บทที่ 3
การศึกษาและวิเคราะห์ระบบป้องกันและตรวจสอบการบุกรุกระบบเครือข่าย

ระบบป้องกันและตรวจสอบผู้บุกรุกระบบเครือข่าย เป็นการนำเอาเทคโนโลยีทางด้านระบบป้องกันการบุกรุก และระบบตรวจสอบผู้บุกรุกเครือข่าย ที่เป็นโอเพ่นซอร์สมาประยุกต์ใช้ร่วมกันบนระบบปฏิบัติการลีนุกซ์ เพื่อให้ได้ระบบที่มีประสิทธิภาพในการเตือนภัยและป้องกันการบุกรุกระบบบนเครือข่ายให้ดีขึ้น โดยการออกแบบให้ระบบสามารถตรวจจับและป้องกันการบุกรุกได้อย่างอัตโนมัติ และสามารถตรวจสอบรายงานการบุกรุกที่เกิดขึ้นได้อย่างง่ายดายจากทางเว็บเพ็จ

โครงสร้างของระบบป้องกันและตรวจสอบผู้บุกรุกระบบเครือข่าย บนระบบปฏิบัติการลีนุกซ์ประกอบด้วย 4 ส่วนหลัก ได้แก่

3.1 ระบบตรวจสอบผู้บุกรุกระบบเครือข่าย

3.2 ระบบรักษาความปลอดภัยไฟร์วอลล์

3.3 ส่วนประสานการทำงานระบบตรวจสอบผู้บุกรุกเครือข่ายและระบบรักษาความปลอดภัยไฟร์วอลล์

3.4 ส่วนแสดงผลและสืบค้นข้อมูล

3.1 ระบบตรวจสอบผู้บุกรุกระบบเครือข่าย

สำหรับโปรแกรมที่นำมาใช้เป็นระบบตรวจสอบผู้บุกรุกระบบเครือข่ายสำหรับการศึกษาระบบป้องกันและตรวจสอบผู้บุกรุกเครือข่ายนี้คือโปรแกรม Snort เวอร์ชัน 2.0.0 ซึ่งมีส่วนประกอบที่สำคัญ 4 ส่วนได้แก่

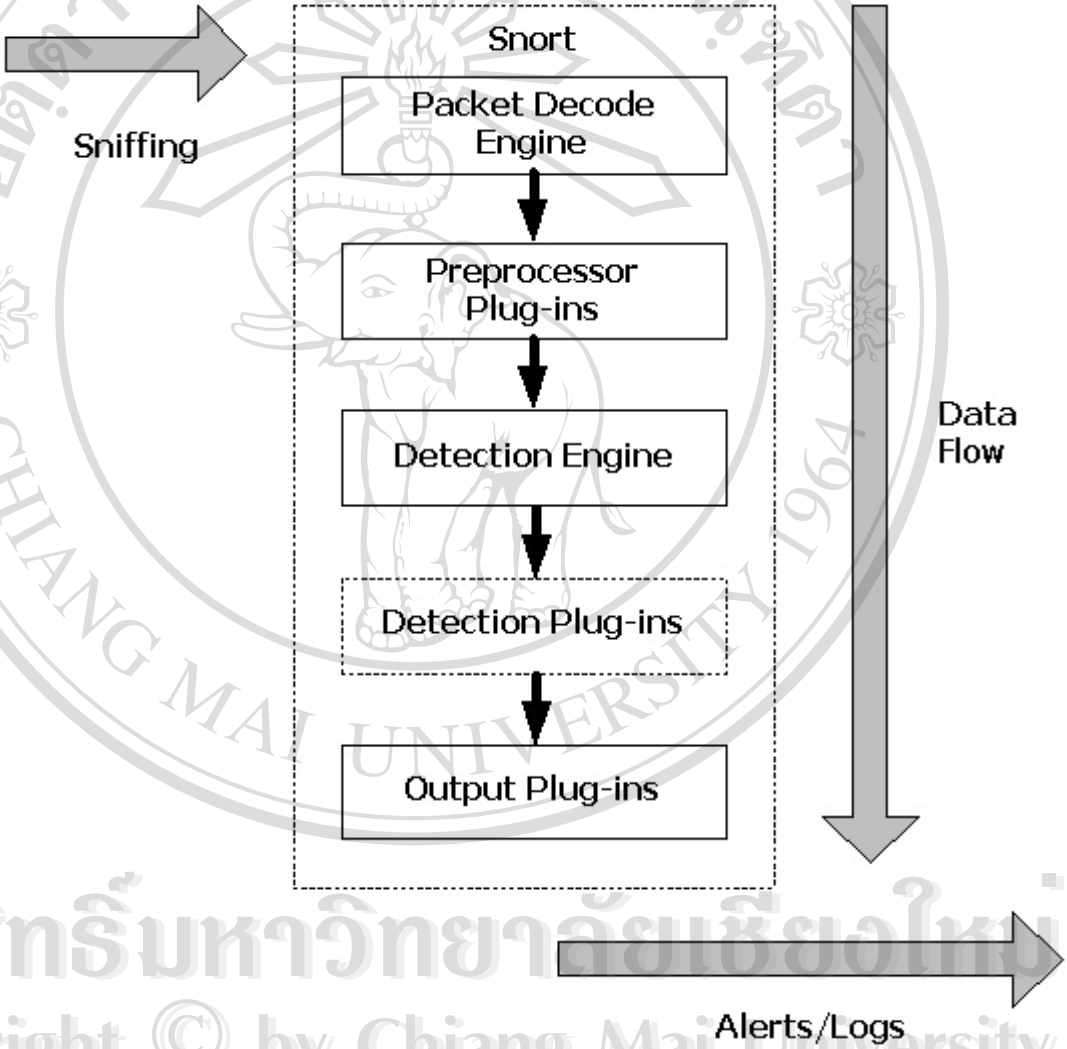
3.1.1 Packet Decode Engine ทำหน้าที่รับข้อมูลจากส่วนที่เชื่อมต่อกับระบบเครือข่ายผ่านทางไลบารี Libpcap[1] แล้วทำการถอดรหัส (Decode) ข้อมูลเพื่อส่งไปยังส่วนอื่นต่อไป

3.1.2 Preprocessor Plug-ins เป็นส่วนที่ทำหน้าที่รับแพ็กเก็ตข้อมูลที่ผ่านการถอดรหัสเข้ามาเพื่อเตรียมความพร้อมแพ็กเก็ตก่อนส่งไปตรวจสอบยังส่วน Detection Engine ต่อไป

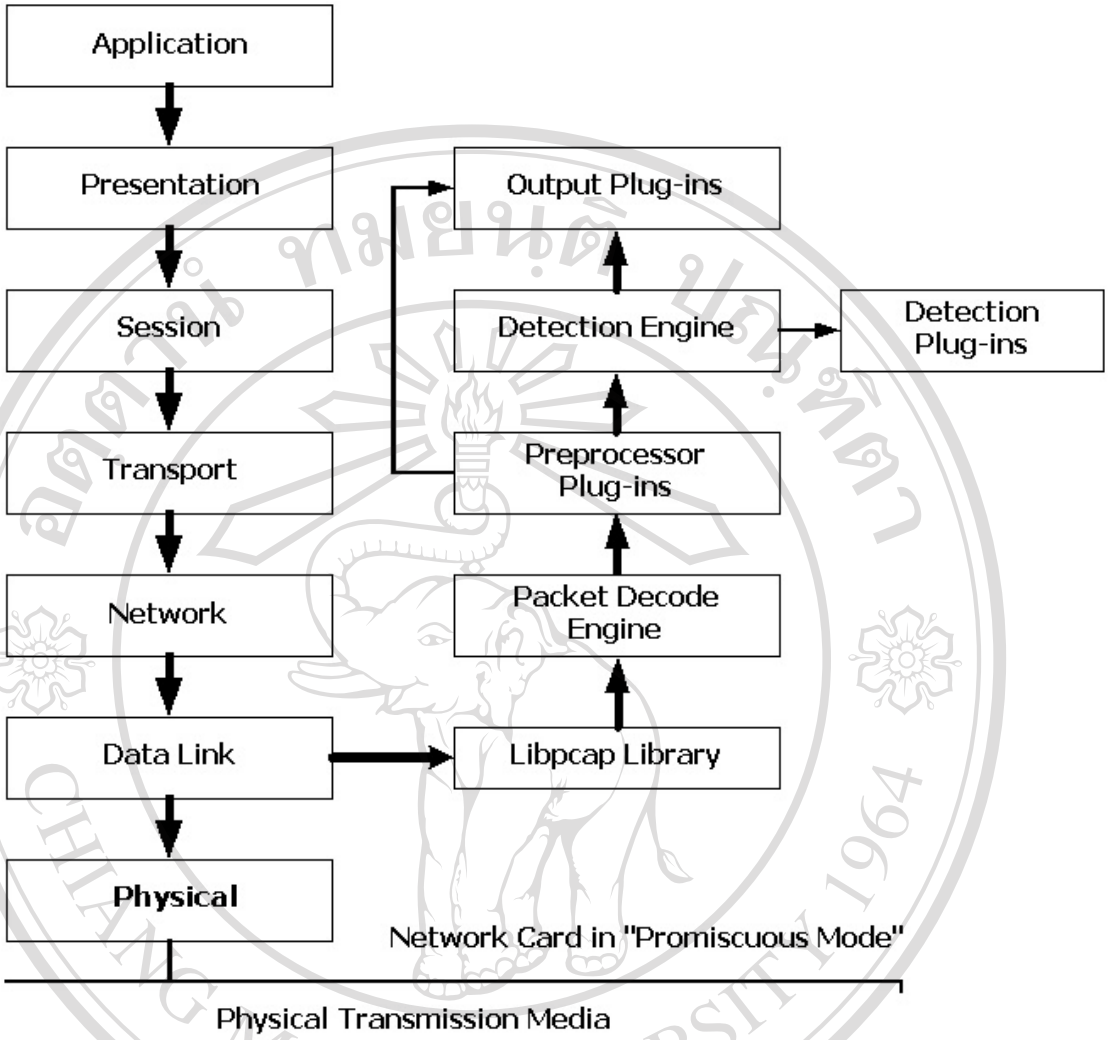
[1]ไลบารี Libpcap ถูกเขียนขึ้นมาเพื่อเป็นส่วนของในโปรแกรมประเภทดักจับข้อมูลที่เรียกว่า TCPDump ซึ่งอนุญาตให้ผู้พัฒนานำไปเขียนโค้ดเพื่อรับแพ็กเก็ตจากเลเยอร์ Data Link บนระบบปฏิบัติการยูนิกซ์ โดยไม่ขึ้นกับ Network Interface Card (NIC) และไดร์เวอร์ที่ต่างกันระบบปฏิบัติการเหล่านั้น

3.1.3 Detection Engine เป็นส่วนที่ทำหน้าที่ตรวจสอบแต่ละแพ็กเก็ตว่าเป็นการบุกรุกระบบเครือข่ายหรือไม่

3.1.4 Output Plug-ins เป็นส่วนที่ส่งผลของระบบตรวจสอบผู้บุกรุกออกไปในรูปแบบต่างๆ เช่น การแจ้งเตือน (Alerts) หรือการเก็บบันทึกข้อมูล (Logs) เป็นต้น



รูปที่ 3.1 ภาพแสดงการทำงานส่วนประกอบหลักของโปรแกรม Snort



รูปที่ 3.2 ภาพแสดงการทำงานของโปรแกรม Snort เทียบกับโมเดล OSI

3.1.1 Packet Decode Engine

แพ็กเก็ตที่วิ่งอยู่บนระบบเครือข่ายจะถูกดักจับเข้าสู่ระบบตรวจสอบผู้บุกรุกผ่านทางการทำงานในส่วนนี้ โดยอาศัยกลไกดังนี้

- เซ็ท Network Interface Card (NIC) ให้อยู่ในโหมด Promiscuous[2]
- ดักจับแพ็กเก็ตเข้าสู่ Network Interface Card ผ่านทางไลบารี Libpcap

[2]ปกติพฤติกรรมการทำงานของ Network Interface Card จะไม่สนใจต่อข้อมูลที่ไม่ได้มีจุดหมายมายัง Mac Address ของตนเอง การเปลี่ยนโหมดการทำงานให้อยู่ในโหมด Promiscuous จะทำให้ Network Interface Card ไม่ต้องเช็คหมายเลข Mac Address ปลายทางของแพ็กเก็ตที่รับเข้ามา ซึ่งจะทำให้สามารถเห็นแพ็กเก็ตทั้งหมดที่วิ่งอยู่บนเครือข่ายได้

เมื่อแพ็กเก็ตข้อมูลเดินทางจากระบบเครือข่ายเข้ามายัง NIC ที่ทำงานในโหมด Promiscuous จะส่งต่อไปยังส่วน Packet Decode Engine โดยผ่านทางไลบารี Libpcap เพื่อทำการถอดรหัสข้อมูลที่รับมาจากเลเยอร์ Data Link ซึ่งจะทำให้สามารถแยกประเภทของโปรโตคอลต่างๆในระดับเลเยอร์นี้ได้ เช่น Ethernet, 802.11 หรือ Token Ring เป็นต้น รวมถึงโปรโตคอลในระดับที่สูงขึ้นไปด้วย ได้แก่ IP, TCP และ UDP ในระหว่างกระบวนการถอดรหัสนี้ Decode Engine จะทำการลิงค์ข้อมูลดิบที่ได้มาไปยังโครงสร้างสำหรับการวิเคราะห์ต่อภายหลังในส่วนของ Preprocessors และ Detection Engine ต่อไป

3.1.2 Preprocessors

ก่อนที่แพ็กเก็ตจะถูกส่งไปตรวจสอบต่อยังส่วน Detection Engine จะต้องผ่านส่วน Preprocessor นี้ก่อน เพื่อตรวจสอบความถูกต้องเบื้องต้นของข้อมูลและจัดโครงสร้างข้อมูลให้อยู่ในรูปแบบที่พร้อมถูกตรวจสอบในส่วนต่อไป โดยสามารถแยกเป็นส่วนการทำงานย่อยได้ดังนี้

1.) The _decode Family of Preprocessors

หน้าที่ประการหนึ่งของ Preprocessor คือการทำให้แพ็กเก็ตอยู่ในรูปแบบที่เป็นบรรทัดฐานก่อนส่งต่อไปยังส่วน Detection Engine ตัวอย่างในกรณีของ HTTP Request เมื่อผู้ใช้เปิดโปรแกรมเว็บเบราว์เซอร์แล้วพิมพ์ URI (Universal Resource Identifier) เช่น http://10.1.1.231/scripts/hackme.exe ลงไป โปรโตคอล HTTP จะแปลงอักษรไบนารีเหล่านั้นให้อยู่ในรูปแบบของ %XX โดย XX เป็นค่าเลขฐานสิบหกของตัวอักษรนั้นๆ ฉะนั้นจาก URI ข้างต้นจะถูกแปลงเป็นhttp://10.1.1.231/%73%63%72%69%70%74%73/%68%61%63%6B%6D%65.%65%78%65 และเมื่อ HTTP Request นี้ถูกส่งถึงเว็บเซิร์ฟเวอร์ ส่วนของ URI นี้จะถูกแปลงกลับเป็น http://10.1.1.231/scripts/hackme.exe ตามเดิม

สำหรับในโปรแกรม Snort หน้าที่นี้จะเป็นของ Http_decode Preprocessor ในการแปลง URI ก่อนที่แพ็กเก็ตจะถูกส่งไปยัง Detection Engine เพื่อให้อยู่ในรูปแบบที่พร้อมถูกตรวจสอบโดยรูปแบบที่เราเข้าใจและสามารถกำหนดขึ้นได้เอง

นอกจากนี้ยังมี Decode Preprocessor อื่นๆอีกได้แก่ Telnet_decode, Ftp_decode และ Rpc_decode ที่ทำหน้าที่เช่นเดียวกับ Http_decode ในการแปลงแพ็กเก็ตให้อยู่ในรูปแบบที่เป็นบรรทัดฐานของบริการ Telnet, FTP และ RPC ตามลำดับ ก่อนจะถูกส่งต่อไปยัง Detection Engine

2.) The Frag2 Preprocessor

การแฟร็กเมนเตชัน (Fragmentation) ถือเป็นเรื่องปกติที่จำเป็นในการส่งข้อมูลเพื่อติดต่อสื่อสารกันบน Internet Protocol (IP) เพราะเนื่องจากดาตาแกรม[3] (Datagram) ที่ถูกส่งจาก IP ถ้ามีขนาดใหญ่กว่า MTU[4] ในเลเยอร์ Physical นั้น โปรโตคอล IP จะทำการแตกข้อมูลดาตาแกรมให้เป็นส่วนเล็กๆก่อนส่งเข้าไปยังอินเตอร์เฟสของเครือข่ายนั้นๆ เมื่อถึงปลายทางแพ็กเก็ตที่ถูกแบ่งเป็นส่วนเล็กๆเหล่านี้ก็จะถูกประกอบกลับคืนมาอีกครั้ง เรียกว่าการรีแอสเซมเบิล (Reassemble)

การแบ่งแพ็กเก็ตออกเป็นส่วนย่อยๆ สามารถใช้เป็นวิธีหนึ่งในการเลี่ยงผ่านการตรวจสอบจากระบบตรวจสอบผู้บุกรุกเครือข่ายประเภท Pattern-based หรือ Signature-based ได้ด้วย เพราะการแบ่งแพ็กเก็ตเป็นส่วนๆจะทำให้เกิดความยุ่งยากในการตรวจสอบสำหรับระบบตรวจสอบผู้บุกรุกประเภทนี้ ซึ่งจะเปรียบเทียบแพ็กเก็ตแบบทีละรายๆกับรูปแบบการบุกรุกที่เตรียมไว้อ้างอิง เครื่องมือประเภทนี้ที่ผู้บุกรุกนิยมใช้ได้แก่โปรแกรม Fragroute เพื่อหลีกเลี่ยงการตรวจจับของระบบตรวจสอบผู้บุกรุก

ในส่วนของ Frag2 Proprecessor ได้มีการระบุถึงรูปแบบการบุกรุกชนิดนี้ไว้ใน spp_frag2.c ซึ่งจะทำการประกอบแพ็กเก็ตที่ถูกแบ่งเหล่านั้นให้สมบูรณ์ขึ้นมาก่อนที่จะส่งไปตรวจสอบยังส่วน Detection Engine ต่อไป

นอกจากนี้การแฟร็กเมนเตชันยังสามารถใช้บุกรุกเข้ามาแบบ Denial Of Service (Dos) ได้ด้วย เนื่องจากการแตกแพ็กเก็ตเป็นส่วนย่อยๆก่อนส่งนั้น เครื่องที่ปลายทางจะต้องมีการเก็บแพ็กเก็ตที่ทยอยกันมาถึงเหล่านั้น เพื่อประกอบกลับเป็นแพ็กเก็ตที่สมบูรณ์ เพื่อใช้ในการพิจารณาต่อไป ซึ่งผู้บุกรุกอาจจะส่งชุดของแพ็กเก็ตที่ถูกแฟร็กเมนต์ไปยังเครื่องเป้าหมายเป็นจำนวนมาก ทำให้เกิด Stack Overflow ขึ้น อาจมีผลให้เครื่องเป้าหมายหยุดทำงาน หรือทำงานแบบผิดปกติไปได้ และอาจมีผลต่อระบบตรวจสอบผู้บุกรุกเองด้วย เพราะยิ่งบนระบบเครือข่ายมีปริมาณแพ็กเก็ตที่ถูกแฟร็กเมนต์อยู่จำนวนมากๆ ก็ยิ่งทำให้ระบบตรวจสอบผู้บุกรุกต้องใช้ทรัพยากรอย่างมากในการเก็บและประกอบแพ็กเก็ตเหล่านั้นขึ้นมาเพื่อใช้ในการพิจารณาตรวจสอบไปด้วย ซึ่งถ้าระบบตรวจสอบผู้บุกรุกมีทรัพยากรไม่พอ ก็จะทำงานไม่ทัน การตรวจสอบการบุกรุกก็จะขาดประสิทธิภาพลง เครื่องมือที่ผู้บุกรุกนิยมใช้ในการบุกรุกแบบนี้ได้แกโปรแกรม Stick ซึ่งจะใช้การ

