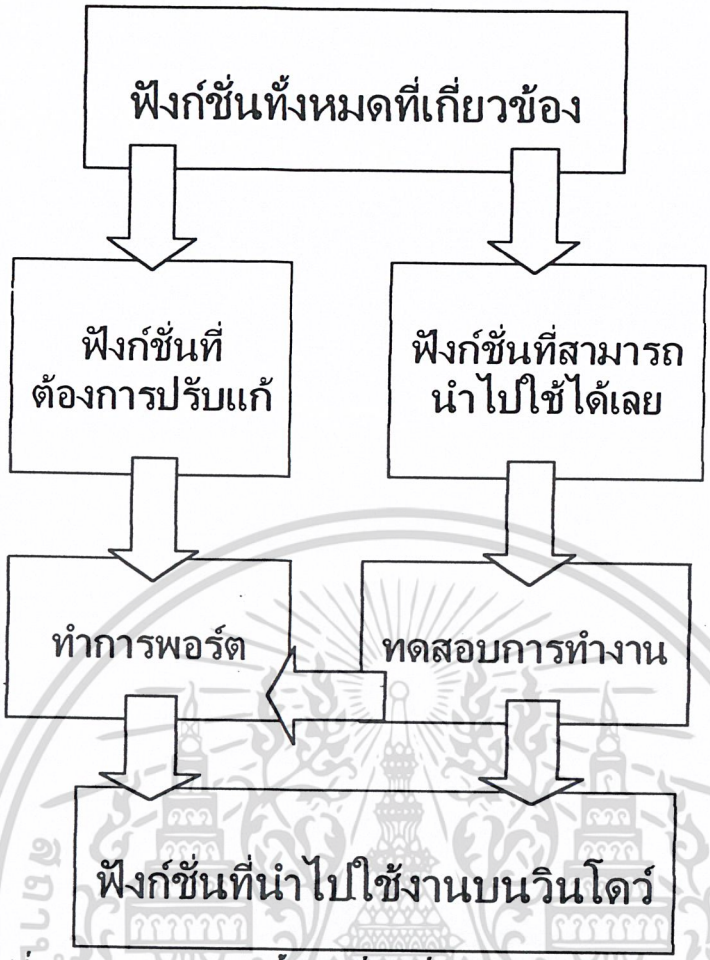
เมื่อผ่านขั้นตอนการพิจารณาหน้าที่การทำงานของแต่ละฟังก์ชั่นที่เกี่ยวข้องดังกล่าวก็จะแบ่งฟังก์ชั่นได้เป็น 2 ส่วนดังรูป



รูปที่ 5.3 แสดงการพิจารณาหน้าที่การทำงานและแบ่งฟังก์ชั่นการทำงานออกเป็น 2 ส่วน

โดยทุกๆ ฟังก์ชั่นที่เกี่ยวข้องกับส่วนโปรแกรมที่มีการอ้างถึงโดยตัวแปรภาษาแอนเอเอสแอลจะต้องนำมาเข้ากระบวนการนี้ทุกๆ ฟังก์ชั่นโดยลักษณะการปรับแก้จะเป็นดังรูปที่ 5.4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.4 แสดงกระบวนการทั้งหมดที่ฟังก์ชั่นจะต้องผ่านเข้ามาทำการพิจารณา

จากรูปที่ 5.4 จะเห็นว่าฟังก์ชั่นที่เกี่ยวข้องกับการทำงานของตัวแปลภาษาเอนเอเอสแอล จะต้องถูกพิจารณาและถูกแบ่งเป็นฟังก์ชั่นที่ต้องทำการปรับแก้ และฟังก์ชั่นที่สามารถนำไปใช้ได้เลย โดยส่วนที่ต้องทำการแก้ไขก็ทำการพอร์ตฟังก์ชั่นการทำงานนั้นๆ ให้สามารถทำงานได้ หลังจากที่ทำการพอร์ตแล้วฟังก์ชั่นการทำงานนั้นๆ ก็จะเป็นฟังก์ชั่นที่สามารถนำไปใช้บนวินโดว์ได้ ส่วนฟังก์ชั่นที่พิจารณาเบื้องต้นแล้วสามารถนำไปใช้งานได้เลยก็จะต้องเข้ากระบวนการทดสอบการทำงานบนระบบวินโดว์ก่อนหากตัวงานของฟังก์ชั่นนั้นๆไม่ได้มีส่วนที่ไปเกี่ยวข้องกับกรณีที่เราพิจารณาแต่ไปเรียกใช้ฟังก์ชั่นที่ต้องการปรับแก้ก็จะต้องนำส่วนงานนั้นมาทำการพอร์ตด้วยเช่นกันซึ่งสุดท้ายเมื่อเสร็จสิ้นกระบวนการพอร์ตก็จะเป็นฟังก์ชั่นทีสามารถนำไปใช้งานบนวินโดว์ได้เช่นเดียวกับฟังก์ชั่นที่พิจารณาว่าสามารถใช้งานได้แล้วทดสอบการทำงานผ่าน

รายชื่อฟังก์ชั่นที่ผ่านกระบวนการดังกล่าวสามารถแสดงดังต่อไปนี้

- tree_cell* alloc_tree_cell(int, char*);
- tree_cell* alloc_expr_cell(int, int, tree_cell*, tree_cell*);
- tree_cell* alloc_RE_cell(int, int, tree_cell*, char*);
- tree_cell* alloc_typed_cell(int);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- int nasl_is_leaf(const tree_cell*);
- char* get_line_nb(const tree_cell*);
- tree_cell* dup_cell(const tree_cell*);
- void nasl_dump_tree(const tree_cell*);
- void ref_cell(tree_cell*);
- void deref_cell(tree_cell*);
- const char* nasl_type_name(int);
- int cell_type(const tree_cell*);
- char* dump_cell_val(const tree_cell*);
- tree_cell* nasl_affect(tree_cell*, tree_cell*);
- void clear_unnamed_var(anon_nasl_var*);
- const char* var2str(const anon_nasl_var*);
- anon_nasl_var* nasl_get_var_by_num(nasl_array*, int, int);
- nasl_iterator nasl_array_iterator(tree_cell*);
- tree_cell* nasl_iterate_array(nasl_iterator*);
- int add_var_to_list(nasl_array*, int, const anon_nasl_var*);
- int add_var_to_array(nasl_array*, char*, const anon_nasl_var*);
- int array_max_index(nasl_array*);
- void free_array(nasl_array *);
- tree_cell* copy_ref_array(const tree_cell*);
- int hash_str2(const char*, int);
- tree_cell* var2cell(anon_nasl_var*);
- const char* array2str(const nasl_array*);
- static int hash_str(const char* s);
- void free_func_chain(nasl_func*);
- ExtFunc void arg_add_value(struct arglist *, const char *, int, long, void *);
- ExtFunc int arg_set_value(struct arglist *, const char *, long, void *);
- ExtFunc int arg_set_type(struct arglist *, const char *, int);
- ExtFunc void * arg_get_value(struct arglist *, const char *);
- ExtFunc int arg_get_type(struct arglist *, const char *);
- ExtFunc int arg_get_length(struct arglist *, const char *);
- ExtFunc void arg_dump(struct arglist *, int);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ExtFunc void arg_dup(struct arglist *, struct arglist *);
- ExtFunc void arg_free(struct arglist *);
- ExtFunc void arg_free_all(struct arglist *);
- ExtFunc void arg_free_name(char*);
- static int mkhash(const char * name);
- static void cache_init();
- static struct name_cache * cache_get_name(char * name);
- static struct name_cache * cache_add_name(char * name);
- static char * cache_inc(char * name);
- static void cache_dec(char * name);
- ExtFunc void arg_free_name(name);
- ExtFunc void static struct arglist * arg_get(struct arglist * arg, const char * name);
- ExtFunc arg_add_value(struct arglist * arglst, const char * name, int type, long length, void * value);
- ExtFunc int arg_set_value(struct arglist * arglst, const char * name, long length, void *value);
- ExtFunc int arg_set_type(struct arglist * arglst, const char * name, int type);
- ExtFunc void * arg_get_value(struct arglist * args, const char * name);
- ExtFunc int arg_get_length(struct arglist * args, const char * name);
- ExtFunc int arg_get_type(struct arglist * args, const char * name);
- ExtFunc void arg_dup(struct arglist * dst, struct arglist * src);
- ExtFunc void arg_dump(struct arglist * args, int level);
- ExtFunc void arg_free(struct arglist* arg);
- ExtFunc void rg_free_all(struct arglist* arg);
- int init_nasl_ctx(naslctxt*, const char*);
- void nasl_clean_ctx(naslctxt*);
- extern int nasl_regcomp (regex_t *preg, const char *pattern, int cflags);
- extern int nasl_regexec (const regex_t *preg, const char *string, size_t nmatch,regmatch_t pmatch[], int eflags);
- extern size_t nasl_regerror (int errcode, const regex_t *preg, char *errbuf, size_t errbuf_size);
- extern reg_syntax_t nasl_re_set_syntax (reg_syntax_t syntax);

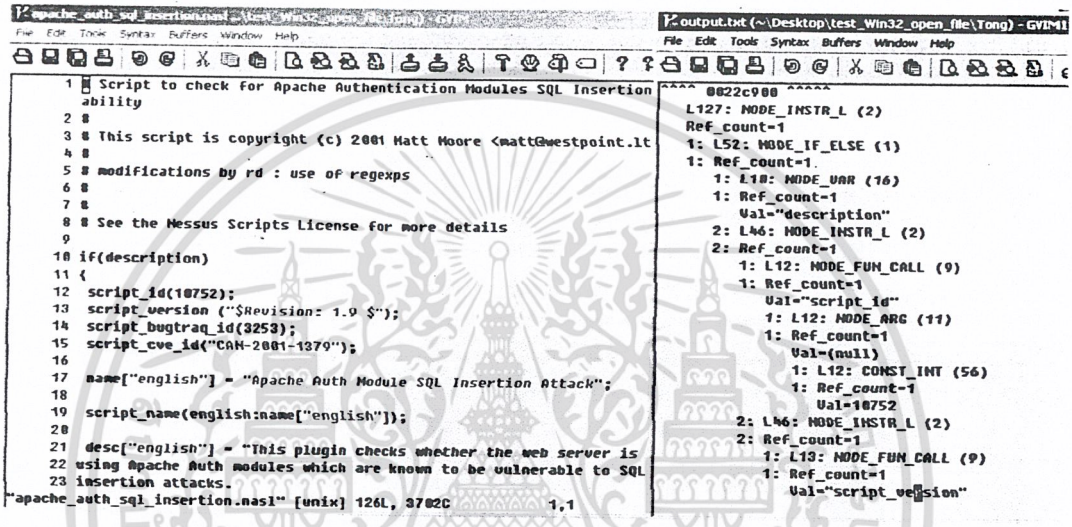
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- extern void nasl_regfree (regex_t *preg);
- int init_nasl_ctx(naslctxt*, const char*);
- void nasl_clean_ctx(naslctxt*);
- struct arglist * init(char * hostname, struct in_addr ip);
- static struct arglist * init_hostinfos(char * hostname, struct in_addr * ip);
- tree_cell* nasl_exec(lex_ctxt* lexic, tree_cell* st);
- static int expo(int x, int y);
- Int cell_cmp(lex_ctxt* lexic, tree_cell* c1, tree_cell* c2);
- tree_cell* bool2cell(int x);
- tree_cell* int2cell(int x);
- char* cell2str(lex_ctxt* lexic, tree_cell* c);
- static int cvt_bool(lex_ctxt* lexic, tree_cell* c);
- Int cell2bool(lex_ctxt* lexic, tree_cell* c);
- Int cell2int(lex_ctxt* lexic, tree_cell* c);
- static void nasl_short_dump(FILE* fp, const tree_cell* c);
- static void nasl_dump_expr(FILE* fp, const tree_cell* c);
- tree_cell* cell2atom(lex_ctxt* lexic, tree_cell* c1);
- tree_cell* decl_nasl_func(lex_ctxt* lexic, tree_cell* decl_node);
- nasl_func* insert_nasl_func(lex_ctxt* lexic, const char* fname, tree_cell* decl_node);
- int compare(const void *arg1, const void *arg2);
- static nasl_func* get_func(lex_ctxt* ctxt, const char* name, int h);
- tree_cell* get_array_elem(lex_ctxt* ctxt, const char* name, tree_cell* idx);
- tree_cell* get_variable_by_name(lex_ctxt* ctxt, const char* name);
- static named_nasl_var* get_var_ref_by_name(lex_ctxt* ctxt, const char* name, int climb);
- static named_nasl_var* get_var_by_name(nasl_array* a, const char* s);
- tree_cell* decl_global_variables(lex_ctxt* lexic, tree_cell* vars);
- tree_cell* decl_local_variables(lex_ctxt* lexic, tree_cell* vars);
- tree_cell* nasl_incr_variable(lex_ctxt* lexic, tree_cell* tc, int pre, int val);
- tree_cell* nasl_read_var_ref(lex_ctxt* lexic, tree_cell* tc);
- static const char* get_var_name(anon_nasl_var *v);
- void nasl_trace(lex_ctxt * lexic, char * msg, ...);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- int nasl_trace_enabled();
- Int init_nasl_ctx(naslctxt* pc, const char* name);