[3]ชื่อเรียกแพ็กเก็ตที่ถูกส่งจาก Internet Protocol ในชั้นเลเยอร์ Network

[4]MTU หรือ Maximum Transfer Unit คือตัวเลขที่บอกถึงปริมาณข้อมูลที่สามารถบรรจุส่งได้สูงสุดบนระดับเลเยอร์ Physical นั้นๆ เช่น Ethernet มี MTU = 1500 ,FDDI มี MTU = 4352 เป็นต้น

โจมตีที่เลียนแบบการบุกรุกที่มีในฐานข้อมูลรูปแบบการบุกรุกของโปรแกรม Snort เข้ามาเพื่อให้ระบบตรวจสอบผู้บุกรุกตรวจเจอ และมีการแจ้งเตือนรวมถึงการบันทึกข้อมูลเป็นจำนวนมาก

3.) The Stream4 Preprocessor

Stream4 Preprocessor เป็นส่วนประกอบหลักที่ใหญ่ที่สุดในโปรแกรม Snort ถูกออกแบบมาให้สามารถรองรับการเชื่อมต่อได้มากกว่า 64,000 คอนเนคชันในเวลาเดียวกัน โดยมีวัตถุประสงค์หลัก 2 ประการคือ TCP Statefulness และ Session Reasembly

ในการเชื่อมต่อการทำงานของโปรโตคอล TCP จากเครื่องต้นทางไปยังปลายทาง จะเกิดการเชื่อมต่อแบบ Three-way Handshake[5]ขึ้น เมื่อการเชื่อมต่อสำเร็จ การส่งข้อมูลจึงเริ่มขึ้นและเมื่อเสร็จสิ้นการส่งข้อมูล การเชื่อมต่อก็จะสิ้นสุดลง โดยในระหว่างการเชื่อมต่อที่เกิดขึ้น Stream4 Preprocessor จะสร้างตารางสำหรับเก็บเซสชันต่างๆที่เกิดขึ้น (และลบเซสชันนั้นออกจากตารางเมื่อการเชื่อมต่อสิ้นสุดลง) เพื่อใช้ในการตรวจสอบเซสชันทั้งหมดของการเชื่อมต่อนั้นๆ โดยเมื่อส่วน Detection Engine จะทำการตรวจสอบแพ็กเก็ต จะมีการตรวจสอบก่อนว่าแพ็กเก็ตนั้นเป็นส่วนหนึ่งที่เกิดจากเชื่อมต่อเซสชันด้วยหรือไม่ ก่อนนำไปเปรียบเทียบกับรูปแบบการบุกรุกต่างๆต่อไป

ประโยชน์อีกประการหนึ่งของการทำงานแบบ Stateful ของโปรแกรม Snort คือสามารถตรวจสอบเทคนิคการสแกนแบบ Out-of-Sequence ได้ เช่น Stealth FIN Scan จากโปรแกรม NMAP เป็นต้น เนื่องจากโปรโตคอล TCP นั้นแพ็กเก็ต FIN จะถูกส่งเมื่อต้องการจบการเชื่อมต่อเท่านั้น ซึ่งถ้าแพ็กเก็ต FIN ถูกส่งมาเพื่อขอจบการเชื่อมต่อแบบ TCP ที่พอร์ตนั้น ฝั่งเครื่องปลายทางก็จะส่งแพ็กเก็ตบางอย่างตอบกลับไปยังเครื่องต้นทาง ซึ่งจะทำให้ผู้บุกรุกทราบสถานะของพอร์ตนั้นได้ว่าเปิดหรือปิดอยู่

4.) The Port Family of Preprocessors

ความสามารถที่สำคัญอีกประการหนึ่งของระบบตรวจสอบผู้บุกรุกคือความสามารถในการตรวจจับการสแกนพอร์ตได้ การถูกสแกนพอร์ตถือเป็นเหตุการณ์ปกติที่เกิดขึ้นได้กับทุกระบบเครือข่ายที่ทำการเชื่อมต่อกับอินเทอร์เน็ต ทำให้ผู้บุกรุกสามารถระบุเครื่องเป้าหมายและพอร์ตที่เปิดให้บริการอยู่ได้ ซึ่งจะนำไปสู่การพยายามบุกรุกเข้าทางจุดอ่อนของพอร์ตที่เปิดไว้ต่อไป ซึ่งจากการเชื่อมต่อเซสชันของโปรโตคอล TCP การสแกนพอร์ตจะกระทำเหมือนการเชื่อมต่อเซสชันตามปกติ โดยการส่งแพ็กเก็ต SYN ซึ่งเป็นแพ็กเก็ตที่ใช้ในการเริ่มต้นการติดต่อไปยังเครื่องปลายทางก่อน เมื่อเครื่องปลายทางได้รับแพ็กเก็ต SYN แล้วจะส่งแพ็กเก็ต SYN/ACK

[5] ดูรูปที่ 2.8 หน้า 38

กลับถ้าพอร์ตนั้นเปิดอยู่ และจะส่งแพ็กเก็ต SYN/RST กลับถ้าพอร์ตนั้นปิดอยู่ ฉะนั้นการส่งแพ็กเก็ต SYN ไปยังพอร์ตต่างๆที่เครื่องเป้าหมาย ก็จะสามารถทราบได้ว่าเครื่องนั้นเปิดพอร์ตไหนอยู่บ้าง โดยดูจากแพ็กเก็ตที่ตอบกลับมา

โปรแกรม Snort สามารถตรวจสอบการสแกนพอร์ตได้โดยใช้ Preprocessor ในส่วนนี้ ซึ่งในเวอร์ชัน 2.0.0 มีให้เลือกใช้งานอยู่ 2 ส่วนได้แก่ Portscan Preprocessor และ Portscan2 Preprocessor

โดย Portscan Preprocessor เป็นเวอร์ชันที่ถูกพัฒนาขึ้นมาก่อน ใช้วิธีตรวจสอบการสแกนโดยพิจารณาจากการเชื่อมต่อที่เกิดขึ้นจากเครื่องต้นทางหนึ่ง ไปยังหลายพอร์ตที่เครื่องปลายทาง ภายในช่วงเวลาใดเวลาหนึ่ง ส่วน Portscan2 Preprocessor เป็นเวอร์ชันที่เพิ่งถูกพัฒนาขึ้นมาภายหลัง (ซึ่งในอนาคตจะเป็นเวอร์ชันที่เป็นมาตรฐานที่ใช้ในการตรวจสอบการสแกนพอร์ตสำหรับโปรแกรม Snort) มีฟังก์ชันในการตรวจสอบคล้ายเวอร์ชันแรก แต่ได้มีการแก้ไขโค้ดโปรแกรมใหม่ ให้มีความสมบูรณ์ขึ้น ทำให้มีประสิทธิภาพในการตรวจสอบสูงกว่าเวอร์ชันแรก โดยส่วนที่สำคัญคือการใช้ปลั๊กอิน (Plug-in) ที่คล้ายกับ Stream4 ในการตรวจสอบสถานะการเชื่อมต่อด้วย ฉะนั้น Portscan2 Preprocessor จึงสามารถตรวจสอบได้ว่าแพ็กเก็ต SYN-ACK นั้นเป็นส่วนหนึ่งของการเชื่อมต่อ ณ เวลาขณะนั้นจริงหรือเป็นส่วนหนึ่งจากการสแกน ส่วนรูปแบบเอาท์พุตของ Portscan2 ยังให้รายละเอียดข้อมูลที่มากกว่า Portscan เวอร์ชันแรกด้วย และยังสนับสนุนการตรวจสอบโดยแยกคิดเวลา Threshold ของเครื่องและพอร์ตในเวลาที่กำหนดได้ด้วย

การตัดสินใจเลือกใช้ Preprocessor ตัวใดขึ้นอยู่กับความเหมาะสมของระบบ โดย Portscan Preprocessor เป็นการตรวจสอบการสแกนที่งานและมีความเสถียรสูง ส่วน Portscan2 Preprocessor เป็นเวอร์ชันที่ปรับปรุงใหม่ มีการแก้ไขออปชันบางอย่าง ค่า Threshold และสามารถตรวจจับการสแกนแบบ Stealth Scan ได้ดีกว่า แต่ก็มีการใช้ทรัพยากร (หน่วยความจำและหน่วยประมวลผลกลาง) ที่เยอะกว่าด้วย

ข้อจำกัดของการตรวจสอบ Portscan คือไม่สามารถตรวจจับการสแกนแบบ Slow Scan ได้ เช่นถ้าเครื่องเป้าหมายถูกสแกนอย่างช้าๆหรือหนึ่งพอร์ตต่อหนึ่งชั่วโมง ระบบจะไม่สามารถเก็บสถานะของเซสชันการสแกนพอร์ตนั้นได้ จึงตรวจไม่พบการสแกนพอร์ตนั้นๆ

Stealth Portscaning

ภายใน TCP Header จะมีส่วนของ Flag ที่ใช้ในการระบุสถานะต่างๆของแพ็กเก็ต ซึ่งประกอบด้วย URG, ACK, PSH, RST, SYN และ FIN ซึ่งใช้สำหรับสถานะที่ต่างกันไปตามแต่ละเซสชัน[6] ซึ่งผู้บุกรุกสามารถสร้างแพ็กเก็ตที่มีค่า Flag ต่างๆส่งให้เครื่องปลายทาง แล้วเช็คค่า Flag ในแพ็กเก็ตที่ส่งตอบกลับมา ก็จะทราบถึงสถานะของพอร์ตนั้นๆได้ และเป็นการพยายามหลีกเลี่ยงการถูกตรวจจับโดยโปรแกรมตรวจสอบการสแกนพอร์ตด้วย

ตัวอย่างการสแกนโดยการเซ็ทค่า TCP Flag ต่างๆกัน เช่น

- Full XMAS Scan โดยการเซ็ทค่า TCP Flag เป็น FIN, URG, PSH ซึ่งจะทำให้เครื่องปลายทางส่ง RST ตอบกลับมาถ้าพอร์ตนั้นปิดอยู่
- TCP FIN Scan โดยการเซ็ทค่า TCP Flag เป็น FIN เช่นเดียวกับ XMAS Scan ซึ่งจะทำให้เครื่องปลายทางส่ง RST ตอบกลับมาถ้าพอร์ตนั้นปิดอยู่
- NULL Scan โดยการเซ็ทค่า TCP Flag เป็น no option เช่นเดียวกับ XMAS Scan ซึ่งจะทำให้เครื่องปลายทางส่ง RST ตอบกลับมาถ้าพอร์ตนั้นปิดอยู่

5.) The Other Preprocessors

PreProcessors	หน้าที่
Rpc_decode	Decodes RPC, similar to the HTTP decoder
Telnet_decode	Decodes Telnet and FTP, similar to the HTTP decoder
Conversation	Provides basic conversation status on protocols (used by the portscan2 preprocessor)
Back Orifice detector	Decodes Back Orifice network traffic
Arpspoof	Experimental ARP misuse detection code
Asn1_decode	Experimental ASN1 detection code
Fnord	Polymorphic shellcode analyzer and detector

ตารางที่ 3.1 Preprocessors ประเภทอื่นๆ[7]

[6] ดูตารางที่ 2.1 หน้า 39

[7] Brian Caswell, Snort 2.0 Intrusion Detection (United States of America : Syngress, 2003), pp. 114

3.1.2 Detection Engine

แพ็กเก็ตที่รับเข้ามาจากเครือข่าย หลังจากผ่านการทำงานในสองส่วนแรกแล้ว จะถูกเก็บเข้าสู่โครงสร้างข้อมูล (Data Structure) ซึ่งผ่านกระบวนการจัดโครงสร้าง การกลั่นกรอง และการถอดรหัส มาอยู่ในรูปแบบที่พร้อมต่อการถูกตรวจสอบต่อไปในส่วน Detection Engine นี้

Detection Engine ถือเป็นส่วนที่สำคัญที่สุดของระบบตรวจสอบผู้บุกรุก เพราะทำหน้าที่รับผิดชอบต่อการตรวจจับพฤติกรรมการบุกรุกต่างๆที่ปรากฏอยู่ในแพ็กเก็ต ซึ่งในโปรแกรม Snort นี้ส่วน Detection Engine จะใช้วิธีตรวจสอบแบบอ้างอิงกฎ (Rule-base) จากเทคนิคการบุกรุกต่างๆที่ทราบ โดยกฎข้อต่างๆจะถูกนำเปรียบเทียบกับข้อมูลของแพ็กเก็ตที่ถูกเก็บอยู่ในโครงสร้างข้อมูล ถ้าแพ็กเก็ตใดตรงกับรูปแบบการบุกรุกตามกฎข้อใดข้อหนึ่ง ก็จะเกิดการกระทำบางอย่างขึ้น เช่น การเก็บบันทึกเหตุการณ์ (Logging) การแจ้งเตือน (Alert) หรือมิฉะนั้นแพ็กเก็ตนั้นก็จะถูกดร็อปไป

Detection Engine จะเป็นการทำงานแบบ ณ เวลาที่เกิดขึ้นจริง (Real-time) ประสิทธิภาพการทำงานในส่วนนี้จะขึ้นอยู่กับองค์ประกอบดังต่อไปนี้

- สมรรถนะของเครื่องที่โปรแกรม Snort ทำงานอยู่
- จำนวนของกฎที่ใช้ในการเปรียบเทียบข้อมูลการบุกรุก
- ความเร็วของช่องทางที่เชื่อมต่อระบบเครือข่ายของเครื่องที่โปรแกรม Snort ทำงานอยู่
- ความหนาแน่นของข้อมูลบนเครือข่าย

ฉะนั้นการออกแบบระบบตรวจสอบผู้บุกรุกบนเครือข่าย นอกจากต้องคำนึงถึงตำแหน่งการวางระบบแล้ว ยังต้องพิจารณาองค์ประกอบดังกล่าวด้วย

ระบบตรวจสอบแพ็กเก็ตในส่วนนี้จะสามารถจำแนกส่วนต่างๆของแพ็กเก็ต เพื่อใช้ในการนำไปเปรียบเทียบกับกฎต่างๆได้ โดยสามารถแยกส่วนต่างๆของแพ็กเก็ตที่ใช้ในการวิเคราะห์ได้เป็น

- IP Header
- Transport Layer Header (TCP Header และ UDP Header)
- Application Layer Header (เช่น DNS Header, FTP Header, SNMP Header เป็นต้น)
- Packet Payload (ส่วนข้อมูลของแพ็กเก็ต)

โดยใช้วิธีการตรวจสอบกฎทั้งหมดที่มีกับแพ็กเก็ตหนึ่งๆ ซึ่งถ้าแพ็กเก็ตนี้ตรงกับกฎมากกว่าหนึ่งข้อ การแจ้งเตือนจะดูจากกฎข้อที่มีลำดับความสำคัญ (Priority) ที่สูงกว่าเท่านั้น

กฎต่างๆที่ใช้เป็นตัวอ้างอิงเปรียบเทียบเพื่อตรวจสอบแพ็กเก็ตนั้น ถูกเก็บอยู่ในลักษณะของไฟล์ข้อความ (Text File) โดยมีการแบ่งกลุ่มตามประเภทของการบุกรุก เช่นไฟล์ ftp.rules จะเก็บข้อมูลการบุกรุกที่เข้ามาทางบริการ FTP และจุดบกพร่องต่างๆของโปรแกรมสำหรับบริการนี้ เป็นต้น โดยกฎทั้งหมดประกอบด้วยไฟล์ดังต่อไปนี้

attack-responses.rules	icmp-info.rules	p2p.rules	telnet.rules
backdoor.rules	icmp.rules	policy.rules	tftp.rules
bad-traffic.rules	imap.rules	pop2.rules	virus.rules
chat.rules	info.rules	pop3.rules	web-attacks.rules
ddos.rules	local.rules	porn.rules	web-cgi.rules
deleted.rules	misc.rules	rpc.rules	web-client.rules
dns.rules	multimedia.rules	rservices.rules	web-coldfusion.rules
dos.rules	mysql.rules	scan.rules	web-frontpage.rules
experimental.rules	netbios.rules	shellcode.rules	web-iis.rules
exploit.rules	nntp.rules	smtp.rules	web-misc.rules
finger.rules	oracle.rules	snmp.rules	web-php.rules
ftp.rules	other-ids.rules	sql.rules	x11.rules

กฎแต่ละข้อมีโครงสร้างหลักสองส่วน ได้แก่ Rule Header และ Rule Options ดังรูปที่ 3.3

Rule Header	Rule Options

รูปที่ 3.3 โครงสร้างหลักของกฎในโปรแกรม Snort

ในส่วน Rule Header ประกอบด้วยโครงสร้างย่อย ดังรูปที่ 3.4

Action	Protocol	Source Address	Port	Direction	Destination Address	Port

รูปที่ 3.4 โครงสร้างส่วนประกอบย่อยของ Rule Header

1. Action เป็นส่วนที่ใช้ระบุประเภทของ Action เมื่อข้อมูลภายในแพ็กเก็ตตรงตามเงื่อนไขของกฎ ประกอบด้วย

- Pass เป็น Action ให้เพิกเฉยต่อแพ็กเก็ตนี้ และปล่อยให้การตรวจสอบดำเนินต่อไปกับกฎข้อที่เหลือ

- Log เป็น Action ให้ทำการเก็บบันทึกข้อมูลของแพ็กเก็ตนั้น ตามที่กำหนดไว้ในไฟล์คอนฟิกุเรชันของตัวเซ็นเซอร์ เช่น เก็บบันทึกลงฐานข้อมูลต่างๆ เป็นต้น

- Alert เป็น Action ให้ทำการเก็บบันทึกแพ็กเก็ตนั้น คล้าย (Log Action) แต่มีการแจ้งเตือนภัยด้วย ตามที่กำหนดไว้ในไฟล์คอนฟิกุเรชัน เช่น ส่งสัญญาณเตือนภัยไปเก็บยังไฟล์หรือไปยังคอนโซล เป็นต้น

- Activate เป็น Action ที่ใช้เมื่อต้องการให้ตรวจสอบแพ็กเก็ตนี้ด้วยกฎข้ออื่น (ที่ใช้ Dynamic Action) อีก

- Dynamic เป็น Action ที่ถูกใช้โดยกฎข้ออื่นที่ใช้ Activate Action เพื่อให้ทำการตรวจสอบแพ็กเก็ตจากกฎข้อนั้นด้วยกฎในข้อที่เป็น Dynamic Action อีก