อย่างไรก็ดีดังที่ได้กล่าวไว้แล้วว่าโครงงานนี้ เป็นการสร้างตัวแปลภาษาเอนเอเอสแอลดังนั้นจึงสนใจในการตรวจสอบว่าทรีที่สร้างขึ้นมามีการทำงานอย่างไรและมันสามารถทำงานได้ตามที่มุ่งหวังหรือไม่จึงทำการ นำผลการทำงานลง file ไว้ หากแต่ถ้าเป็นการนำไปประกอบเป็น Nessus ส่วนแม่ข่ายแล้วจุดมุ่งหมายของการใช้งานจะแตกต่างกันไป การที่จะเก็บผลการตรวจสอบในส่วนนี้จึงจะไม่จำเป็นต่อไป โดยผลการทำงานของ dump tree ก็จะเป็นดังนี้



รูปที่ 5.5 ขวามือเป็นสคริปที่นำมาเข้าขั้นตอนการแปลภาษาและขวามือเป็นผลลัพธ์การ dump tree

ซึ่งต่อจากนี้จะมีการนำไปแปลเป็นภาษาซีต่อไป อย่างไรก็ตามเนื่องจากในขั้นตอนการ แปลภาษานั้นจะต้องมีการเรียกใช้ nessus-library มากมายที่เกี่ยวข้องกับการเขียนโปรแกรมผ่านเครือข่ายซึ่งจากที่ได้มีการติดต่อกับ Ahmed El Deeb ที่เป็นผู้ควบคุมโครงการ Win-nessus ดังที่ได้กล่าวไว้เบื้องต้นแล้ว พบว่าได้มีความพยายามสร้าง nessus-library ในส่วนนี้ขึ้น จึงหยุดการพัฒนาไว้ที่จุดนี้พร้อมเตรียมส่งโปรแกรมตัวแปลภาษาเอนเอเอสแอลที่ได้พัฒนาไว้ไปที่ CVS ของ win-nessus เพื่อให้บรรลุผลออกมาเป็น Win-nessus ได้ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
ลักษณะการทำงานของตัวแปรภาษาเอนเอเอสแอล

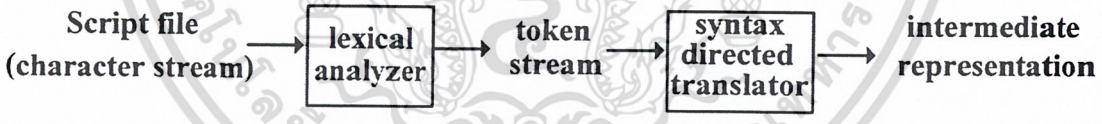
6.1 การทำงานของตัวแปรภาษาเอนเอเอสแอล

โปรแกรมตัวแปลภาษาเอนเอเอสพัฒนาขึ้นจาก Dec-C++ เวอร์ชัน 4.18 ซึ่งเป็น GNU software โดยการทำงานจะใช้เครื่องมือเฟลก (Flex) เวอร์ชัน 2.5.4 และไบสัน (Bison) เวอร์ชัน 1.24 ซึ่งเป็น GNU software เช่นเดียวกัน เป็นเครื่องมือในสร้างชุดคำสั่งภาษาซี ขั้นตอนแรกสร้างโครงงานโปรแกรมภาษาซีที่ทำงานบนคอนโซล (Project Console Application C Language) โดย Dec-C++ ซึ่งการทำงานในส่วนนี้จะได้ออกมาเป็นไฟล์ปฏิบัติงาน (Execute File) ที่มีการสั่งงานทางคอมมานโหมด (command mode) จึงได้ใช้ Visual C++ เวอร์ชัน 6 เป็นส่วนช่วยในการเรียกใช้งาน และกราฟิกยูเซอร์อินเตอร์เฟส (GUI) ให้เรียกใช้ทดสอบการทำงานได้ง่ายขึ้น

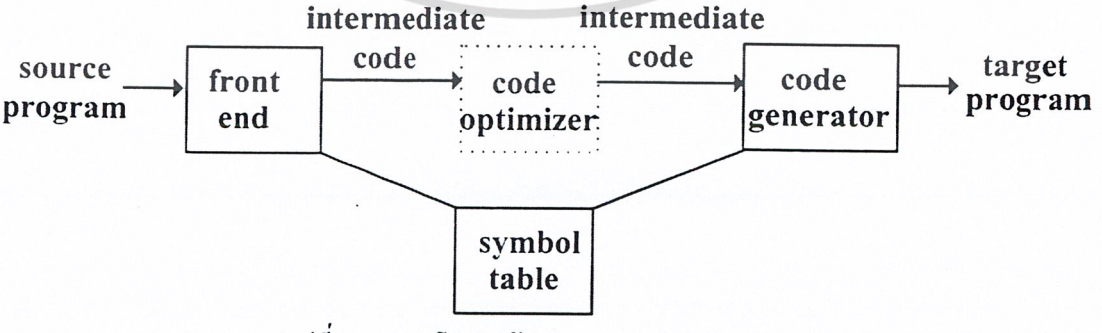
รายละเอียดการทำงานในแต่ละส่วน

เนื่องจากจะเห็นว่าจากการแบ่งงานจะมีส่วนงานที่แยกออกจากกัน 2 ส่วน อย่างเห็นได้ชัด คือ ส่วนที่ได้ออกมาเป็นไฟล์ปฏิบัติงานที่ทำงานในส่วนของตัวแปลภาษากับส่วนงานที่เป็นกราฟิกยูเซอร์อินเตอร์เฟสควบคุมการทำงานของระบบงานส่วนตัวแปลภาษา

ในส่วนงานตัวแปลภาษานั้นมีการแบ่งโครงสร้างออกเป็นส่วนงานอย่างชัดเจนซึ่งแสดงได้ดังรูปที่ 6.1 และ รูปที่ 6.2



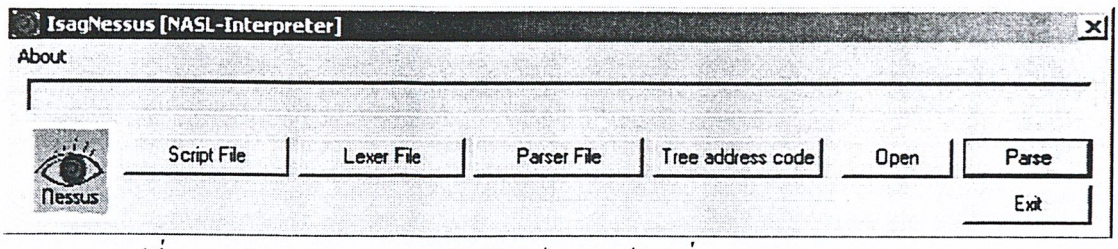
รูปที่ 6.1 แสดงโครงสร้างของ Compiler front end



รูปที่ 6.2 แสดงโครงสร้างของ Compiler back end

ในส่วนงาน GUI นั้นเป็นส่วนที่ไปเรียกใช้ไฟล์ปฏิบัติงานที่ถูกสร้างขึ้นมาจากส่วนงาน ตัวแปลภาษาซึ่งมีลักษณะโปรแกรมดังรูปที่ 6.3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