2. Protocol เป็นส่วนที่ใช้ระบุชนิดโปรโตคอลของแพ็กเก็ตที่กฎจะใช้พิจารณาตรวจสอบ ประกอบด้วยโปรโตคอล IP, ICMP, TCP และ UDP

3. Address เป็นส่วนที่ใช้ระบุแอดเดรส ประกอบด้วยสองส่วนได้แก่

- Source Address ใช้ระบุแหล่งที่มาของแพ็กเก็ต

- Destination Address ใช้ระบุแหล่งปลายทางของแพ็กเก็ต

ซึ่งการระบุแอดเดรสของแพ็กเก็ตสามารถกำหนดได้ทั้งแบบหมายเลขไอพีแอดเดรส (IP Address) และหมายเลขเน็ตเวิร์ค (Network ID) โดยสามารถระบุได้ตามรูปแบบ Classless Inter Domain Routing Address (CIDR) ดังตารางที่ 3.2

CIDR Block	Subnet Mask	Number of Class C Address	Hosts
/14	255.252.0.0	1024	262144
/15	255.254.0.0	512	131072
/16	255.255.0.0	256	65536
/17	255.255.128.0	128	32768
/18	255.255.192.0	64	16384
/19	255.255.224.0	32	8192
/20	255.255.240.0	16	4096
/21	255.255.248.0	8	2048
/22	255.255.252.0	4	1024
/23	255.255.254.0	2	512
/24	255.255.255.0	1	256

/25	255.255.255.128	?	128
/26	255.255.255.192	?	64
/27	255.255.255.224	1/8	32
/28	255.255.255.240	1/16	16
/29	255.255.255.248	1/32	8
/30	255.255.255.252	1/64	4
/31	255.255.255.254	1/128	2
/32	255.255.255.255	1/256	1

ตารางที่ 3.2 CIDR Block Addressing ที่มา: http://www.rfc-editor.org/rfcsearch.html

เช่น แอดเดรส 192.168.1.3/32 ใช้ระบุแทนหมายเลขไอพีแอดเดรส 192.168.1.3 แอดเดรส 192.168.1.0/24 ใช้ระบุแทนช่วงหมายเลขไอพีแอดเดรสตั้งแต่ 192.168.1.0-192.168.1.255 เป็นต้น
หมายเหตุ : สามารถใช้คีย์เวิร์ด Any แทนทุกหมายเลขแอดเดรสได้

4. Port Number ใช้ระบุหมายเลขพอร์ตในแพ็กเก็ตจากต้นทางไปยังปลายทาง โดยสามารถระบุแบบเจาะจงหรือระบุเป็นช่วงหมายเลขพอร์ตก็ได้ เช่น ถ้าระบุ Source port 23 หมายถึง การตรวจสอบแพ็กเก็ตที่เริ่มต้นมาจาก Telnet เซิร์ฟเวอร์

วิธีระบุหมายเลขพอร์ตได้แก่

- Port Ranges เช่น ตั้งแต่พอร์ต 1024-2048

alert udp any 1024:2048 -> any any (msg: "udp ports";)

- Upper and Lower Boundaries เช่น ช่วงตั้งแต่ 0-1024 ระบุโดยใช้ :1024 และช่วงตั้งแต่พอร์ต 1000 ขึ้นไป ระบุโดยใช้ 1000:

alert tcp any :1024 -> any :1024 (msg: "tcp ports 0-1024";)

alert tcp any any -> any 1000: (msg: "tcp upper 1000 ports";)

- Negation Symbol เช่น ต้องการระบุพอร์ตทุกพอร์ตยกเว้นพอร์ตที่ระบุ

log upd any any -> any !23 (msg: "Everything but upd port 23";)

5. Direction ใช้ระบุทิศทางในการพิจารณาแหล่งที่มาและปลายทางของแพ็กเก็ต โดยสัญลักษณ์ที่ใช้ได้แก่

- -> ด้านซ้ายของสัญลักษณ์ใช้แทนแหล่งมา (Source Address, Port) ส่วนด้านขวาใช้แทนปลายทาง (Destination Address, Port) ของแพ็กเก็ต

- <- ด้านขวาของสัญลักษณ์ใช้แทนแหล่งมา (Source Address, Port) ส่วนด้านซ้ายใช้แทนปลายทาง (Destination Address, Port) ของแพ็กเก็ต

- <> ใช้เมื่อต้องการพิจารณาแพ็กเก็ตที่เคลื่อนที่ไปในทั้งสองทิศทาง

โครงสร้างของ Rule Options ระบุถึงรายละเอียดต่างๆภายในแพ็กเก็ตที่จำเป็นต่อการตรวจสอบ เช่น การระบุค่าภายในฟิลด์และโปรโตคอลต่างๆ โครงสร้างภายในประกอบด้วยส่วนย่อย เรียกว่า Section แต่ละ Section ถูกแบ่งโดยเครื่องหมาย ; (Semicolon) และประกอบด้วยสองส่วนได้แก่ คีย์เวิร์ด (Keyword) และ อาร์กิวเม็นต์ (Argument) แยกจากกันโดยเครื่องหมาย : (Colon) เช่น msg: "Detected confidential"; คีย์เวิร์ดของ Session นี้คือ msg ส่วนอาร์กิวเม็นต์คือ "Detection confidential"

รูปแบบต่างๆในการกำหนด Rule Option ได้แก่

1. Rule Content เป็นคีย์เวิร์ดที่ใช้ในการตรวจสอบส่วนเนื้อหาของแพ็กเก็ต เพื่อตรวจสอบหาเนื้อหาที่เป็นการบุกรุก

1.1 ASCII Content การระบุเนื้อหาที่ต้องการตรวจสอบแบบแอสกี โดยระบุได้ภายในเครื่องหมาย " (Quotation) ปกติภายในกฎข้อหนึ่งสามารถระบุเนื้อหาแบบนี้ได้เพียงแบบเดียวเท่านั้น ตัวอย่างเช่น

Alert tcp any any -> any any (**content: "malicious string /etc/passwd"**; msg: "Searching for ASCII Garbage!" ;)

ซึ่งเป็นกฎที่ใช้ในการค้นหาสติง "malicious string /etc/passwd" ภายในส่วนเนื้อหาของแพ็กเก็ต

1.2 Binary Content เป็นการระบุเนื้อหาที่ต้องการตรวจสอบแบบไบนารี โดยระบุภายในเครื่องหมาย | (Pipe) ตัวอย่างเช่น

Alert tcp any any -> any any (**content: "|0000 0101 EFFF|"**; msg: "Searching for Garbage!";)

ซึ่งข้อมูลที่เป็นไบนารีสามารถถูกดักจับได้ง่าย และเป็นวิธีที่นิยมใช้กันในโปรแกรมดักจับข้อมูล (Sniff) เช่น TCPDump, Ethereal, Iris เป็นต้น

1.3 ASCII and Binary Content เป็นการระบุเนื้อหาที่ต้องการตรวจสอบทั้งแบบแอสกีและไบนารี ภายในกฎข้อเดียวกัน ตัวอย่างเช่น

Alert tcp any any -> any any (**content: "|0101 FFFF|/etc/passwd|E234|"**; msg: "Searching for Mixed Garbage!";)

1.4 The Offset Keyword ใช้ในการระบุตำแหน่งเริ่มต้นในการค้นหาหรือตรวจสอบเนื้อหาภายในแพ็กเก็ต ใช้เป็นออปชันร่วมกับออปชันหลักส่วนอื่น ตัวอย่างเช่น

Alert tcp 192.168.1.0/24 any -> any any (content: "HTTP"; **offset: 4**; msg: "HTTP matched";)

ใช้ในการค้นหาคำ "HTTP" ตั้งแต่ไบต์ที่สี่ของข้อมูลเนื้อหา

1.5 The Depth Keyword ใช้ในการระบุช่วงของเนื้อหาที่ต้องการตรวจสอบ (เป็นไบต์) ใช้เป็นออปชันเสริมร่วมกับออปชันหลักส่วนอื่นเพื่อช่วยในการ Minimize CPU และ Optimize Speed ของตัวเซ็นเซอร์ ตัวอย่างเช่น

Alert tcp 192.168.1.0/24 any -> any any (content: "HTTP"; offset: 4; **depth: 40**; msg: "HTTP matched";)

ใช้ในการค้นหาคำ "HTTP" ระหว่างตัวอักษรที่ 4 ถึง 40 ของในเนื้อหา

1.6 The Nocase Keyword เป็นออปชันหนึ่งในสามตัวที่สามารถใช้ร่วมกับ Keyword Content อื่นอีกสองตัวได้แก่ Offset และ Depth ใช้ในการค้นหาข้อมูลในแพ็กเก็ตแบบ Intensive เหมาะสำหรับการตรวจสอบแพ็กเก็ตของโปรโตคอล TCP เช่น บริการ Telnet, FTP เป็นต้น ตัวอย่างเช่น

Alert tcp any any -> any 23 (content: "administrator"; **nocase**;)

ใช้ตรวจสอบหาคำว่า "administrator" ภายในแพ็กเก็ตบริการ Telnet

1.7 The Session Keyword ใช้ในการดักจับข้อมูลประเภทเคลียร์เท็กซ์ (Clear-Text) จากแพ็กเก็ตที่ใช้ในการติดต่อสื่อสารโดยโปรโตคอล TCP และสามารถแสดงผลออกมาได้สองแบบ ได้แก่แสดงข้อมูลในช่วงเวลาการติดต่อทั้งหมด (Session: ALL;) หรือให้แสดงเฉพาะข้อมูลที่สามารถพิมพ์ออกมาได้ (Session: printable;) ตัวอย่างเช่น

Alert tcp any any -> 192.168.1.0/24 110 (**session: printable**;)

ใช้ในการเก็บข้อมูลจากการเชื่อมต่อของบริการ POP3 แบบสามารถพิมพ์ได้

1.8 Uniform Resource Identifier Content ใช้ในการตรวจสอบข้อมูลในส่วน URI คือแทนที่จะทำการตรวจสอบเปรียบเทียบกับข้อมูลทั้งแพ็กเก็ต สามารถใช้ออปชันนี้ในการระบุเพื่อให้ตรวจสอบเฉพาะส่วนเรียกขอ (Request) ที่เป็น URI ของแพ็กเก็ตเท่านั้น ตัวอย่างเช่น

Alert tcp any any -> any 80 (msg: "WEB-CGI /cgi-bin/phf"; **uricontent: "/cgi-bin/phf"**;)

1.9 Regular Expressions เป็นการใช้สัญลักษณ์พิเศษช่วยในการเปรียบเทียบค้นหาคำในการตรวจสอบเนื้อหาในแพ็กเก็ต สามารถใช้ได้สองสัญลักษณ์ได้แก่ ? (Question mark) ซึ่งใช้แทนตัวอักษรใดหนึ่งตัว และ * (Asterisk) ใช้แทนตัวอักษรตั้งแต่หนึ่งตัวขึ้นไป ตัวอย่างเช่น

Alert tcp $OUTSIDE any -> $DMZ 80 (content: "../*../"; **regex**; msg: "Bad Example of a dot dot attack";)

1.10 Flow Control เป็นออปชันที่ใช้ในการกำหนดการเชื่อมต่อของแพ็กเก็ตบนโปรโตคอล TCP ใช้งานร่วมกับออปชันอื่น โดยการใช้งานจะอยู่ในรูปแบบ flow: <OPTION> โดยค่า OPTION ที่สามารถกำหนดได้มีดังตารางที่ 3.3

ค่าออปชัน	คำอธิบาย
to_server	Passes true on packets sent to the server
from_server	Passes true on packets sent from the server.
to_client	Passes true on packets sent to the client.
from_client	Passes true on packets sent from the client.
only_stream	Only activates on reconstructed packets or packets within an established stream.
no_stream	This instruction is the opposite of the previous example and does not pass packets that are recon-structed or within an established stream.
established	The established instruction will activate on packets that are part of an established TCP connection or session.
stateless	Modified from the original Snort stateless instruction, the Flow Control's stateless option is geared toward activating on packets regardless of state. Various static attacking tools such as stick send stateless packets in hopes of executing a system- or network-wide denial-of-service (DoS) attack. The stateless option must be used without the flow: prefix.

ตารางที่ 3.3 Flow Control ออปชัน

ตัวอย่างเช่น

alert tcp $DMZ any -> $EXTERNAL any (msg: "Server Potentially Sending Sensitive Info"; **flow: from_server**; content:"root:: ";)

ใช้ในการตรวจสอบแพ็กเก็ตที่ถูกส่งจากเครื่องเซิร์ฟเวอร์ที่มีการเรียกดูไฟล์รหัสผ่านของระบบปฏิบัติการยูนิกซ์ไปยังเครื่องภายนอก

1.11 Content-list Option เป็นออปชันที่ใช้ระบุชื่อไฟล์ที่เก็บคีย์เวิร์ดคำที่ต้องการค้นหาภายในส่วนเนื้อหาของแพ็กเก็ต ตัวอย่างเช่น

alert ip any any -> 192.168.1.0/24 any (**content_list: "porn"**; msg: "porn word matched";)

เป็นการค้นหาคำที่ระบุไว้ในไฟล์ชื่อ porn ประกอบด้วยคีย์เวิร์ดสามคำ ได้แก่ "porn" , "hardcore" และ "under 18" ในส่วนเนื้อหาของแพ็กเก็ต

2. IP Options เป็นคีย์เวิร์ดที่ใช้ในการตรวจสอบการบุกรุกที่มีพื้นฐานอยู่บนอินเทอร์เน็ตโปรโตคอล (IP)

2.1 Fragmentation Bits ในส่วนหัวของแพ็กเก็ต IP ประกอบด้วย Flag bit สามตัว ซึ่งใช้ในการแฟร็กเมนเตชันและรีแอสแซมบลีแพ็กเก็ต ได้แก่

- R (Reserved Bit) ซึ่งถูกสงวนไว้ใช้สำหรับการใช้งานในอนาคต
- D (Don't Fragment Bit) ถ้าบิตนี้ถูกเซ็ทแสดงว่าแพ็กเก็ต IP นี้ไม่ควรถูกแฟร็กเมนต์
- M (Move Fragment Bit) ถ้าบิตนี้ถูกเซ็ทแสดงว่ามีการแฟร็กเมนต์แพ็กเก็ตนี้หลายครั้งบนทางที่ถูกส่งไป ถ้าบิตนี้ไม่ถูกเซ็ทแสดงว่านี่เป็นการแฟร็กเมนต์ครั้งสุดท้าย (หรือครั้งเดียว) ของแพ็กเก็ต IP นี้

ซึ่งบรรดาผู้บุกรุกอาจใช้ประโยชน์จากบิตเหล่านี้ในการโจมตีหรือสืบหาข้อมูลระบบเครือข่ายได้ เช่น การเซ็ทบิต D สามารถใช้ในการตรวจสอบค่า MTU ต่ำสุดและสูงสุดบนเส้นทางจากเครื่องต้นทางไปยังเครื่องปลายทางนั้นๆได้ ฉะนั้นการใช้ออปชัน Fragment bit สามารถตรวจสอบได้ว่า Flag bit นั้นถูกเซ็ทหรือไม่ ตัวอย่างเช่น

alert icmp any any -> 192.168.1.0/24 any (**fragbits: D**; msg: "Don't fragment bit set";)

2.2 Equivalent Source and Destination IP Option ใช้ในการตรวจสอบแพ็กเก็ตที่มีการเปลี่ยนหรือปลอมแปลงให้มีแหล่งต้นทางและปลายทางเป็นที่เดียวกัน ตัวอย่างเช่น

alert ip any any -> any any (msg:" Same Source and Destination IP Address"; **sameip**;)

2.3 IP Protocol Option ใช้ในการตรวจสอบออปชันที่ถูกเพิ่มขึ้นมาในส่วนหัวของแพ็กเก็ต IP ซึ่งปกติจะมีแค่ 20 ไบต์ โดยการใช้งานจะอยู่ในรูปแแบ ipopts: <IP_OPTION>; โดยค่า IP_OPTION ที่สามารถกำหนดได้ มีดังตารางที่ 3.4

IP ออปชัน	คำอธิบาย
eol	Used to specify the end of an IP list
lsrr	IP loose source routing
nop	Used when there is no IP option set
rr	Record route

satid	The IP stream identifier
sec	The IP security option, also known as IPSec
ssrr	IP strict source routing
ts	The timestamp field

ตารางที่ 3.4 Snort IP ออปชัน ที่มา http://www.rfc-editor.org/rfc/rfc791.txt

ซึ่งผู้บุกรุกสามารถใช้ออปชันเหล่านี้ในการสำรวจข้อมูลบนระบบเครือข่ายได้เช่น การใช้ออปชัน lsrr และ ssrr ช่วยให้ผู้บุกรุกสามารถตรวจสอบได้ว่ามีเส้นทางนั้นบนระบบเครือข่ายอยู่จริงหรือไม่ ตัวอย่างเช่น

alert ip any any -> any any (**ipopts: lsrr**; msg: "Loose source routing attemp";)

2.4 ID Option ใช้ในการตรวจสอบค่าหมายเลขแฟร็กเมนต์ในฟิลด์ Fragment ID ในส่วนหัวของแพ็กเก็ต IP โดยการระบุค่าหมายเลขที่ต้องการตรวจสอบ ในรูปแบบ id: <ID_NUMBER>

2.5 Type of Service Option ใช้ในการตรวจสอบค่าที่ระบุในฟิลด์ Type of Service (TOS) ในส่วนหัวของแพ็กเก็ต IP ตัวอย่างเช่น

alert tcp $EXTERNAL any -> $CISCO any (msg: "Cisco TOS Example"; **tos: !"0"**;)

ใช้ในการตรวจสอบแพ็กเก็ตจากภายนอกที่ไปยังอุปกรณ์ Cisco ที่มีค่าใน TOS Field ไม่เท่ากับศูนย์

หมายเหตุ : ในอุปกรณ์ Cisco รุ่นเก่าบางรุ่น TOS Field ที่รับเข้ามาต้องเป็นศูนย์เท่านั้น

2.6 Time-To-Live Option ใช้ในการตรวจสอบค่าในฟิลด์ TTL ในส่วนหัวของแพ็กเก็ต IP โดยระบุกค่า TTL ที่ต้องการตรวจสอบในรูปแแบบ ttl: <TTL_VALUE>

ซึ่งค่า TTL นี้สามารถใช้ในการสำรวจข้อมูลเส้นทางบนระบบเครือข่ายได้ด้วยโปรแกรมประเภทสำรวจเส้นทางเช่น traceroute, tracert, netroute เป็นต้น โดยที่เครื่องต้นทางจะส่งแพ็กเก็ต UDP ที่มีค่า TTL ค่าหนึ่ง เข้าไปบนเส้นทางที่ต้องการสำรวจ ค่า TTL นี้จะลดลงทุกครั้งที่แพ็กเก็ตเดินทางผ่าน Hop หนึ่งๆ จนเมื่อค่า TTL มีค่าเป็นศูนย์ เราเตอร์จะส่งแพ็กเก็ต ICMP ตอบกลับมายังเครื่องต้นทาง และจากข้อมูลแพ็กเก็ต ICMP นี่เอง ที่เครื่องต้นทางก็จะทราบถึงหมายเลขไอพีแอดเดรสของเราเตอร์ตัวนั้นด้วย เช่นถ้าโปรแกรม traceroute ส่งแพ็กเก็ต UDP ที่มีค่า TTL เป็นห้า เมื่อแพ็กเก็ตเดินทางไปถึงยังเราเตอร์ที่ Hop ที่ห้า ค่า TTL ก็จะเท่ากับศูนย์และเราเตอร์ก็จะส่งแพ็กเก็ต ICMp ตอบกลับมา เป็นต้น ทำให้สามารถตรวจสอบได้ว่าใครกำลังพยายามใช้โปรแกรมประเภทนี้ตรวจสอบระบบเครือข่ายของเราอยู่หรือไม่

2.7 Ip-proto Option เป็นออปชันที่ใช้โดย IP Proto plug-in ในการช่วยตรวจสอบค่าโปรโตคอลในส่วนหัวของแพ็กเก็ต IP โดยใช้การอ้างอิงหมายเลขและชื่อโปรโตคอลจากไฟล์ /etc/protocols เช่น

:

ax.25 93 AX.25 # AX.25 Frames

ipip 94 IPIP # Yet Another IP encapsulation

micp 95 MICP # Mobile Internetworking Control Pro.

scc-sp 96 SCC-SP # Semaphore Communications Sec. Pro.

etherip 97 ETHERIP # Ethernet-within-IP Encapsulation

encap 98 ENCAP # Yet Another IP encapsulation

:

ตัวอย่างเช่น

alert ip any any -> any any (**ip_proto: ipip**; msg: "IP-IP tunneling detected";)

หรืออาจระบุโดยใช้ค่าหมายเลขโปรโตคอลแทนได้ เช่น

alert ip any any -> any any (**ip_proto: 94**msg: "IP-IP tunneling detected";)

3. TCP Options เป็นคีย์เวิร์ดที่ใช้ในการตรวจสอบการบุกรุกที่มีพื้นฐานอยู่บนโปรโตคอล TCP

3.1 Sequence Number Option ใช้ในการตรวจสอบหมายเลขลำดับของแพ็กเก็ตในฟิลด์ Sequence Number บนแพ็กเก็ต TCP ในรูปแบบ seq: <SEQUENCE_NUMBER>

3.2 TCP Flags Option ใช้ในการตรวจสอบค่าที่ระบุในฟิลด์ Flags ในส่วนหัวของแพ็กเก็ต TCP ในรูปแบบ flags: <TCP_VALUES>; โดยค่า TCP_VALUE ที่สามารถกำหนดได้มีดังตารางที่ 3.5

TCP Flags	คำอธิบาย
A	The option to check if the ACK flag is set.
F	The option to check if the FIN flag is set.
P	The option to check if the PSH flag is set.
R	The option to check if the RST flag is set.
S	The option to check if the SYN flag is set
U	The option to check if the URG flag is set.