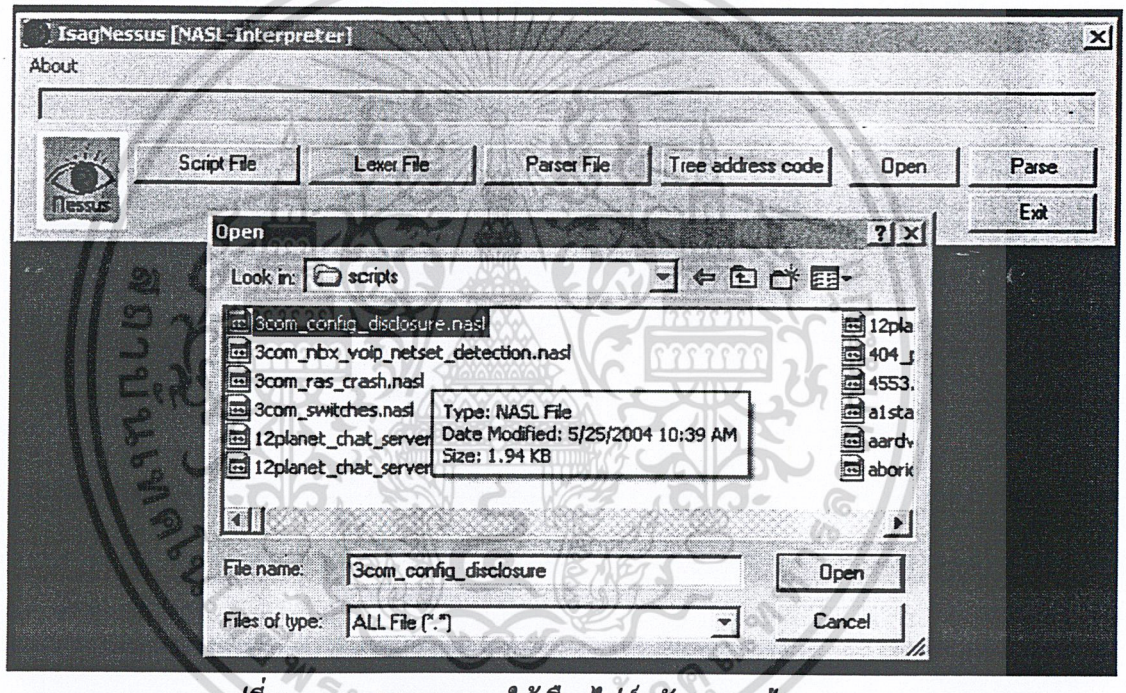


รูปที่ 6.3 แสดงลักษณะของกราฟิกยูเซอร์อินเตอร์เฟสที่เรียกใช้ส่วนงานตัวแปลภาษา

การใช้งานโปรแกรมตัวแปลภาษาเอนเอเอสแอล มีขั้นตอนดังนี้

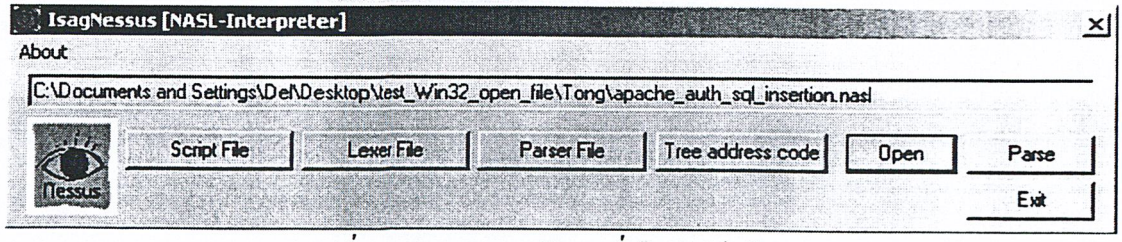
1. เลือกไฟล์สคริปต้องการนำเข้ากระบวนการแปลภาษา

โดยกดที่ปุ่ม Open จะมี Dialog Box ให้สามารถเลือกไฟล์ *.nasl ได้ ให้เลือกสคริปเอนเอเอสแอลที่เราจะนำมาเข้ากระบวนการแปลภาษา



รูปที่ 6.4 แสดง Dialog Box ให้เลือกไฟล์หลังจากกดปุ่ม Open

เมื่อเลือกไฟล์ .nasl ได้หนึ่งไฟล์แล้วก็จะมีชื่อเส้นทางคำสั่งในระบบดอส พร้อมชื่อไฟล์ที่เลือกไว้แล้วแสดงขึ้นในกล่องแสดงรายชื่อ ดังรูป

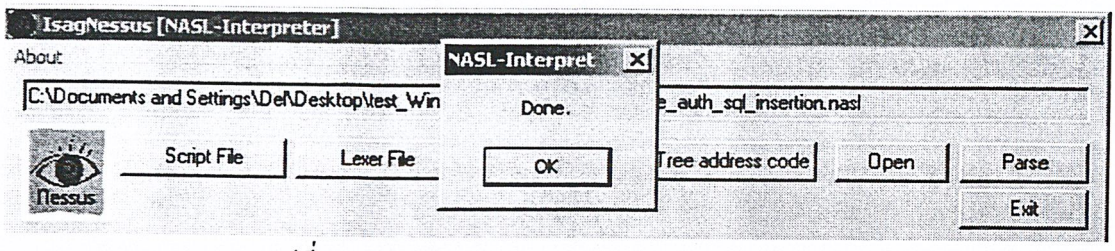


รูปที่ 6.5 แสดง List หลังจากที่เลือกไฟล์แล้ว

2. กดปุ่ม Parse เพื่อกระทำตามกระบวนการแปลภาษา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

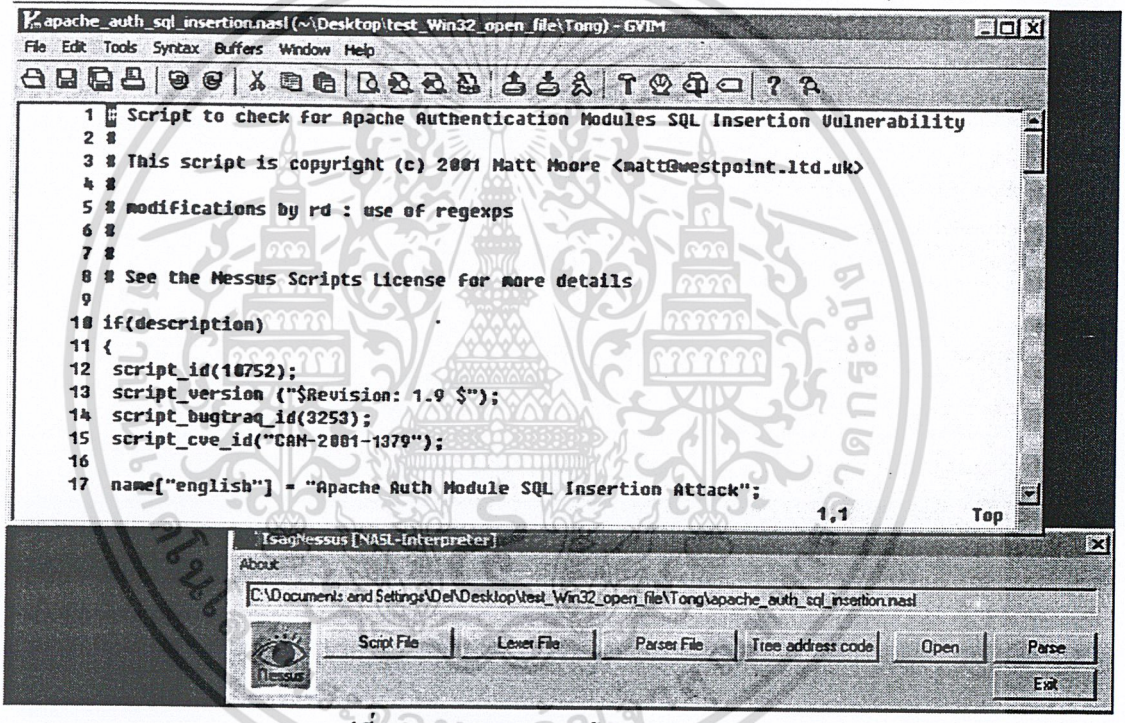
เมื่อกดปุ่ม Parse แล้วเมื่อโปรแกรมทำงานเสร็จลงจะมีข้อความบอกว่าทำงานแล้ว



รูปที่ 6.6 แสดง Message Box หลังจากทำงานเสร็จ

ให้กดปุ่ม Ok เพื่อรับทราบข้อความ ดังกล่าว

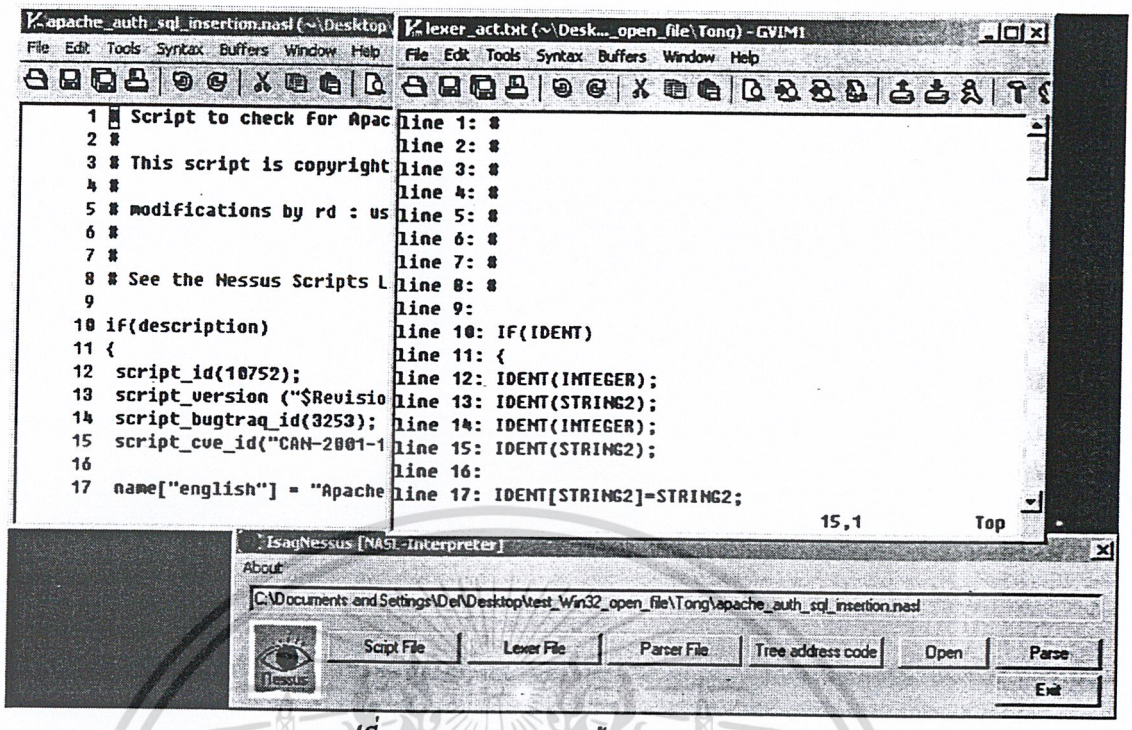
3. สามารถตรวจสอบการทำงานของโปรแกรมได้โดยกดปุ่มสคริปไฟล์เพื่อตรวจสอบ สคริปไฟล์ ที่นำมาเป็นอินพุต ของโปรแกรมว่าได้ผมรับถูกต้องกับที่ต้องการหรือไม่โดยกดปุ่ม Script File



รูปที่ 6.7 แสดงผลการกดปุ่ม Script File

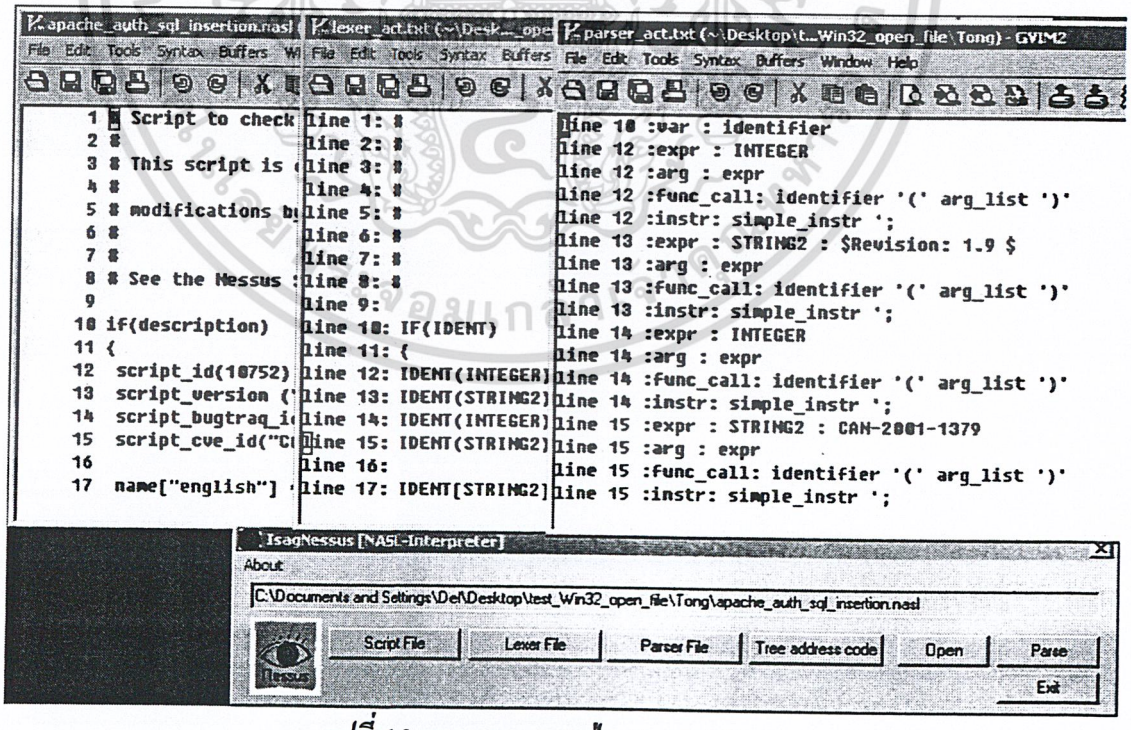
ซึ่งจะมีการไปเรียก Editor vim ขึ้นมาเพื่อแสดงรายละเอียดไฟล์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.8 แสดงผลการกดปุ่ม Lexer File