0	A unique option to detect if no TCP flag has been set within the packet.
1	The 1 option determines if the reserved bit 1 is set within the packet.
2	The 2 option determines if the reserved bit 2 is set within the packet.
+	The addition sign is used to determine if a specific flag is set and followed by other TCP flags. Ex: A+ triggers on any packet with the ACK flag set in addition to other flags.
*	The asterisk is a wild card character that you can use to specify any flag that matches on any specified flags. Ex: *AS triggers on all packets that have the ACK or SYN flag set.
!	Likewise to most negation commands, this checks to see if the packet does not have the specified flag set. Ex: !S triggers on all packets that do not have the SYN flag set.

ตารางที่ 3.5 Snort TCP Flags ที่มา http://www.rfc-editor.org/rfc/rfc793.txt

3.3 TCP ACK Option ใช้ในการตรวจสอบว่าค่าในฟิลด์ Acknowledge Number ถูกเซ็ทเป็น Non-True Value หรือไม่ โดยฟิลด์นี้จะแสดงหมายเลขของแพ็กเก็ต TCP ที่จะถูกส่งต่อไป ซึ่งฟิลด์นี้จะสำคัญก็ต่อเมื่อค่าในฟิลด์ Flag ถูกเซ็ทเป็น ACK เท่านั้น

ตัวอย่างเช่น โปรแกรม NMAP สามารถใช้เทคนิคนี้ในการสแกนเครื่องเป้าหมาย โดยการส่งแพ็กเก็ต TCP ไปยังพอร์ตที่ต้องการตรวจสอบ พร้อมเซ็ทค่า Flag เป็น ACK และ ACK Number เป็นศูนย์ และเป็นเพราะแพ็กเก็ตนี้จะถูกปฏิเสธกลับมาโดยเครื่องปลายทาง ด้วยแพ็กเก็ตที่ถูกเซ็ทค่าใน Flag Field เป็น RST เมื่อโปรแกรม NMAP ได้รับแพ็กเก็ต RST นี้ ก็แสดงว่าเครื่องปลายทางมีอยู่จริง ซึ่งวิธีนี้สามารถใช้ได้กับเครื่องที่ป้องกันการสแกนด้วยการปฏิเสธการ Ping แบบ ICMP Echo Request ตัวอย่างเช่น

alert tcp any any -> 192.168.1.0/24 any (flags: A; **ack: 0**; msg: "TCP Ping Detected";)

4. ICMP Options เป็นคีย์เวิร์ดที่ใช้ในการตรวจสอบการบุกรุกที่มีพื้นฐานอยู่บนโปรโตคอล ICMP

3.1 ICMP ID Option ใช้ในการตรวจสอบค่าในฟิลด์ Identifier ภายในส่วนหัวของแพ็กเก็ต ICMP ซึ่งฟิลด์ Identifier จะปรากฏในแพ็กเก็ต ICMP ที่มีชนิดเป็น ICMP Echo Request และ ICMP Echo Reply ซึ่งถูกใช้โดยโปรแกรม Ping โดยแพ็กเก็ตทั้งสองชนิดจะถูกส่งถึงกันระหว่างเครื่องที่ทำการส่งและรับข้อมูล โดยเครื่องที่ทำการส่งจะส่งแพ็กเก็ต Echo Request

ไปยังเครื่องรับปลายทาง เมื่อแพ็กเก็ตนี้ไปถึงเครื่องปลายทางก็จะส่งแพ็กเก็ต Echo Reply ตอบกลับมา ค่าในฟิลด์นี้จึงใช้ประโยชน์ในการตรวจสอบว่าแพ็กเก็ตใดเป็นแพ็กเก็ตที่ถูกตอบกลับมา ตัวอย่างเช่น

alert icmp any any -> any any (**icmp_id: 100**; msg: "ICMP ID=100";)

3.2 ICMP Sequence Option ใช้ในการตรวจสอบค่าในฟิลด์ Sequence Number ในส่วนหัวของแพ็กเก็ต ICMP ตัวอย่างเช่น

alert icmp any any -> any any (**icmp_seq: 100**; msg: "ICMP Sequence=100";)

3.3 ICMP itype Option ใช้ในการตรวจสอบค่าในฟิลด์ Type ในส่วนหัวของแพ็กเก็ต ICMP เพื่อตรวจสอบการบุกรุกโดยการใช้ที่ระบุในฟิลด์นี้ ตัวอย่างเช่น

alert icmp any any -> any any (itype: 4; msg: "ICMP Source Quench Message received";)

3.4 ICODE Option ใช้ในการตรวจสอบค่าในฟิลด์ Code ในส่วนหัวของแพ็กเก็ต ICMP ซึ่งค่าในฟิลด์นี้ใช้อธิบายรายละเอียดการใช้งานของค่าในฟิลด์ Type อีกที เช่นค่าในฟิลด์ Type เป็น 5 แสดงว่าแพ็กเก็ตนี้เป็นประเภท ICMP Redirect ซึ่งแพ็กเก็ตประเภทนี้เกิดได้จากหลายสาเหตุ ซึ่งสาเหตุเหล่านั้นจะถูกระบุอยู่ในฟิลด์ Code นี้เอง เช่น

- ถ้าฟิลด์ Code = 0 แสดงว่าเป็นแพ็กเก็ต Network Redirect ICMP
- ถ้าฟิลด์ Code = 1 แสดงว่าเป็นแพ็กเก็ต Host Redirect ICMP
- ถ้าฟิลด์ Code = 2 แสดงว่าเป็นแพ็กเก็ต Redirect เนื่องจากประเภทของบริการและระบบเครือข่าย
- ถ้าฟิลด์ Code = 3 แสดงว่าเป็นแพ็กเก็ต Redirect เนื่องจากประเภทของบริการและโฮสต์

5. Rule Identifier Option เป็นคีย์ที่ใช้ในการระบุรายละเอียดต่างๆของกฎในโปรแกรม Snort เช่นหมายเลข ID ที่ใช้ในการอ้างอิง เอกสารที่เกี่ยวข้องและประเภทของกฎ

5.1 Snort ID Option ค่า SID เป็นค่าที่ใช้ในการจำแนกประเภทของกฎ ซึ่งถูกใช้ทั้งในส่วนของโมดูล Output และ Logging ในการอ้างอิงถึงกฎ โดยค่า SID แบ่งได้สามช่วงดังนี้

ช่วงค่า	การใช้งาน
0 - 99	Reserved for future use.
100 - 1,000,000	ใช้สำหรับ Ruleset ของโปรแกรม Snort
above 1,000,000	ใช้โดย Custom Snort Rules.

ตารางที่ 3.6 ช่วงค่า Snort ID

ตัวอย่างเช่น

alert ip any any -> any any (ipopts: lsrr; msg: "Loose source routing attemp; **sid: 1000001**;)

5.2 Rule Revision Number ใช้แสดงค่ารีวิชัน (Revision Number) ในกรณีที่มีการแก้ไขกฎข้อนี้จากรูปแบบดังเดิม ตัวอย่างเช่น

alert ip any any -> any any (ipopts: lsrr; msg: "Loose source routing attemp; **rev: 2**;)

5.3 Severity Identifier Option ใช้ระบุค่าลำดับความร้ายแรงของกฎ ซึ่งมีอยู่สามระดับ ได้แก่ 1, 2 และ 3 โดยที่ค่า 1 หมายถึงร้ายแรงที่สุด ตัวอย่างเช่น

alert ip any any -> $INTERNAL 21974 (**priority: 1**; msg: "Bad Worm Backdoor";)

5.4 Classification Identifier Option ใช้ในการระบุประเภทความร้ายแรงในการบุกรุกสำหรับกฎต่างๆในโปรแกรม Snort ซึ่งแบ่งประเภทตามลำดับความเสี่ยงได้ดังนี้

ประเภทความเสี่ยงสูง	คำอธิบาย
attempted-admin	Attempted administrator privilege gain
attempted-user	Attempted user privilege gain
shellcode-detect	Executable code was detected
successful-admin	Successful administrator privilege gain
successful-user	Successful user privilege gain
trojan-activity	A network Trojan was detected
unsuccessful-user	Unsuccessful user privilege gain
web-application-attack	Web application attack

ตารางที่ 3.7 ประเภทความเสี่ยงสูง (ค่า Priority 1)

ประเภทความเสี่ยงปานกลาง	คำอธิบาย
attempted-dos	Attempted DoS
attempted-recon	Attempted information leak
bad-unknown	Potentially bad traffic
denial-of-service	Detection of DoS attack
misc-attack	Miscellaneous attack
non-standard-protocol	Detection of a nonstandard protocol or event
rpc-portmap-decode	Decode of an RPD query

successful-dos	Denial of service
successful-recon-largescale	Large-scale information leak
successful-recon-limited	Information leak
suspicious-filename-detect	A suspicious filename was detected
suspicious-login	An attempted login using a suspicious user name was detected
system-call-detect	A system call was detected
unusual-client-port-connection	A client was using an unusual port
web-application-activity	Access to a potentially vulnerable Web Application

ตารางที่ 3.8 ประเภทความเสี่ยงปานกลาง (ค่า Priority 2)

ประเภทความเสี่ยงต่ำ	คำอธิบาย
icmp-event	Generic ICMP event
misc-activity	Miscellaneous activity
network-scan	Detection of a network scan
not-suspicious	Not suspicious traffic
protocol-command-decode	Generic protocol command decode
string-detect	A suspicious string was detected
unknown	Unknown traffic

ตารางที่ 3.9 ประเภทความเสี่ยงต่ำ (ค่า Priority 3)

ไฟล์คอนฟิกุเรชันที่ใช้กำหนดการแบ่งประเภทคือ classification.config ซึ่งมีรูปแบบในการกำหนดดังนี้ config classification: name, description, priority โดยที่

- name คือ ชื่อของประเภทการบุกรุก (Classtype ใน Snort Rule)
- description คือ คำอธิบายโดยย่อที่เกี่ยวข้องกับการบุกรุกนั้น
- priority คือ ลำดับความร้ายแรงของการบุกรุก

เช่น config classification: DOS, Denial of Service Attack, 2 เป็นต้น ตัวอย่างการใช้งานเช่น

alert udp any any -> 192.168.1.0/24 6838 (msg: "DOS"; content: "Server"; **classtype: DOS**;)

5.5 External References ใช้ในการระบุแหล่งข้อมูลเพิ่มเติมหรือที่ใช้อ้างอิงกฎข้อนี้จากอินเทอร์เน็ต ซึ่งกำหนดอยู่ในไฟล์ misc.rules ตัวอย่างเช่น

alert tcp any any -> any 12345 (reference: CVE, CAN-2002-1010; **reference: URL,** www.poc2.com; msg: "NetBus";)

6. Miscellaneous Rule Option เป็นคีย์ที่ใช้กำหนดออปชันอื่นๆที่เหลือของกฎ

6.1 Messages Option ใช้ในการระบุข้อความที่ใช้ในการแจ้งเตือนภัยหรือที่ปรากฏในล็อกไฟล์ที่บันทึกกฎข้อนี้ ตัวอย่างเช่น

alert tcp $EXTERNAL any -> $INTERNAL 79 (**msg: "FINGER"**;)

6.2 Logging Option ใช้ระบุเมื่อต้องการเก็บบันทึกข้อมูลแพ็กเก็ตลงล็อกไฟล์แบบเฉพาะเจาะจง ตัวอย่างเช่น

alert icmp any any -> any any (logto: "/var/log/snort/**logto_log"**; ttl: 100;)

จากกฎข้อนี้จะทำการบันทึกข้อมูลแพ็กเก็ต ICMP ที่มีค่า TTL = 100 ลงไฟล์ชื่อ logto_log ในไดเรกทอรี /var/log/snort/

6.3 Tag Option เป็นออปชันที่สำคัญอีกออปชันหนึ่ง ใช้เมื่อต้องการบันทึกข้อมูลแพ็กเก็ตอื่นที่เกี่ยวข้องกับแพ็กเก็ตที่ตรวจเจอจากกฎข้อนี้ เพื่อใช้ในการวิเคราะห์ข้อมูลแพ็กเก็ตทั้งหมดที่เกี่ยวข้องกับการบุกรุกนี้ โดยมีรูปแบบการใช้งานคือ tag: <TYPE>,<COUNT>,<METRIC>[,DIRECTION] ดังตาราง

อาร์กูเมนต์	คำอธิบาย
TYPE	ระบุได้เป็น session หรือ host ถ้าเป็น session จะเก็บบันทึกเฉพาะข้อมูลแพ็กเก็ตที่เกี่ยวข้องกับการเชื่อมต่อนี้ ถ้าเป็น hostจะเก็บบันทึกเฉพาะข้อมูลแพ็กเก็ตทั้งหมดที่มาจากโฮสต์นี้
COUNT	จำนวนแพ็กเก็ตหรือระยะเวลาที่จะทำการเก็บบันทึก
METRIC	หน่วยของ COUNT เป็น packets หรือ seconds
DIRECTION	ใช้ระบุทิศทางของแพ็กเก็ตเป็น src หรือ dst

ตารางที่ 3.10 อาร์กูเมนต์ที่ใช้ร่วมกับคีย์เวิร์ด tag

ตัวอย่างเช่น

alert tcp 192.168.1.0/24 23 -> any any (content: "boota"; msg: "Detected boota"; **tag: session,100, packets**;)

6.4 Dsize Option ใช้ระบุความยาวข้อมูลของแพ็กเก็ตในการตรวจสอบการบุกรุกประเภท Buffer Overflow โดยตรวจสอบจากค่าความยาวข้อมูลในแพ็กเก็ตว่ามีขนาดใหญ่

เล็กกว่าหรือเท่ากับตัวเลขที่ระบุไว้หรือไม่ รูปแบบการใช้งาน dsize: (<, > หรือไม่มี) length (< > length); ตัวอย่างเช่น

alert ip any any -> 192.168.1.0/24 any (**dsize: >6000**; msg: "Large size IP packet detected";)

6.5 RPC Option ใช้ในการตรวจสอบแพ็กเก็ตจากบริการแบบ Remote Procedure Call (RPC) โดยมีรูปแบบการใช้งานคือ rpm: <APPLICATION>,<PROCEDURE>,<VERSION>; ตัวอย่างเช่น

alert udp $EXTERNAL any -> $HOME 111 (rpc: 100023,*,*; msg: "RPC Statmon Connection";)

6.6 Real-Time Countermeasures Option เป็นออปชันที่สำคัญมาก ใช้ในการหยุดยั้งการบุกรุกที่ตรวจพบเจอแบบอัตโนมัต โดยการส่งแพ็กเก็ตตอบสนองกลับไปยังแหล่งที่มาของแพ็กเก็ตนั้น โดยมีการตอบสนองในรูปแบบต่างๆ ดังตาราง

อาร์กูเมนต์	คำอธิบาย
Rst_sed	Sends a TCP Reset packet to the sender of the packet.
Rst_rcv	Sends a TCP Reset packet to the receiver of the packet.
Rst_all	Sends a TCP Reset packet to both sender and receiver of the packet.
Icmp_net	Sends a ICMP Network Unreacheable packet to the sender.
Icmp_host	Sends a ICMP Host Unreacheable packet to the sender.
Icmp_port	Sends a ICMP Port Unreacheable packet to the sender.
Icmp_all	Sends all of the above mentioned packets to the sender.

ตารางที่ 3.11 อาร์กูเมนต์ที่ใช้ร่วมกับคีย์เวิร์ด resp

ตัวอย่างเช่น

alert tcp any any -> 192.168.1.0/24 8080 (**resp: rst_snd**;)

หมายเหตุ : การใช้ออปชันนี้ต้องคอมไพล์โปรแกรม Snort ให้รู้จักปลั๊กอิน FlexResp ก่อน โดยคอมไพล์ด้วยออปชัน --with-flexresp

6.7 React Option เป็นออปชันที่ใช้บล็อคการติดต่อกับบางโฮสต์หรือบางบริการ ตัวอย่างเช่น

alert tcp 192.168.1.0/24 any -> any 80 (msg: "Outgoing HTTP connection"; **react: block**;)

ซึ่งจะทำการบล็อคการติดต่อที่ผ่านบริการ HTTP ที่มาจากเครือข่ายภายในของเราเอง (192.168.1.0/24) โดยทำให้การติดต่อจบลงด้วยการส่งแพ็กเก็ต TCP FIN ไปยังเครื่องทั้งสองด้าน (ด้านส่งและรับ) ทุกครั้งที่มีการตรวจพบเจอด้วยกฎข้อนี้

หรืออาจใช้อาร์กูเมนต์อีกตัวคือ warn ให้ทำการแจ้งเตือนไปยังเครื่องต้นทางแทนได้ ตัวอย่างเช่น

alert tcp 192.168.1.0/24 any -> any 80 (msg: "Outgoing HTTP connection"; **react: warn, msg**;)

หมายเหตุ : การใช้ออปชันนี้ต้องคอมไพล์โปรแกรม Snort ให้รู้จักปลั๊กอิน FlexResp ก่อน โดยคอมไพล์ด้วยออปชัน --with-flexresp

3.1.4 Output Plug-ins

ทำหน้าที่สำคัญในการแสดงผลลัพธ์ของโปรแกรม Snort เก็บบันทึกข้อมูลการบุกรุก และส่งสัญญาณเตือนภัย ซึ่งส่วนปลั๊กอิน Output นี้จะถูกเรียกใช้โดยส่วนอื่นๆในโปรแกรม Snort ทั้งสิ้น ได้แก่

- ส่วน Packet Decode Engine ใช้ปลั๊กอิน Output ในการออกผลลัพธ์ของข้อมูลแพ็กเก็ตที่รับเข้ามาในรูปแบบของ TCPDump และ ASCII decode
- ส่วน Preprocessor ใช้ปลั๊กอิน Output ในการส่งสัญญาณเตือนภัย เช่นในส่วนของ PortScan2 Preprocessor
- ส่วน Detection Engine ใช้ปลั๊กอิน Output ในการส่งสัญญาณเตือนภัยและเก็บบันทึกข้อมูล

ซึ่งขึ้นอยู่กับการกำหนดในไฟล์คอนฟิกุเรชันของโปรแกรม Snort โดยสามารถกำหนดเอาท์พุตในรูปแบบต่างๆได้ดังนี้

- บันทึกลงไฟล์ /var/log/snort/alerts ในรูปแบบของเท็กซ์ไฟล์
- ส่งสัญญาณเตือนภัยแบบ SNMP
- ส่งข้อมูลต่อไปให้โปรแกรม Syslog เพื่อบันทึกลงล็อกไฟล์ของระบบ
- บันทึกลงฐานข้อมูล เช่น MySql, Postgresql, Oracle, MS-Sql และ UnixODBC ในกรณีที่ใช้ฐานข้อมูลอื่น เช่น DB2, Informix หรืออื่นๆ สามารถใช้ UnixODBC เป็นตัวกลางในการเชื่อมต่อได้
- แปลงผลลัพธ์ให้อยู่ในรูปแบบของ eXtensible Markup Language (XML)
- ส่งสัญญาณเตือนภัยไปยังอุปกรณ์ที่ทำหน้าที่รักษาความปลอดภัยเช่น เราเตอร์ หรือไฟร์วอลล์
- ส่งสัญญาณเตือนภัยแบบ Server Message Block (SMB)

3.1.5 การกำหนดไฟล์คอนฟิกุเรชัน snort.conf

ไฟล์คอนฟิกุเรชัน Snort.conf เป็นไฟล์ที่ใช้กำหนดการทำงานของโปรแกรม Snort โดยเมื่อมีการสั่งให้โปรแกรม Snort เริ่มทำงาน จะต้องทำการอ่านไฟล์คอนฟิกุเรชันนี้ด้วยทุกครั้ง ตัวอย่างคำสั่งการเรียกใช้โปรแกรม Snort เช่น

/usr/local/bin/snort -u snort -g snort -i eth0 -d -D -c /etc/snort/snort.conf

โดยไฟล์คอนฟิกุเรชัน Snort.conf ประกอบด้วยการกำหนดค่าใน 5 ส่วนหลัก ได้แก่

1) การกำหนดค่าตัวแปร

ซึ่งใช้ประกอบในกฎข้อต่างๆของโปรแกรม Snort ตัวอย่างการกำหนดค่าตัวแปรในรูปแบบต่างๆ เช่น

- var HOME_NET [10.0.0.0/8,192.168.0.0/16] เป็นการกำหนดวงเครือข่ายภายใน โดยประกอบด้วย 2 วงได้แก่ 10.0.0.0 netmask 255.0.0.0 และ 192.168.0.0 netmask 255.255.0.0

- var EXTERNAL_NET any เป็นการกำหนดค่าเน็ตเวิร์คสำหรับภายนอกเป็นทุกๆไอพี แอดเดรส

- var DNS_SERVERS [10.4.28.10, 10.4.0.2] เป็นการกำหนดไอพีแอดเดรสสำหรับเซิร์ฟเวอร์ DNS

เป็นต้น

2) การกำหนดค่า Preprocessor

ซึ่งจะถูกเรียกทำงานในส่วน Detection Engine ของโปรแกรม Snort รูปแบบในการกำหนดได้แก่

preprocessor <preprocessor_name> [: <configuration_options>]

โดยที่ preprocessor_name ใช้ระบุชื่อ preprocessor ที่ต้องการเรียกให้ทำงาน

configuration_options ใช้ระบุค่าออปชันย่อยภายใต้ preprocessor นั้น

ตัวอย่างการกำหนดค่าเช่น

preprocessor stream4: detect_scans เป็นการกำหนดให้ส่วน preprocessor stream4 ทำการตรวจสอบการสแกนพอร์ต เป็นต้น

3) การกำหนดค่า Output Module

ซึ่งควบคุมส่วนออกผลลัพธ์ของโปรแกรม Snort เช่นการเก็บบันทึกล็อคลงไฟล์ ฐานข้อมูล หรือเรียกเอาท์พุตปลั๊กอินเพื่อส่งคำร้องต่อไปยังไฟร์วอลล์ เป็นต้น

รูปแบบในการกำหนดได้แก่

output <output_module_name> [: <configuration_options>]

โดยที่ output_module_name ใช้ระบุประเภทของเอาท์พุตที่ใช้เก็บผลลัพท์ของโปรแกรม

configuration_options ใช้ระบุออปชันย่อยสำหรับการทำงานในส่วนเอาท์พุต

ตัวอย่างการกำหนดค่าเช่น

output database: log, mysql, user=snort password=xxx dbname=snort host=localhost ซึ่งเป็นการกำหนดให้ทำการเก็บเอาท์พุตลงฐานข้อมูล Mysql โดยใช้บัญชีชื่อ snort รหัสผ่าน xxx ชื่อฐานข้อมูล snort และอนุญาตการเรียกใช้งานผ่านเครื่องตัวเองเท่านั้น เป็นต้น

4) การกำหนดไฟล์กฎ

เป็นการระบุไฟล์กฎต่างๆที่เก็บรูปแบบการบุกรุกที่ใช้ในการเปรียบเทียบตรวจสอบหาการบุกรุก ซึ่งใช้คีย์เวิร์ด include ระบุอยู่ในส่วนท้ายของไฟล์คอนฟิกุเรชัน snort.conf

รูปแบบในการกำหนดได้แก่

include <rule_file_name>

โดยที่ rule_file_name ใช้ระบุตำแหน่งไฟล์กฎ

ตัวอย่างการกำหนดเช่น

include myrules.rules

ตัวอย่างไฟล์คอนฟิกุเรชัน snort.conf

```
# Variable Definitions
var HOME_NET any
var EXTERNAL_NET any
var HTTP_SERVERS $HOME_NET
var DNS_SERVERS $HOME_NET
var RULE_PATH ./

# Preprocessor
preprocessor frag2
preprocessor stream4: detect_scans
preprocessor stream4_reassemble
```

```
preprocessor  http_decode: 80 -unicode  -cginull
preprocessor  unicode: 80 -unicode  -cginull
preprocessor  bo:  -nobrute
preprocessor  telnet_decode
preprocessor  portscan: $HOME_NET  4  3  portscan.log
preprocessor  arpspoof
# Output modules
output alert_fwsam: 10.4.28.30:17277/mypassword
output alert_syslog: LOG_AUTH LOG_ALERT
output log_tcpdum: snort.log
output database: log, mysql, user=snort password=mysql_password dbname=snort host=localhost
output xml: log, file=/var/log/snortxml
# Rules and include files
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
```

include $RULE_PATH/web-client.rules

include $RULE_PATH/web-php.rules

include $RULE_PATH/sql.rules

include $RULE_PATH/x11.rules

include $RULE_PATH/icmp.rules

include $RULE_PATH/netbios.rules

include $RULE_PATH/misc.rules

include $RULE_PATH/attack-responses.rules

include $RULE_PATH/oracle.rules

include $RULE_PATH/mysql.rules

include $RULE_PATH/snmp.rules

include $RULE_PATH/smtp.rules

include $RULE_PATH/imap.rules

include $RULE_PATH/pop2.rules

include $RULE_PATH/pop3.rules

include $RULE_PATH/nntp.rules

include $RULE_PATH/other-ids.rules

include $RULE_PATH/web-attacks.rules

include $RULE_PATH/backdoor.rules

include $RULE_PATH/shellcode.rules

include $RULE_PATH/policy.rules

include $RULE_PATH/porn.rules

include $RULE_PATH/info.rules

include $RULE_PATH/icmp-info.rules

include $RULE_PATH/virus.rules

include $RULE_PATH/chat.rules

include $RULE_PATH/multimedia.rules

include $RULE_PATH/p2p.rules

include $RULE_PATH/experimental.rules

include $RULE_PATH/local.rules

3.2 ระบบรักษาความปลอดภัย

ระบบปฏิบัติการลีนุกซ์สามารถใช้งานเป็นไฟร์วอลล์ได้ตั้งแต่เคอร์เนล 1.1 ซึ่งเป็นเวอร์ชันแรก โดย Alan Cox ใช้ชื่อว่า Ipfw (จาก BSD) ต่อมาลีนุกซ์เคอร์เนลเวอร์ชัน 2.0 ได้ถูกพัฒนาและปรับปรุงได้เครื่องมือที่มีชื่อว่า Ipfwadm โดยเครื่องมือชิ้นนี้อนุญาตให้ผู้ใช้สามารถควบคุมการกลั่นกรองแพ็กเก็ตผ่านกฎที่กำหนดขึ้นเองได้ และต่อมาลีนุกซ์เคอร์เนลเวอร์ชัน 2.2 ก็ได้สร้างเครื่องมือตัวใหม่ชื่อ Ipchains ซึ่งเผยแพร่ในปี 1998 โดย Rusty Russel และทีมงาน ทั้งนี้ Ipchains นี้ถือได้ว่าเป็นพัฒนาการขั้นที่สามของไฟร์วอลล์บนระบบปฏิบัติการลีนุกซ์ จวบจนกระทั่งในปัจจุบันก็มี Netfilter หรือ Iptables ซึ่งถือว่าเป็นพัฒนาการขั้นที่สี่ของไฟร์วอลล์บนระบบปฏิบัติการลีนุกซ์

Iptables นั้นเป็นชื่อใหม่ของโค้ดที่ทำหน้าที่เป็นไฟร์วอลล์ แบบ Stateful Inspection ในลีนุกซ์เคอร์เนลเวอร์ชัน 2.4 เป็นต้นมา ซึ่งได้ถูกออกแบบและปรับปรุงใหม่จากเวอร์ชันก่อนหน้านี้ โดย Iptables นั้นสามารถทำงานย้อนหลังร่วมกับ Ipchains และ Ipfwadm ได้

รูปแบบการใช้งาน iptables เบื้องต้น

Iptables จะมีรูปแบบการใช้งานดังนี้คือ

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

โดยกฎแต่ละบรรทัดที่เขียนขึ้นจะเป็นเป็นตัวบอกเคอร์เนลว่าให้กระทำอย่างไร ในกรณีที่พบแพ็กเก็ตตรงตามที่ระบุไว้

- [table] หมายถึง ตารางที่ต้องการระบุ เช่น iptables -t nat หมายถึงให้ทำงานกับ Nat Table ในกรณีที่ไม่ได้ระบุตาราง Iptables จะถือว่าคำสั่งดังกล่าวระบุถึง Filter Table โดยอัตโนมัติ
- <command> จะเป็นตัวสั่งให้ Iptables ทำในสิ่งที่ต้องการ เช่น iptables -A INPUT ซึ่งหมายถึงให้สร้างกฎใหม่ต่อท้าย INPUT Chain ใน Filter Table
- <match> เป็นส่วนที่ใช้ตรวจสอบว่าแพ็กเก็ตมีข้อมูลตรง (match) กับที่ระบุไว้หรือไม่ เช่น มี source ip address เป็น 1.2.3.4
- <target/jump> เป็นตัวระบุว่าเมื่อเจอแพ็กเก็ตที่ตรงกับที่ระบุไว้จะกระทำ (action) ตามที่ระบุไว้ เช่นถ้าแพ็กเก็ตใดมีหมายเลขไอพีแอดเดรสต้นทางเป็นเป็น 1.2.3.4 ให้ DROP แพ็กเก็ตนั้นทิ้งไป

Table

Iptables สามารถทำงานได้กับตาราง 3 ตารางหลัก สามารถระบุตารางได้โดยใช้ออปชัน -t ตามด้วยชื่อตารางดังนี้

1) Filter Table ใช้สำหรับกรองแพ็กเก็ตมี 3 built-in chain ได้แก่ INPUT, OUTPUT, FORWARD ซึ่งจะได้อธิบายรายละเอียดต่อไป

2) Nat Table ใช้สำหรับการแปลงแอดเดรส (Network Address Translation) มี 3 Built-in Chain คือ PREROUTING, POSTROUTING, OUTPUT ซึ่งรายละเอียดจะได้อธิบายต่อไป

3) Mangle Table เป็นตารางที่ใช้เปลี่ยนแปลงหรือแก้ไขแพ็กเก็ตเช่น เปลี่ยนค่า TTL, MARK ซึ่งปกติจะใช้ในการทำเราติ้ง (Routing) ที่มีความซับซ้อนสูง มี 2 Built-in Chain คือ PREROUTING Chain (ใช้แก้ไขแพ็กเก็ตก่อนที่จะเข้าสู่ไฟร์วอลล์และก่อนเข้าสู่ Routing Decision) และ OUTPUT Chain (ใช้แก้ไขแพ็กเก็ตที่ถูกสร้างโดยไฟร์วอลล์ก่อนที่มันจะถูกส่งไปยัง Routing Decision) ทั้งนี้ไม่สามารถทำ Network Address Translation หรือ Masquerading ที่ Table นี้ได้

Command

- -A เพิ่มกฎใหม่ต่อท้าย Chain (Append Rule) เช่น

  # iptables -A INPUT -p ALL -i eth0 -j ACCEPT
- -D ลบกฎ (Delete Rule) เช่น

  # iptables -D INPUT --dport 80 -j DROP
- -I เพิ่มกฎใหม่ใน Chain (Insert Rule) เช่น

  # iptables -I OUTPUT -p ALL -s 127.0.0.1/32 -j ACCEPT
- -R แทนที่กฎเดิมด้วยกฎใหม่ (Replace Rule)
- -L แสดงกฎทั้งหมดใน Chain (ถ้าไม่ระบุ Chain จะแสดงกฎทั้งหมดใน Filter Table ทั้งสาม Built-in Chain) เช่น

  # iptables -L

  # iptables -L -t nat

  # iptables -L INPUT
- -F ลบกฎทั้งหมดใน Chain ทิ้ง เช่น

  # iptables -F INPUT

  # iptables -F mychain

- -Z ใช้ Reset Byte Counter สำหรับทุกกฎใน Chain ที่กำหนด เช่น

  # iptables -Z INPUT

- -N ใช้สร้าง chain ใหม่ เช่น

  # iptables -N mychain

- -X ลบ chain ที่ไม่มีกฎซึ่งสามารถลบ User-defined Chain ที่ไม่มีกฎได้ แต่ไม่สามารถลบ Built-in Chain ได้ เช่น

  # iptables -X emptychain

- -P เปลี่ยนนโยบายพื้นฐาน (Default Policy) ของ Chain ค่าที่ใช้ได้คือ ACCEPT, DROP ทั้งนี้ค่านี้มีความสำคัญอย่างมากเพราะหากแพ็กเก็ตถูกส่งเข้ามาใน Chain แล้วและไม่ตรงกับกฎใดๆ เลย แพ็กเก็ตนั้นก็ต้องถูกตัดสินใจโดย Policy ของ Chain นั้นๆ เช่น

  # iptables -P FORWARD DROP

  ซึ่งหากแพ็กเก็ตถูกส่งเข้ามายัง FORWARD Chain และไม่ตรงกับกฎใดๆใน FORWARD Chain นี้เลย มันก็จะถูก DROP ทันที

- -E ใช้เปลี่ยนชื่อ Chain ใหม่ เช่น

  # iptables -E myoldchain mynewchain

- การใช้คำสั่งด้านบนนั้นสามารถใช้ร่วมกับออปชันบางอย่างได้ คือ

- -V, --verbose ใช้ร่วมกับ -L, -A, -I, -D, -R เพื่อให้แสดงจำนวนไบต์ที่ตรงกับกฎออกมาด้วย (หน่วยเป็นได้ทั้ง K(x1,000),M(x1,000,000),G(x1,000,000,000)) เช่น

  # iptables -L -v

- -x, --exact ใช้ร่วมกับ -L และ -v เพื่อให้แสดงจำนวนแพ็กเก็ตและไบต์ข้อมูลที่ตรงโดยไม่ให้แสดงผลในหน่วยของ K,M,G เช่น

  # iptables -L OUTPUT -v -x

- -n, --numeric ใช้ร่วมกับ -L เพื่อสั่งให้ Iptables แสดงข้อมูลไอพีแอดเดรสและพอร์ตเป็นตัวเลขเท่านั้น เช่น

- # iptables -L OUTPUT -n

- --line-numbers ใช้ร่วมกับ -L เพื่อแสดงเลขบรรทัดของกฎซึ่งตัวเลขที่แสดงนี้จะสามารถใช้ได้กับคำสั่ง Insert Rule ที่ระบุเป็นลำดับที่ของกฎเช่น

  # iptables -L --line-numbers

- --modprobe=*command* เพื่อโหลดโมดูลที่เกี่ยวข้อง

Match

การตั้งเงื่อนไขของการ match นั้นจะต้องอาศัยความเข้าใจในเรื่อง IP, TCP, UDP, และ ICMP มาบ้างพอสมควร จึงจะสามารถตั้งเงื่อนไขที่เหมาะสมและตรงตามความต้องการได้ ซึ่งมีรายละเอียดดังนี้

- การระบุไอพีแอดเดรสต้นทาง, ปลายทาง

สามารถระบุไอพีแอดเดรสต้นทางของแพ็กเก็ตโดยใช้ -s หรือ --source หรือ --src และสำหรับไอพีแอดเดรสปลายทางก็ใช้ -d หรือ --destination หรือ --dst การระบุไอพีแอดเดรสนั้นสามารถทำได้ 4 แบบด้วยกันคือ

1) ใช้ชื่อเต็มแทน เช่น localhost หรือ www.nectec.or.th

2) ระบุไอพีแอดเดรสโดยตรง เช่น 127.0.0.1 หรือ 202.44.204.33

3) ระบุเป็นหมายเลขเน็ตเวิร์คไอพีแอดเดรส เช่น 202.44.204.0/24 ซึ่งหมายถึงไอพีแอดเดรสตั้งแต่ 202.44.204.0 - 202.44.204.255

4) หรืออาจจะใช้ 202.44.204.0/255.255.255.0 แทน 202.44.204.0/24 ได้

- การทำอินเวอร์ชัน (Inversion)

ในบางกรณีนั้นหากต้องการระบุเป็นอินเวอร์ส เช่น อนุญาตให้ทุกไอพียกเว้นไอพีที่ระบุไว้ ซึ่งการใช้คำสั่งดังกล่าวสามารถทำได้โดยใช้เครื่องหมาย ! นำหน้าอาร์กูเมนต์ที่ต้องการ (เครื่องหมาย ! หมายถึง NOT) เช่น -p ! TCP ซึ่งจะตรงกับโปรโตคอลทุกๆ ตัวที่ไม่ใช่ TCP หรือ -s ! localhost ซึ่งหมายถึงแพ็กเก็ตที่มีไอพีแอดเดรสต้นทางอื่นๆ ยกเว้น localhost (127.0.0.1)

- การระบุโปรโตคอล

สามารถระบุโปรโตคอลที่ต้องการได้ดังนี้คือ TCP, UDP, ICMP หรือสามารถใช้ตัวเลขแทนได้ (สำหรับ *NIX อ้างอิงได้จาก /etc/protocols) และยังสามารถใช้ได้ทั้งตัวอักษรเล็กหรือใหญ่ (ใช้ได้ทั้ง tcp และ TCP) เช่น -p TCP หรือ -p ! tcp