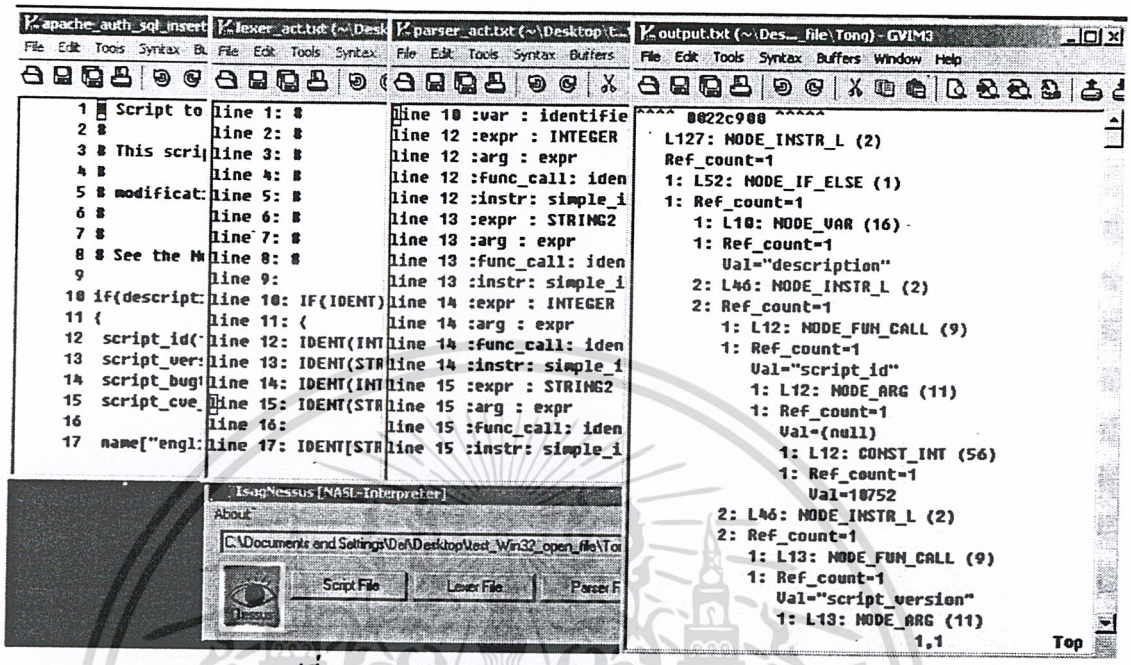
จะเห็นว่าเราก็สามารถนำมาเปรียบเทียบกับสคริปไฟล์ว่าได้ผลโทเคนที่ถูกต้องหรือไม่



รูปที่ 6.9 แสดงผลการกดปุ่ม Parser File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะเห็นว่าหลังจากกดปุ่ม Parser File เราสามารถตรวจสอบความถูกต้องในการที่กฎใดมีการตรวจพบกลุ่มของโทเคน ที่บ่งชี้ว่า ถูกต้องตามแกรมม่าได้หรือไม่



รูปที่ 6.10 แสดงผลการกดปุ่ม Three-address code

จะเห็นว่าหลังจากกดปุ่มจะสามารถตรวจสอบได้ว่า Intermediate Representation ที่ได้ออกมานั้นถูกต้องตามที่ต้องการหรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7

สรุปผลและวิจารณ์

7.1 สรุป

เพื่อเป็นการแสดงให้เห็นถึงปริมาณงานที่ได้ทำลงไปกับส่วนงานทั้งหมดของ Nessus ส่วนแม่ข่าย จะแสดงหน่วยปริมาณงานโดยยึดจากขนาดของ source code ในส่วนงานต่างๆ ดังนี้

ขนาดของ source code ทั้งหมดของโครงงาน Nessus ส่วนแม่ข่าย

ส่วน nessus-core ประกอบไปด้วย

Nessus(client)	1.59	MB
Nessusd(server)	591	KB
Ssl	4.26	KB
Include	28.4	KB

ส่วน nessus-libraries ประกอบไปด้วย

Libnessus	335	KB
Libhosts_gratherer	35.2	KB
Libpcap-nessus	472	KB
Include	96.1	KB

ส่วน libnasl ประกอบไปด้วย

Nasl	534	KB
Include	14	KB

ดังนั้นขนาดของ source code ส่วนแม่ข่ายจะประกอบไปด้วยส่วน Nessusd(server), ssl, include(nessus-core), libnessus, libhosts_gratherer, libpcap-nessus, include(nessus-libraries), nasl, include ซึ่งจะมีขนาดเป็น 2109.96 KB = 2.1 MB

และส่วนงาน nasl-interpreter ที่ได้จัดทำจะประกอบไปด้วยส่วนงาน nasl และ include(libnasl) ดังนั้น source code ของส่วนงานนี้จะมีขนาดเป็น 548 KB ซึ่งสามารถคิดเป็นสัดส่วนได้ดังนี้