- การระบุ interface

-i หรือ --in-interface ตามด้วยชื่ออินเตอร์เฟสใช้เพื่อระบุ Incoming Interface ซึ่งหมายถึงว่าแพ็กเก็ตที่จะตรงกับกฎข้อนี้ต้องเข้ามาจากอินเตอร์เฟสที่กำหนด เช่น -i eth0 หมายความว่าทุกแพ็กเก็ตที่เข้ามาทาง eth0 จะตรงกับกฎข้อนี้ ทั้งนี้ชื่ออินเตอร์เฟสที่สามารถใช้ได้นั้น สามารถตรวจสอบได้โดยใช้คำสั่ง ifconfig และ -o หรือ --out-interface ตามด้วยชื่อของอินเตอร์เฟสใช้เพื่อระบุ Outgoing Interface ซึ่งหมายถึงว่าแพ็กเก็ตที่จะตรงกับกฎข้อนี้กำลังเดินทางผ่านอินเตอร์เฟสที่ระบุไว้ เช่น -o eth1 หรือ -o ! eth1

หมายเหตุ :

1) สำหรับ INPUT Chain นั้นไม่มี Output Interface ดังนั้นหากใช้ -o ร่วมกับ INPUT Chain ก็จะไม่มีแพ็กเก็ตใดที่ตรงกับกฎข้อนี้เลย

2) ทำนองเดียวกันกับ OUTPUT Chain ที่ไม่มี Input Interface ดังนั้นหากใช้ -i ร่วมกับ OUTPUT Chain ก็ไม่มีประโยชน์อันใด

3) FORWARD Chain มีได้ทั้ง Input และ Output Interface

4) หากระบุอินเตอร์เฟสที่ไม่มีอยู่จริง ก็จะไม่มีแพ็กเก็ตใดที่ตรงกับกฎข้อนั้นเลย

5) หากใช้เครื่องหมาย + ร่วมกับอินเตอร์เฟสเช่น ppp+ นั้นจะหมายถึงทุกๆ อินเตอร์เฟส pppเช่น ppp0, ppp1

- Fragment Packet

ในการส่งข้อมูลบนระบบเครือข่ายนั้นเป็นเรื่องปกติที่จะเกิดการแฟรกเมนต์ของแพ็กเก็ตเนื่องจากขนาดของแพ็กเก็ตมีขนาดใหญ่เกินไปที่จะส่งไปในครั้งเดียว จำเป็นต้องมีการแบ่งแพ็กเก็ตออกเป็นหลายๆส่วนย่อยเพื่อทยอยส่งไป โดยเครื่องปลายทางจะทำหน้าที่ประกอบแพ็กเก็ตย่อยเหล่านั้น รวมกันเป็นแพ็กเก็ตที่สมบูรณ์ดังเดิม

ข้อมูลที่เป็นแฟรกเมนต์แพ็กเก็ตนั้นจะมีส่วนหัว (Header) ที่สมบูรณ์แค่แพ็กเก็ตแรกเท่านั้น ส่วนแพ็กเก็ตที่ตามมาจะมีแค่ส่วนหัวบางส่วนคือไอพีแอดเดรสเท่านั้น ไม่มีข้อมูลของโปรโตคอลแนบมาด้วย ดังนั้นการตรวจสอบข้อมูลส่วนหัวของแพ็กเก็ต TCP, UDP, ICMP จึงไม่สามารถทำได้ในแพ็กเก็ตที่สองเป็นต้นมา

หากใช้ NAT บรรดาแฟรกเมนต์แพ็กเก็ตจะถูกประกอบเข้าด้วยกันจนสมบูรณ์ก่อนที่แพ็กเก็ตจะเข้าไปถึง Packet Filtering ดังนั้นจึงไม่มีความจำเป็นที่จะต้องกังวลเกี่ยวกับแฟรกเมนต์แพ็กเก็ต

ดังนั้นถ้าไม่ได้ใช้ NAT ก็ควรทำความเข้าใจไว้ว่า Iptables มีกระบวนการในการทำงานกับแฟรกเมนต์แพ็กเก็ตอย่างไร หลังจากที่แฟรกเมนต์แพ็กเก็ตแรกผ่านเข้ามาแล้ว Iptables สามารถตรวจสอบได้ว่าจะอนุญาตให้ผ่านหรือไม่ ในขณะที่แฟรกเมนต์แพ็กเก็ตที่สองและหลังจากนั้นที่ตามมานั้น จะไม่สามารถตรงกับกฎใดๆ เลย เช่น -p TCP --sport www หรือแม้แต่ -p TCP --sport ! www

อย่างไรก็ตาม สามารถเขียนกฎให้ตรวจสอบทั้งแฟรกเมนต์แพ็กเก็ตตัวที่สองและหลังจากนั้นที่ตามมาได้ด้วยการใช้ -f หรือ --fragment ทั้งนี้อาจจะเขียนในทางตรงข้ามคือไม่ต้องตรวจสอบแฟรกเมนต์แพ็กเก็ตที่สองและหลังจากนั้นโดยใช้ ! -f ก็ได้

ทั้งนี้โดยปกติแล้วมักจะปล่อยให้แฟรกเมนต์แพ็กเก็ตผ่านไป เนื่องจากถ้าสามารถ DROP ตัวแฟรกเมนต์แพ็กเก็ตตัวแรกได้แล้ว มันก็ไม่สามารถถูกประกอบที่เครื่องปลายทางได้ แต่ทั้งนี้แฟรกเมนต์แพ็กเก็ตที่ถูกปล่อยไปดังกล่าวอาจจะทำให้เครื่องที่ได้รับแฮงค์หรือแครชได้ หรืออาจจะเกิดการโจมตีแบบ Denial of Service โดยใช้แฟรกเมนต์แพ็กเก็ตได้

- TCP extension

ถ้ามีการเรียกใช้ -p tcp ตัว TCP extension ก็จะถูกโหลดมาใช้งานโดยอัตโนมัติ โดยมีออปชันให้เลือกใช้งานดังนี้

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

2) เช่น # iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP โดย ALL นั้นหมายถึงทุกๆ Flag (SYN,ACK,FIN,RST,URG,PSH) และถ้า flag SYN,ACK ถูกเซ็ทพร้อมกันก็ให้ Drop Packet นั้นทิ้งไป นอกจากนี้ยังสามารถใช้ NONE ซึ่งหมายถึงไม่มี Flag ใดถูกเซ็ทได้

3) --syn เป็นสัญลักษณ์ย่อของ --tcp-flags SYN,RST,ACK SYN

4) --source-port หรือ --sport สามารถใช้ได้ทั้งตัวเลขและตัวอักษร (อ้างอิงจากไฟล์ /etc/services) และระบุเป็นพอร์ตเดี่ยวหรือช่วงของพอร์ตก็ได้

5) เช่น --sport 21:25 หมายถึง port 21 - 25 , --sport 25: หมายถึงพอร์ตที่มากกว่าหรือเท่ากับ 25 , --sport :25 หมายถึงพอร์ตที่น้อยกว่าหรือเท่ากับ 25

6) --destination-port หรือ --dport มีรูปแบบการใช้งานเช่นเดียวกันกับ --sport

7) --tcp-option ใช้ตรวจสอบออปชันแพ็กเก็ต TCP ว่าตรงกับเลขที่ระบุไว้หรือไม่

- อธิบาย flag ของ TCP เพิ่มเติม

การเชื่อมต่อโดยใช้โปรโตคอล TCP นั้น เครื่องที่เริ่มสร้างการเชื่อมต่อจะเป็นผู้ส่ง SYN Packet มายังเครื่องปลายทาง ดังนั้นหากไม่ต้องการให้ให้เครื่องใดเป็นผู้เริ่มสร้างการติดต่อก็สามารถบล็อคไอพีดังกล่าวได้ โดยใช้ --syn เช่น -p TCP -s x.x.x.x --syn หากยังไม่เข้าใจรูปแบบการเชื่อมต่อแบบ TCP นี้แล้ว ก็เป็นการยากที่จะสร้างกฎสำหรับ Iptables ดังนั้นจึงขอแนะนำให้ไปศึกษาหลักการทำงานเบื้องต้นของทั้งโปรโตคอล TCP, UDP, ICMP มาก่อน

- UDP extension

คล้ายกันกับ TCP โดยตัว UDP extension มีออปชันให้เลือกใช้เพียงแค่ 2 อย่างเท่านั้น คือ --source-port (--sport) และ --destination-port (--dport) โดยต้องระบุ -p udp ด้วย

- ICMP extension

โดยการระบุ -p icmp ก็สามารถใช้งาน ICMP Extension ได้ โดยมีออปชันให้เลือกคือ --icmp-type เช่น --icmp-type host-unreachable (หรือใช้เลข 3 แทนได้) นอกจากนี้ยังสามารถระบุ Type/Code ได้ เช่น 3/3 ซึ่งหมายถึง Port Unreachable

ประเภท	โค้ด	คำอธิบาย
0	0	Echo Reply
3	0	Network Unreachable
3	1	Host Unreachable
3	2	Protocol Unreachable
3	3	Port Unreachable
3	4	Fragmentation needed but no frag. bit set
3	5	Source routing failed
3	6	Destination network unknown
3	7	Destination host unknown
3	8	Source host isolated (obsolete)
3	9	Destination network administratively prohibited
3	10	Destination host administratively prohibited
3	11	Network unreachable for TOS
3	12	Host unreachable for TOS
3	13	Communication administratively prohibited by filtering
3	14	Host precedence violation
3	15	Precedence cutoff in effect
4	0	Source quelch
5	0	Redirect for network
5	1	Redirect for host
5	2	Redirect for TOS and network
5	3	Redirect for TOS and host
8	0	Echo request
9	0	Router advertisement

10	0	Route sollicitation
11	0	TTL equals 0 during transit
11	1	TTL equals 0 during reassembly
12	0	IP header bad (catchall error)
12	1	Required options missing
13	0	Timestamp request (obsolete)
14	0	Timestamp reply (obsolete)
15	0	Information request (obsolete)
16	0	Information reply (obsolete)
17	0	Address mask request
18	0	Address mask reply

ตารางที่ 3.12 แสดงรายละเอียดของ ICMP message

- Match Extension

เป็น Netfilter Package ที่อยู่ในช่วงทดลองใช้ รูปแบบการใช้งานให้ใช้ -m ตามด้วยเงื่อนไขที่ต้องการ เช่น -m mac ทั้งนี้มีออปชันให้เลือกใช้งานดังต่อไปนี้

1) Mac

รูปแบบการใช้งาน: -m mac หรือ --match mac ใช้ตรวจสอบ Source MAC Address ว่าตรงกับค่าที่ระบุไว้หรือไม่ มีประโยชน์สำหรับ PREROUTING, INPUT Chain โดยมีออปชันให้ใช้งานคือ

--mac-source เช่น --mac-source 00:55:81:CC:42:FF

2) limit

รูปแบบการใช้งาน: -m limit หรือ --match limit ใช้เพื่อจำกัดจำนวนของการตรวจสอบที่อาจจะมากเกินไป เป็นประโยชน์สำหรับกฎที่วางไว้ตอนท้ายสุดของ Chain (ใช้ร่วมกันกับ DROP Policy) ซึ่งส่วนใหญ่เป็นกฎที่ใช้เก็บข้อมูลลงล็อกไฟล์ ซึ่งถ้าผู้บุกรุกส่งแพ็กเก็ตที่ไม่เข้าข่ายกฎใดๆ ใน Chain จนกระทั่งมาถึงกฎที่ทำหน้าที่เก็บล็อกนี้ ถ้าแพ็กเก็ตที่เข้ามามีจำนวนมากก็อาจจะทำให้ฮาร์ดดิสก์เต็มได้ ดังนั้นจึงต้องใช้จำกัดจำนวนในการเก็บข้อมูลลงล็อก ซึ่งมีออปชันให้ใช้งานดังนี้คือ

--limit ตามด้วยตัวเลข ซึ่งบอกถึงจำนวนครั้งสูงสุดของการตรวจสอบที่ตรงกับกฎที่ยินยอมต่อ 1 วินาที เช่น --limit 5/s ทั้งนี้ยังสามารถใช้หน่วยเวลาอื่นได้ เช่น /second /minute /hour /day เช่น -m limit --limit 3/minute

--limit-burst ตามด้วยตัวเลข แสดงถึงจำนวนมากที่สุดของแพ็กเก็ตที่ตรงกับกฎนี้ ค่าดีฟอลต์ของมันคือ 5

ตัวอย่างการใช้ --limit และ --limit-burst ร่วมกัน เช่น

iptables -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG

โดยส่วนใหญ่นิยมวางกฎนี้ไว้เป็นกฎสุดท้ายใน Chain โดยเฉพาะ Chain ที่มี Default Policy เป็น DROP เพื่อเป็นตัวเก็บหลักฐานว่ามีแพ็กเก็ตใดที่ถูกส่งมาและไม่ผ่านการตรวจสอบจากกฎและกำลังจะถูก DROP โดย Default Policy โดยกฎด้านบนนี้กำหนดจำนวนตรวจสอบสูงสุดไว้ 3 ครั้งต่อนาที ซึ่งแสดงว่าใน 1 นาทีจะมีการบันทึกล็อกได้สูงสุด 3 ครั้งเท่านั้น และมีค่า Burst เท่ากับ 3 ซึ่งอธิบายได้ว่า ถ้าสมมุติมีแพ็กเก็ตที่ตรงกับกฎนี้ 3 ครั้งภายใน 2 วินาที และถึงแม้ว่าจะมีแพ็กเก็ตที่ตรงส่งมาอีกก็จะไม่มีการบันทึกลงล็อกแต่อย่างใด และจะต้องรอไปอีก 1 นาทีจึงจะมีการเริ่มการบันทึกล็อกใหม่อีกครั้ง ซึ่งมีประโยชน์ในกรณีที่มีคนต้องการส่งแพ็กเก็ตเพื่อ Flood Log หรือทำให้ล็อกในเครื่องเต็ม ทั้งนี้นิยมใช้ร่วมกับ --log-level (อ้างอิงค่าจากเลเวลใน Syslogd เพื่อกำหนดค่าเลเวลสำหรับ Syslog) และ --log-prefix เพื่อใช้อธิบายเพิ่มเติม เช่น # iptables -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --log-level DEBUG --log-prefix "Packet died: "

นอกจากนี้ยังใช้ป้องกันการโจมตีแบบ Denial of Service เช่น SYN flood ได้ด้วย เช่น

iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT

ใช้ป้องกันการโจมตีแบบ Ping of Death

#iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT

โดยปกติมักใช้วิธี drop icmp packet ทิ้งทั้งหมด เพราะถือว่า ICMP packet เป็นข้อมูลที่มีอันตรายยิ่งและสามารถปลอมแปลงได้ง่ายหรือใช้ป้องกันการถูก scan

iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT

โดยปกติไม่นิยมใช้วิธีนี้นักเพราะมีทางเลือกที่ดีกว่าคือการตรวจสอบสอบจาก State Match ว่าเป็นการเชื่อมต่อใหม่หรือไม่ (state = new) ถ้าใช่และ SYN Bit ไม่ถูกเซ็ทตัวแพ็กเก็ตนั้นก็จะถูก DROP ทิ้งไป

3) owner

รูปแบบการใช้งาน: -m owner หรือ --match owner ใช้ตรวจสอบลักษณะของแพ็กเก็ตว่าใครเป็นผู้สร้าง ซึ่งสามารถใช้ได้กับ OUTPUT Chain เท่านั้น และใช้ได้กับบางแพ็กเก็ตที่มีเจ้าของ เช่น ICMP Packet นั้นใช้ไม่ได้เพราะไม่มีเจ้าของ มีออปชันให้ใช้งานดังนี้คือ

--uid-owner *userid*

ใช้ตรวจสอบว่าแพ็กเก็ตถูกสร้างโดย User Id ที่ระบุไว้หรือไม่ (ใช้ตัวเลขแทน User Id เท่านั้น)

--gid-owner *groupid*

ใช้ตรวจสอบว่าแพ็กเก็ตถูกสร้างโดยผู้ใช้ที่อยู่ใน Group Id ที่ระบุไว้หรือไม่ (ใช้ตัวเลขแทน Group Id เท่านั้น)

--pid-owner *processid*

ใช้ตรวจสอบว่าแพ็กเก็ตถูกสร้างขึ้นจากโพรเซสที่มี Process Id ตรงกับที่ระบุไว้หรือไม่

--sid-owner *sessionid*

ใช้ตรวจสอบว่าแพ็กเก็ตถูกสร้างโดยโพรเซสที่อยู่ใน Session Group ที่กำหนดไว้หรือไม่

4) unclean

รูปแบบการใช้งาน: -m unclean หรือ --match unclean เป็นโมดูลที่อยู่ในระหว่างการทดลองใช้งาน นอกจากนี้ยังไม่มีออปชันสำหรับใช้งาน และโปรดระมัดระวังหากจะนำไปใช้กับเครื่องที่ต้องการความปลอดภัย เนื่องจากอาจจะยังมีข้อบกพร่องของโปรแกรมอยู่

โดยแพ็กเก็ตที่เข้าข่าย unclean คือ

- แพ็กเก็ต ICMP/TCP/UDP ที่มีส่วนหัวสั้นหรือไม่สมบูรณ์
- แพ็กเก็ต TCP, UDP ที่มีไอพีแอดเดรสต้นทางหรือปลายทางเป็นศูนย์
- แพ็กเก็ต TCP ที่ใช้ Flag ผสมกันแบบผิดปกติ
- แพ็กเก็ตที่ใช้ TCP ออปชัน IP ออปชันเกินความยาวที่กำหนดไว้ หรือมีความยาวของออปชันเป็นศูนย์

- แฟรกเมนต์แพ็กเก็ตที่ไม่สมบูรณ์ ทั้งด้านความยาวและค่า Offset ที่เหลื่อมซ้อนกัน เช่น Ping of Death

Multiport ใช้ร่วมกับ --sport หรือ --dport ในกรณีที่ต้องการระบุพอร์ตจำนวนมากกว่าหนึ่ง เช่น -m multiport -p tcp --sport 25,80,53

- The State Match

รูปแบบการใช้งาน: -m state หรือ --match state เป็นโมดูลที่ใช้ประโยชน์ได้เป็นอย่างดี มีออปชันให้ใช้งานดังนี้

1) NEW

รูปแบบการใช้งาน: -m state --state new หรือ --match state --state new หมายถึงแพ็กเก็ตที่เป็นตัวสร้างคอนเนคชันใหม่

2) ESTABLISHED

รูปแบบการใช้งาน: -m state --state established หรือ --match state --state established หมายถึงแพ็กเก็ตที่เกี่ยวข้องกันกับคอนเนคชันที่สร้างไว้แล้ว เช่นแพ็กเก็ต Echo Reply หรือแพ็กเก็ตที่ส่งข้อมูลออกไปจากเว็บเซิร์ฟเวอร์ เมื่อมีบริการ Request Web เข้ามา

3) RELATED

รูปแบบการใช้งาน: -m state --state related หรือ --match state --state related เป็นแพ็กเก็ตที่เกี่ยวข้องกับคอนเนคชันที่สร้างไว้แล้ว แต่ไม่ใช่ส่วนหนึ่งส่วนใดของคอนเนคชันนั้น เช่น แพ็กเก็ตข้อมูล FTP data (พอร์ต 20) ที่เกิดขึ้นจากการใช้คำสั่งใน FTP Command (port 21)