สัดส่วนของส่วนงาน nasl-interpreter ต่อ nessus ส่วนแม่ข่าย

= (548/2109.96)*100

= 26 % ของงานส่วน nessus server

7.2 ปัญหาและอุปสรรค

ในการพัฒนาโปรแกรมแปลภาษาเอนเอเอสแอลนี้ มีปัญหาและอุปสรรคในการพัฒนาหลายประการดังนี้

1) เนื่องจากการพัฒนาโปรแกรมแปลภาษาเอนเอเอสแอลนี้เป็นการพอร์ตโปรแกรมจากระบบปฏิบัติการ ยูนิกส์ให้สามารถทำงานบนระบบปฏิบัติการวินโดว์ได้จึงใช้เวลากับการศึกษาลักษณะการเขียนโปรแกรมค่อนข้างมาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2) โปรแกรมตัวแปลภาษาเอนเอเอสแอลนั้นไม่สามารถเรียกใช้งานโดยคอมมานโหมดได้ เนื่องมาจากเกิดปัญหาด้านการจองพื้นที่ของโปรเซสซึ่งหากเรียกใช้งานโดยคอมมานโหมดจะมีการใช้พื้นที่ร่วมกันระหว่างโปรเซสและทำให้เกิดการทำงานผิดพลาดได้
3) เนื่องจากการพัฒนาโปรแกรมแปลภาษาเอนเอเอสแอลนี้เป็นการพัฒนาร่วมกับชาวต่างชาติการติดต่อสื่อสารระหว่างผู้ร่วมโครงการ Win-nessus จึงจะใช้เวลาในการโต้ตอบแต่ละครั้งนั้นใช้เวลา
4) ตัวแปลภาษาเอนเอเอสแอลยังไม่สามารถทำการแปลภาษาซีออกมาได้เนื่องจากในขั้นตอนการสร้างโปรแกรมภาษาซีนั้นจะต้องมีการเรียกใช้งานไลบาร์ลี่ที่เกี่ยวข้องกับการเขียนโปรแกรมผ่านเครือข่ายซึ่งพัฒนาโดยผู้ร่วมโครงการ Win-nessus ผู้อื่นและโครงการนี้ไม่มีค่าตอบแทนโดยตรงในการพัฒนาจึงต้องขึ้นอยู่กับผู้พัฒนาในส่วนงานนั้นสามารถให้เวลาในการพัฒนาได้

7.3 แนวทางการพัฒนาต่อในอนาคต

1) เนื่องจากตัวแปลภาษาเอนเอเอสแอลยังไม่สามารถทำการแปลภาษาซีออกมาได้เนื่องจากในขั้นตอนการสร้างโปรแกรมภาษาซีนั้นจะต้องมีการเรียกใช้งานไลบาร์ลี่ที่เกี่ยวข้องกับการเขียนโปรแกรมผ่านเครือข่าย หากมีผู้ที่มีความสามารถในการเขียนโปรแกรมผ่านเครือข่ายให้เวลาในการสร้างฟังก์ชั่นการทำงานทางเครือข่ายที่เขียนโดยภาษาซีที่จะถูกเรียกใช้ในเวลาแปลจากสื่อกลางไปเป็นภาษาซี
2) เนื่องมาจากปัญหาที่ไม่สามารถเรียกใช้งานโดยคอมมานโหมดได้เนื่องมาจากเกิดปัญหาด้านการจองพื้นที่ของโปรเซสหากมีการสร้างฟังก์ชั่นหน่วงการจองพื้นที่ให้สามารถเรียกใช้งานไม่พร้อมกันได้ก็จะสามารถช่วยให้ทำงานได้ อย่างไรก็ตามฟังก์ชั่นที่จองพื้นที่นี้จะต้องสร้างขึ้นแก้ไขทุกๆ ฟังก์ชั่นที่เกี่ยวข้องกับการจองพื้นที่ เช่น malloc หรือ strdup เป็นต้น
3) ให้สร้างตัวแปรภาษาเอนเอเอสแอลเป็นเครื่องมือคล้าย IDE (Integrate Development Environment) ที่มีให้พัฒนาโปรแกรมทั่วไป ซึ่งหากสามารถพัฒนาภาษาได้โดยง่ายก็จะมีผู้สนใจทำการพัฒนา สคริปเอนเอเอสแอล ขยายเป็นวงกว้างมากขึ้น
4) นำไปใช้เป็นส่วนหนึ่งในการพัฒนาโปรแกรม Nessus ส่วนแม่ข่าย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

หนังสืออ้างอิง

[1] Alfred Aho, Ravi Sethi, Jeffrey Ullman, "Compilers Principles, Techniques abd Tools", Addison Wesley, 1999

[2] John Levine, Tony Mason, Doug Brown, "lex & yacc", O'Reilly, 1992

[3] Charles Donnelly, Richard Stallman, "Bison", Free Software Foundation, 2002

[4] Vern Paxson, "Flex, version 2.5", Free Software Foundation, 1995

[5] Richared Stevens, "Advanced Programming in the UNIX Environment", Pearson Education, 1993

[6] Microsoft Corporation, "UNIX Application Migration Guide", Microsoft, 2003

[7] ดวงแก้ว สวามิภักดิ์, "ระบบดำเนินงาน ยูนิกซ์", ซีเอ็ด, 2521

[8] วรวุฒิ เที่ยงธรรม, สันติ ศรีลาศักดิ์, "รู้จักลีนุกซ์", ออฟเซท เพรส, 2542

[9] ยุทธนา ลีลาศวัฒนกุล, "คู่มือการเขียนโปรแกรมและใช้งาน Visual C++6.0", อินโฟเพรส, 2544

[10] มัณฑนา ปราการสมุทร, "การเขียนชุดคำสั่งภาษาซี", ดวงกมลสมัย, 2534

เว็บไซต์อ้างอิง

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

[12] http://www.sourceforge.net

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

[14] http://www.nectec.or.th

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้