4) INVALID

รูปแบบการใช้งาน: -m state --state invalid หรือ --match state --state invalid เป็นแพ็กเก็ตที่ไม่เกี่ยวข้องกับส่วนอื่นเลย เช่น ICMP Echo Reply ที่เกิดขึ้น โดยที่ไม่มีเครื่องได้ในระบบส่ง Echo Request ออกไปเลย (กรณีเช่นนี้เกิดขึ้นได้เนื่องจากอาจจะโดนโจมตีแบบ Smurf Attack)

การระบุ target

เมื่อมีแพ็กเก็ตที่ตรงกับกฎแล้ว ต้องกำหนดเป้าหมาย (Target) สำหรับแพ็กเก็ตไว้ด้วย โดยปกติจะใช้กัน 2 เป้าหมายคือ DROP และ ACCEPT นอกจากนี้ยังมีเป้าหมายแบบอื่นได้คือ

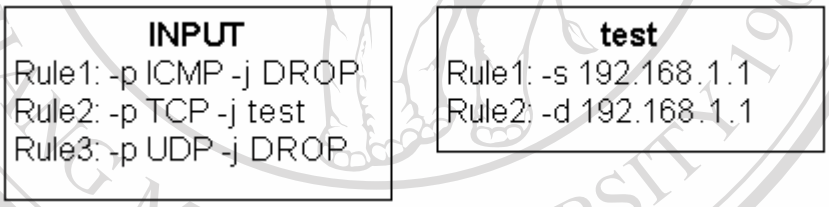
- User-defined Chain

เนื่องจาก Iptables อนุญาตให้ผู้ใช้สามารถสร้าง Chain ขึ้นมาได้ใหม่นอกเหนือจาก Built-in Chain ทั้งสามตัว (INPUT, OUTPUT, FORWARD) ทั้งนี้จะต้องใช้ตัวอักษรตัวเล็กทั้งหมดสำหรับ Chain ที่ผู้ใช้สร้างขึ้นเอง

เมื่อแพ็กเก็ตตรงกับกฎข้อที่เป็น User-defined Chain ตัวแพ็กเก็ตจะถูกนำไปตรวจสอบใหม่โดย User-defined Chain นั้นๆ และถ้าใน Chain นั้นๆ ไม่มีการตัดสินใจใดๆ ตัวแพ็กเก็ตก็สามารถย้อนกลับมายังกฎถัดไปใน Chain ที่เริ่มต้นเดินทางได้

เช่น ถ้าแพ็กเก็ต TCP เดินทางจาก 192.168.1.1 ไปยัง 1.2.3.4 ดังนั้นแพ็กเก็ตจะเข้าสู่ INPUT Chain และไม่ตรงกับกฎข้อที่ 1 แต่ตรงกับกฎข้อที่ 2 ซึ่งมีเป้าหมายเป็น Test ดังนั้นแพ็กเก็ตจะเข้าสู่ Test Chain และตรงกับกฎข้อที่ 1 แต่เนื่องจากกฎข้อที่ 1 ของ Test ไม่ได้ระบุเป้าหมาย ดังนั้นแพ็กเก็ตจึงผ่านไปยังกฎข้อที่ 2 ซึ่งไม่ตรง จากนั้นแพ็กเก็ตจึงจะเดินทางกลับไปยังกฎข้อที่ 3 ของ INPUT Chain อีกครั้ง ซึ่งก็ไม่ตรงเช่นกัน ในกรณีที่ผ่านกฎทั้งหมดแล้วแต่ไม่ตรงหรือตรงแต่ไม่มีเป้าหมายนั้นแพ็กเก็ตจะถูก DROP หรือ ACCEPT ก็ขึ้นอยู่กับ Default Policy ของ Chain นั้นๆ ซึ่งสามารถตั้งค่าได้ง่ายๆ

เช่น # iptables -P INPUT DROP หรือ # iptables -P FORWARD ACCEPT



- New Target

เป็นเป้าหมายที่สร้างเพิ่มเติมขึ้นมาคือ

1) LOG

เป็นโมดูลที่มีความสามารถในการเก็บข้อมูลลงล็อก (มี Syslog Facility เป็นเคอร์เนล) สำหรับแพ็กเก็ตที่ตรงกับกฎที่ระบุเป้าหมายเป็น LOG มีออปชันให้เลือกใช้งานดังนี้คือ

--log-level

เป็นการระบุระดับความสำคัญของล็อกซึ่งกำหนดได้ตั้งแต่ Debug, Info, Notice, Warning, Crit, Alert, Emerg รายละเอียดเกี่ยวกับ Syslog สามารถอ่านได้ที่ http://thaicert.nectec.or.th/paper/unix_linux/linux_syslog.php

--log-prefix

ตามด้วยชุดของตัวอักษรยาวไม่เกิน 29 ตัว ซึ่งชุดของตัวอักษรดังกล่าวจะปรากฏอยู่บนล็อกไฟล์

2) REJECT

คล้ายกับ DROP เพียงแต่จะส่ง ICMP Port Unreachable กลับไปยังผู้ที่ส่งแพ็กเก็ตมา (ข้อยกเว้นคือ ICMP Error Message ไม่ตอบสนองกับ ICMP Error Message ด้วยกันเอง เพราะอาจจะทำให้เกิดลูปที่ไม่รู้จบ) ทั้งนี้สามารถใช้ร่วมกับ --reject-with ตามด้วยอาร์กูเมนตที่ต้องการได้ รายละเอียดโปรดศึกษาจากคู่มือการใช้งาน Iptables ที่มาพร้อมตัวโปรแกรม (ด้วยคำสั่ง #man iptables)

- Special Built-in Target

1) RETURN

กรณีแพ็กเก็ตตรงกับกฎข้อที่มีเป้าหมายเป็น RETURN นั้นเสมือนกับเป็นคำสั่งให้ออกไปจาก Chain ปัจจุบัน เช่นหากตรงกับกฎที่อยู่ใน Built-in Chain (INPUT, FORWARD, OUTPUT) แพ็กเก็ตดังกล่าวจะถูกโยนไปยัง Default Policy ของ Chain นั้นๆ และหากแพ็กเก็ตตรงกับกฎที่เป็น User-defined Chain ตัวแพ็กเก็ตจะถูกโยนออกมา Chain ก่อนหน้านั้น

2) QUEUE

เป็น Chain พิเศษ ใช้สำหรับส่งต่อแพ็กเก็ตไปยังแอฟริเคชันที่เขียนขึ้นมารองรับโดยเฉพาะ โดยจะต้องมี Queue Handler และแอฟริเคชันเป็นส่วนประกอบที่จะทำงานร่วมกัน

3.3 ส่วนประสานการทำงานระบบตรวจสอบผู้บุกรุกเครือข่ายและระบบรักษาความปลอดภัยไฟร์วอลล์

โปรแกรม Snortsam เป็นเอาพุตท์ปลั๊กอิน (Output Plug-in) ตัวหนึ่งที่ทำงานในส่วน Detection Engine ของระบบตรวจสอบผู้บุกรุกระบบเครือข่าย ในโปรแกรม Snort ทำงานโดยการเพิ่มคีย์เวิร์ดพิเศษในกฎให้เอาพุตท์เป็นการส่งคำร้องไปบอกไฟร์วอลล์ให้ทำการบล็อคแพ็กเก็ตที่ตรวจสอบเจอโดยกฎข้อนี้ ซึ่งสามารถทำงานร่วมกับไฟร์วอลล์ได้หลายประเภท ได้แก่

- Checkpoint Firewall-1
- Cisco PIX firewalls
- Cisco Routers (using ACL's)
- Netscreen firewalls
- IP Filter (ipf), available for various Unix-like OS'es such as FreeBSD
- OpenBSD's Packet Filter (pf)
- Linux IPchains

- Linux IPtables
- WatchGuard Firebox firewalls

การทำงานของ Snortsam ประกอบด้วยสองส่วนได้แก่

3.3.1 ส่วนเซนเซอร์ของโปรแกรม Snort โดยจะต้องระบุไอพีแอดเดรสของตัวเอเจนท์ในไฟล์คอนฟิกุเรชันของโปรแกรม Snort (snort.conf) และระบุคีย์เวิร์ดในกฎให้ทำการส่งคำร้องไปยังไฟร์วอลล์ให้ทำการบล็อคแพ็กเก็ตนั้น โดยคำร้องที่ส่งไปจะถูกเข้ารหัสก่อนส่งออกด้วยแพ็กเก็ต TCP ไปยังเอเจนท์ต่างๆที่ระบุอยู่ในไฟล์คอนฟิกุเรชัน

- ในไฟล์ snort.conf

รูปแบบที่ใช้กำหนดคือ output alert_fwsam: <SnortSam Station>:<port>/<password>

โดยที่ <SnortSam Station> ใช้ระบุไอพีแอดเดรสหรือชื่อเครื่องของเอเจนท์

<port> ใช้ระบุพอร์ตที่เครื่องเอเจนท์ที่ใช้ส่งแพ็กเก็ตคำร้องไป

<password> ใช้ระบุรหัสที่ใช้ในการเข้ารหัสคำร้องที่ส่งไปยังเอเจนท์

ตัวอย่างการใช้งานเช่น

output alert_fwsam: firewall/idspassword

output alert_fwsam: fw1.domain.tld:898/mykey

output alert_fwsam: 192.168.0.1/borderfw 192.168.1.254/wanfw

- ในไฟล์กฎ

รูปแบบที่ใช้กำหนดคีย์เวิร์ดคือ fwsam: who[how],time;

โดยที่ who ใช้ระบุไอพีแอดเดรสที่ต้องการบล็อค คีย์ที่ใช้ได้แก่ src, source, dst, dest และ destination (เช่นในกฎระบุเป็น homenet -> any ถ้าต้องการบล็อค homenet ก็ต้องใช้คีย์ src เป็นต้น)

how เป็นออปชันเสริม ใช้ระบุคอนเนคชันในการบล็อคแพ็กเก็ต คีย์ที่ใช้ได้แก่ in, out, src, dest, either, both, this, conn และ connection

time ใช้ระบุระยะเวลาในการบล็อคแพ็กเก็ตนั้น คีย์เวิร์ดที่ใช้ได้แก่ days, months, weeks, years, hours, mins และ seconds

ตัวอย่างการใช้งานเช่น

fwsam: src[either], 15 min

หรือ dst[in], 2 days 4 hours

หรือ src, 1 hour

ตัวอย่างการใช้งานร่วมกับกฎในโปรแกรม Snort เช่น

alert tcp any any -> $HTTP_SERVERS 80 (msg: "WEB-MISC http directory traversal"; flags: A+; content: "..\\"; referecnce: arachnids,298; fwsam: 15 minutes;)

ซึ่งกฎข้อนี้ใช้ตรวจสอบแพ็กเก็ตจากทุกที่ที่วิ่งเข้ามายังเครื่องเว็บเซิร์ฟเวอร์ทางพอร์ต 80 ฉะนั้น src ในที่นี้หมายถึงผู้บุกรุก ส่วน dst หมายถึงเครื่องเว็บเซิร์ฟเวอร์ของเรา การระบุคีย์เวิร์ด fwsam ดังข้างต้นเป็นการสั่งให้บล็อคแพ็กเก็ตที่มาจากต้นทาง (src) เป็นเวลา 15 นาที

ขั้นตอนการทำงาน

1. กำหนดไอพีแอดเดรสตัวเอเจนท์ และคอนฟิกกฎข้อที่ต้องการส่งคำร้องต่อไปให้ตัวเอเจนท์

2. เมื่อกฎทำการแจ้งบล็อค ตัวเซนเซอร์จะทำการเข้ารหัสแพ็กเก็ตคำร้องเพื่อส่งไปยังตัวเอเจนท์

3.3.2 ส่วนเอเจนท์ที่ไฟร์วอลล์ จะระบุไอพีแอดเดรสตัวเซนเซอร์ของโปรแกรม Snort ในไฟล์คอนฟิกุเรชันของโปรแกรม Snortsam (snortsam.conf) โดยทำงานในโหมดแบ็กกราวน์หรือเดมอนบนเครื่องที่ทำหน้าที่ไฟร์วอลล์ให้แก่ระบบเครือข่าย เมื่อมีแพ็กเก็ตคำร้องเข้ามาจะทำการตรวจสอบก่อน ถ้าเป็นแพ็กเก็ตที่ส่งมาจากตัวเซนเซอร์ที่ระบุไว้ จึงส่งคำร้องนั้นต่อไปให้ยังไฟร์วอลล์เพื่อบล็อคแพ็กเก็ตตามที่รายงานเข้ามา

รูปแบบที่ใช้กำหนด ได้แก่

port <port_number>

accept <censor_ip>,<password>

iptables <adapter> <logoption>

email <smtpserver> <recipient> <sender>

โดย port ใช้ระบุหมายเลขพอร์ตที่คอยรับแพ็กเก็ตคำร้อง

accept ใช้ระบุไอพีแอดเดรสของตัวเซนเซอร์ Snort และคีย์รหัสที่อนุญาตให้รับเข้ามา

iptables ใช้ระบุประเภทของไฟร์วอลล์ อินเตอร์เฟสที่จะทำการบล็อค และล็อกไฟล์ที่ใช้บันทึก

email ใช้ระบุการส่งอีเมล์เพื่อแจ้งการบล็อคและการปลดบล็อคของตัวไฟร์วอลล์

ตัวอย่างการใช้งานเช่น

port 17277

accept 192.168.101.98, nwadmin

iptables eth1 syslog.info

email 127.0.0.1 admin@mydomain.com snortsam@mydomain.com

โดยรูปแบบคำสั่งที่ไฟร์วอลล์ใช้ในการบล็อคแพ็กเก็ตคือ

iptables -I FORWARD -i eth1 -s {ip_addr_to_be_blocked} -j DROP

iptables -I INPUT -i eth1 -s {ip_addr_to_be_blocked} -j DROP

ส่วนคำสั่งที่ใช้ในการปลดบล็อคคือ

iptables -D FORWARD -i eth1 -s {ip_addr_to_be_unblocked} -j DROP

iptables -D INPUT -i eth1 -s {ip_addr_to_be_unblocked} -j DROP

หมายเหตุ : - รูปแบบคำสั่งในการบล็อคและปลดบล็อคสามารถแก้ไขได้ที่ไฟล์ ssp_iptables.c

- อินเตอร์เฟส eth0 ปกติหมายถึง เน็ตเวิร์คภายใน (internal net) ส่วนอินเตอร์เฟส eth1 ปกติหมายถึง เน็ตเวิร์คภายนอก (external net)

ขั้นตอนการทำงาน

1. กำหนดไอพีแอดเดรสตัวเซ็นเซอร์ที่อนุญาตให้ส่งคำร้องเข้ามา
2. เมื่อมีคำร้องเข้ามา จะทำการตรวจสอบแหล่งที่มาก่อนว่าได้รับอนุญาตหรือไม่
3. ทำการถอดรหัสคำร้อง
4. ตรวจสอบรหัสผ่านหรือคีย์ระหว่างตัวเซ็นเซอร์และเอเจนท์ว่าตรงกันหรือไม่
5. ตรวจสอบไอพีแอดเดรสที่ต้องทำการบล็อคที่ส่งมากับคำร้องว่ายังถูกบล็อคอยู่หรือไม่ (ถ้าถูกบล็อคอยู่ให้ข้ามคำร้องนั้นไป)
6. ตรวจสอบไอพีแอดเดรสที่ต้องทำการบล็อคที่ส่งมากับคำร้อง กับ White-list ที่กำหนดในไฟล์ snortsam.conf (รายการไอพีแอดเดรสที่จะไม่ถูกบล็อคอย่างเด็ดขาดเช่น ไอพีของ Internet root DNS เป็นต้น)

ตัวอย่างเช่น dontblock a.root-servers.net

dontblock 192.168.10.0/24

7. ตรวจสอบระยะเวลาที่ถูกร้องขอให้ทำการบล็อค กับค่าที่ถูกกำหนดไว้ในออปชัน override ในไฟล์ snortsam.conf (ปกติกฎจของ Snort แต่ละข้อจะสามารถกำหนดช่วงเวลาในการบล็อคได้เอง แต่มีบางกรณีที่เราต้องการยกเว้นไม่ให้ถูกบล็อคเป็นเวลานาน เช่น Proxy Server

ตัวอย่างเช่น override proxy.science.cmu.ac.th, 5 min หรือ override 192.168.1.0/24, 10 sec

8. ส่งคำร้องต่อไปยังไฟร์วอลล์ เพื่อให้ทำการบล็อค (ด้วยรูปแบบคำสั่ง iptables ดังที่กล่าวมาในข้างต้น)

3.4 ส่วนแสดงผลและสืบค้นข้อมูล

ส่วนแสดงผลและสืบค้นข้อมูลที่ได้จากส่วนตรวจสอบผู้บุกรุกระบบเครือข่ายโดยโปรแกรม Snort ในที่นี้ใช้โปรแกรม ACID ซึ่งประกอบด้วยสคริปต์และไฟล์คอนฟิกุเรชันที่เขียนด้วยภาษา PHP จำนวนมาก ทำหน้าที่ในการเก็บและวิเคราะห์ข้อมูลลงฐานข้อมูลไปพร้อมๆกัน และแสดงผลในรูปแบบของเว็บอินเตอร์เฟส โดยผู้ใช้สามารถใช้เว็บเบราว์เซอร์ในการติดต่อกับ ACID ได้

นอกจากโปรแกรม ACID แล้วเครื่องที่ทำหน้าที่แสดงผลและสืบค้นข้อมูลนี้ต้องประกอบด้วยโปรแกรมเว็บเซิร์ฟเวอร์ ดาต้าเบสเซิร์ฟเวอร์ โปรแกรม PHP และเครื่องมือที่เกี่ยวข้องอื่นๆ (ดูภาคผนวก ก)

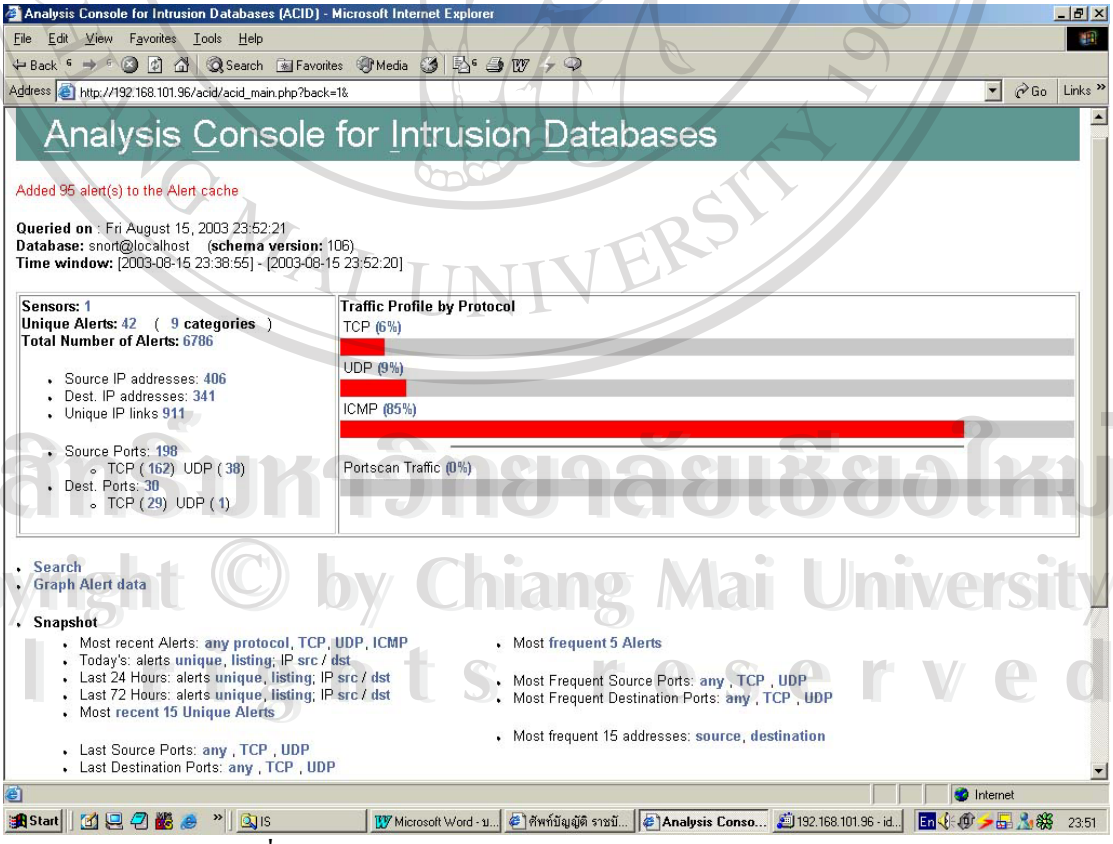
ส่วนประกอบหลักต่างๆ ของโปรแกรม ACID ได้แก่

3.4.1 หน้าหลักของโปรแกรม ACID (ACID Main Page)

เป็นส่วนการแสดงผลหลักของโปรแกรม ซึ่งมีลิงค์แยกเป็นเซกชันย่อยสำหรับแสดงผลในรูปแบบต่างๆกัน ดังนี้

- แสดงจำนวนเซ็นเซอร์ (Sensor) ทั้งหมดที่ใช้ตรวจสอบและเก็บบันทึกข้อมูล
- แสดงจำนวนการแจ้งเตือนการบุกรุกที่เกิดขึ้นจำแนกตามรูปแบบการบุกรุกแต่ละประเภท
- แสดงจำนวนการเตือนภัยทั้งหมด
- แสดงจำนวนไอพีแอดเดรสต้นทางและปลายทางทั้งหมด
- แสดงจำนวนไอพีแอดเดรสทั้งหมด
- แสดงจำนวนหมายเลขพอร์ต (TCP และ UDP) ทั้งต้นทางและปลายทาง
- แสดงจำนวนการแจ้งเตือนภัยทั้งหมดแยกตามโปรโตคอล (TCP, UDP, ICMP และจำนวนการสแกนพอร์ต)
- ส่วนการสืบค้นในรูปแบบต่างๆ
- ส่วนการแสดงผลในรูปแบบกราฟิก (Graph Alert Data)
- ส่วนแสดงผลอย่างเฉพาะเจาะจงแบบต่างๆ
- การแจ้งเตือนภัยที่เกิดขึ้น 15 ครั้งล่าสุดแยกตามประเภทโปรโตคอล TCP, UDP, ICMP หรือทั้งหมด
- การแจ้งเตือนภัยที่เกิดขึ้นภายในวันนี้
- การแจ้งเตือนภัยที่เกิดขึ้นภายใน 24 ชั่วโมงล่าสุด
- การแจ้งเตือนภัยที่เกิดขึ้นภายใน 72 ชั่วโมงล่าสุด

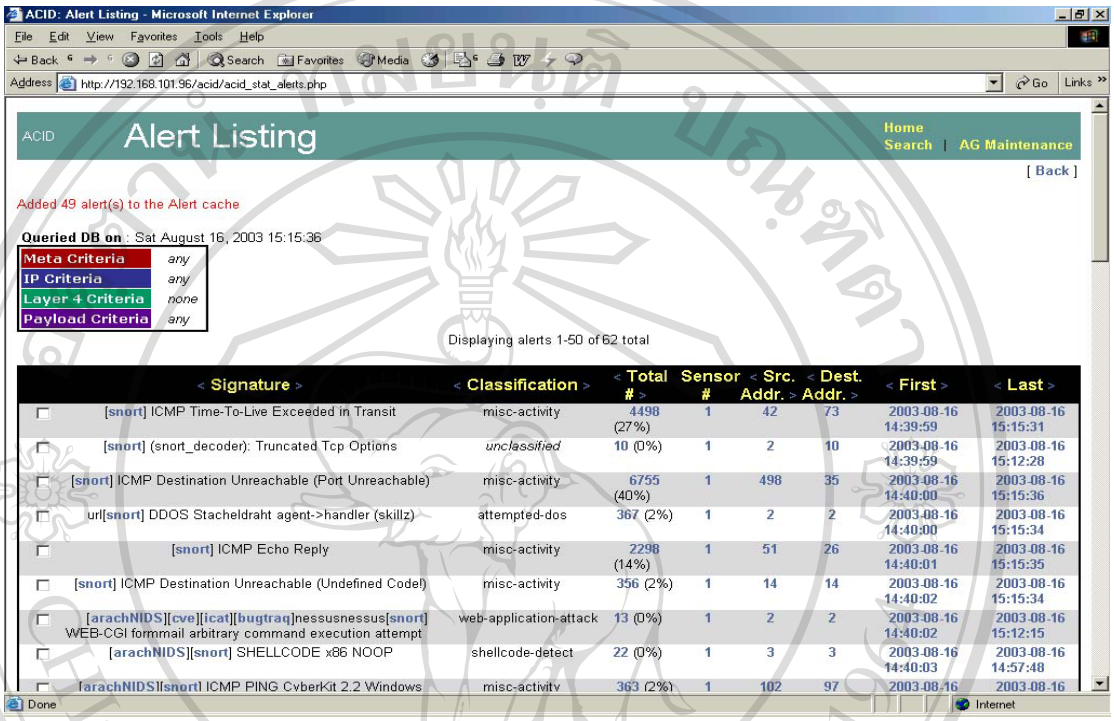
- การแจ้งเตือนภัยที่เกิดขึ้นล่าสุด 15 รูปแบบ
- การแจ้งเตือนภัยที่เกิดขึ้น 15 ครั้งล่าสุดแยกตามหมายเลขพอร์ตต้นทางของแต่ละโปรโตคอล (TCP,UDP หรือทั้งหมด)
- การแจ้งเตือนภัยที่เกิดขึ้น 15 ครั้งล่าสุดแยกตามหมายเลขพอร์ตปลายทางของแต่ละโปรโตคอล (TCP,UDP หรือทั้งหมด)
- การแจ้งเตือนภัยที่เกิดขึ้นบ่อยที่สุด 5 รูปแบบ
- การแจ้งเตือนภัยที่เกิดขึ้นบ่อยที่สุดแยกตามหมายเลขพอร์ตต้นทางของแต่ละโปรโตคอล (TCP,UDP หรือทั้งหมด)
- การแจ้งเตือนภัยที่เกิดขึ้นบ่อยที่สุดแยกตามหมายเลขพอร์ตปลายทางของแต่ละโปรโตคอล (TCP,UDP หรือทั้งหมด)
- ส่วนการแสดงผลแบบระบุช่วงเวลา แสดงผลแบบกราฟิก
- ส่วนการจัดการการจัดกลุ่มการแจ้งเตือนภัย
- ส่วนแสดงสถานะการทำงานต่างๆของระบบเช่น PHP ฐานข้อมูล และ Cache เป็นต้น



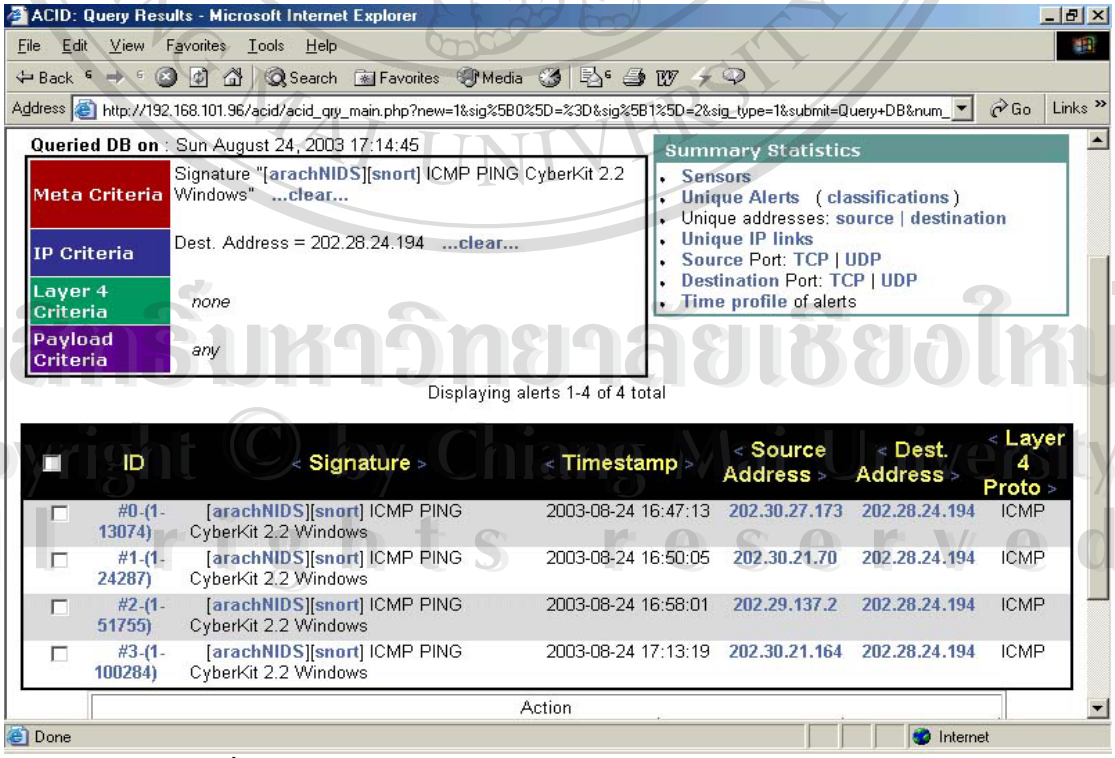
รูปที่ 3.5 หน้าหลักของโปรแกรม ACID (ACID Main Page)

3.4.2 ส่วนแสดงผลข้อมูล

จากหน้าหลัก (Main Page) สามารถคลิ๊ก (Click) เข้าไปยังเซกชันย่อยต่างๆ เพื่อดูรายละเอียดข้อมูลการบุกรุกได้



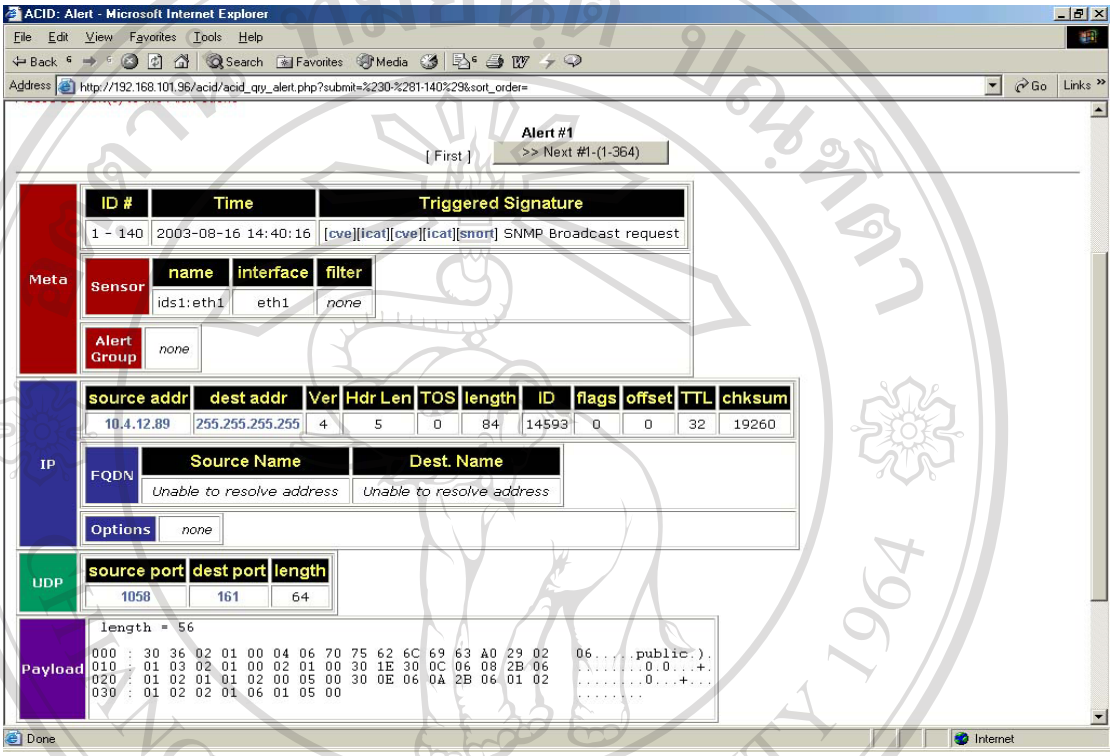
รูปที่ 3.6 ตัวอย่างการแสดงรายละเอียดข้อมูลการบุกรุกแยกประเภทตามรูปแบบการบุกรุก



รูปที่ 3.7 ตัวอย่างการแสดงรายละเอียดข้อมูลการบุกรุกแต่ละรูปแบบ

3.4.3 ส่วนแสดงรายละเอียดภายในแพ็กเก็ต

แสดงข้อมูลรายละเอียดต่างๆภายในแพ็กเก็ต เช่น เงื่อนไขในการแสดงผล/ค้นหา หมายเลขลำดับ (ID) วันเวลาที่เกิดการบุกรุก คำอธิบายการบุกรุก ข้อมูลตัวเซ็นเซอร์ที่ตรวจจับการบุกรุกนี้ ข้อมูลในส่วนหัวและนส่วนเนื้อหาของแพ็กเก็ต เป็นต้น



รูปที่ 3.8 ตัวอย่างการแสดงรายละเอียดข้อมูลการบุกรุกภายในแพ็กเก็ต

3.4.4 ส่วนสืบค้นข้อมูล

ส่วนสืบค้นข้อมูลของโปรแกรม ACID ใช้สำหรับค้นหาข้อมูล โดยมีพารามิเตอร์ที่ใช้ในการค้นหาดังนี้

ส่วน Meta Criteria

- Sensor สำหรับระบบรวมศูนย์ข้อมูลที่ใช้เซ็นเซอร์ในการตรวจจับหลายตัว
- Alert Group ระบุกรุ๊ปเตือนภัยที่ต้องการค้นหา
- Signature ระบุรูปแบบการบุกรุกที่ต้องการค้นหา
- Classification ระบุประเภทการบุกรุกที่ต้องการค้นหา

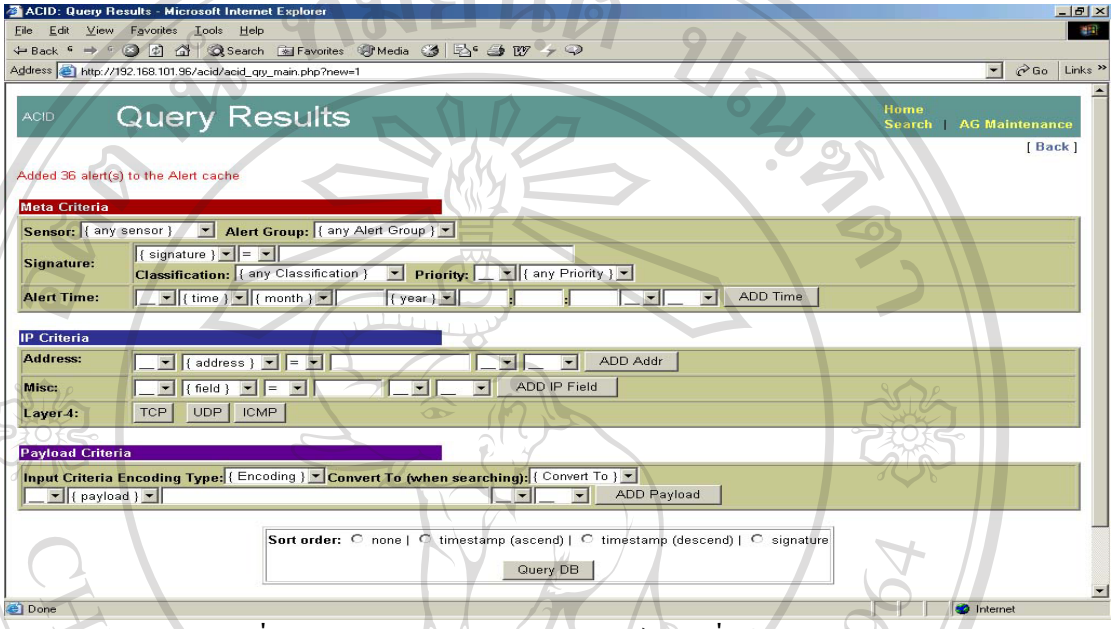
ส่วน IP Criteria

- Alert Time ระบุช่วงเวลาที่เกิดการบุกรุกที่ต้องการค้นหา
- Address ระบุค่าไอพีแอดเดรสที่ต้องการค้นหา
- Misc ระบุค่ารายละเอียดภายในส่วนหัวของแพ็กเก็ตที่ต้องการค้นหา

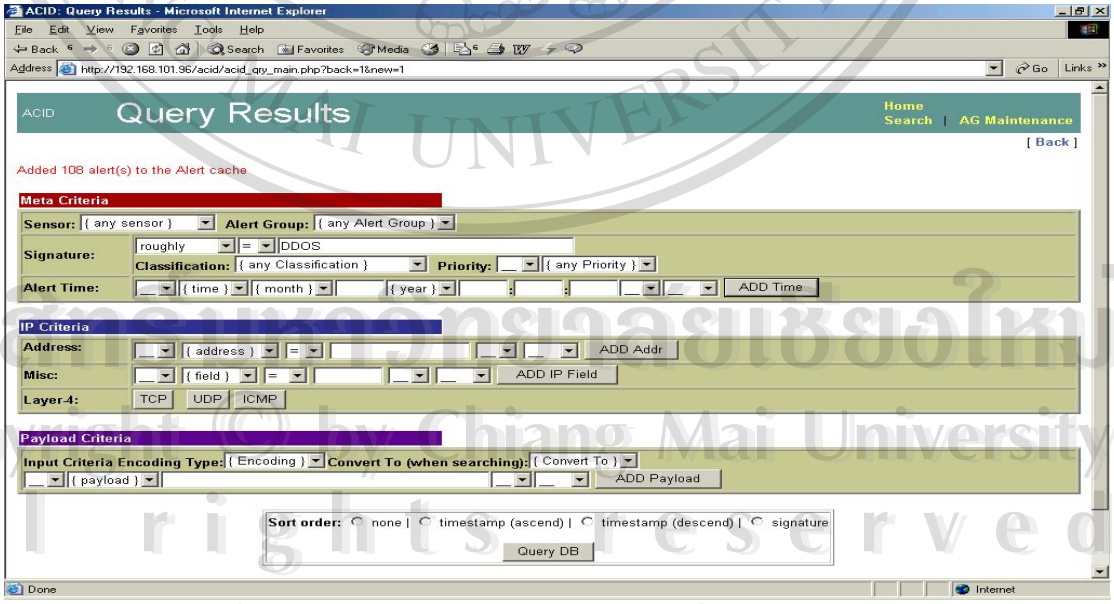
- Layer-4 ระบุชนิดโปรโตคอลของแพ็กเก็ต (TCP, UDP หรือ ICMP)

ส่วน Payload Criteria

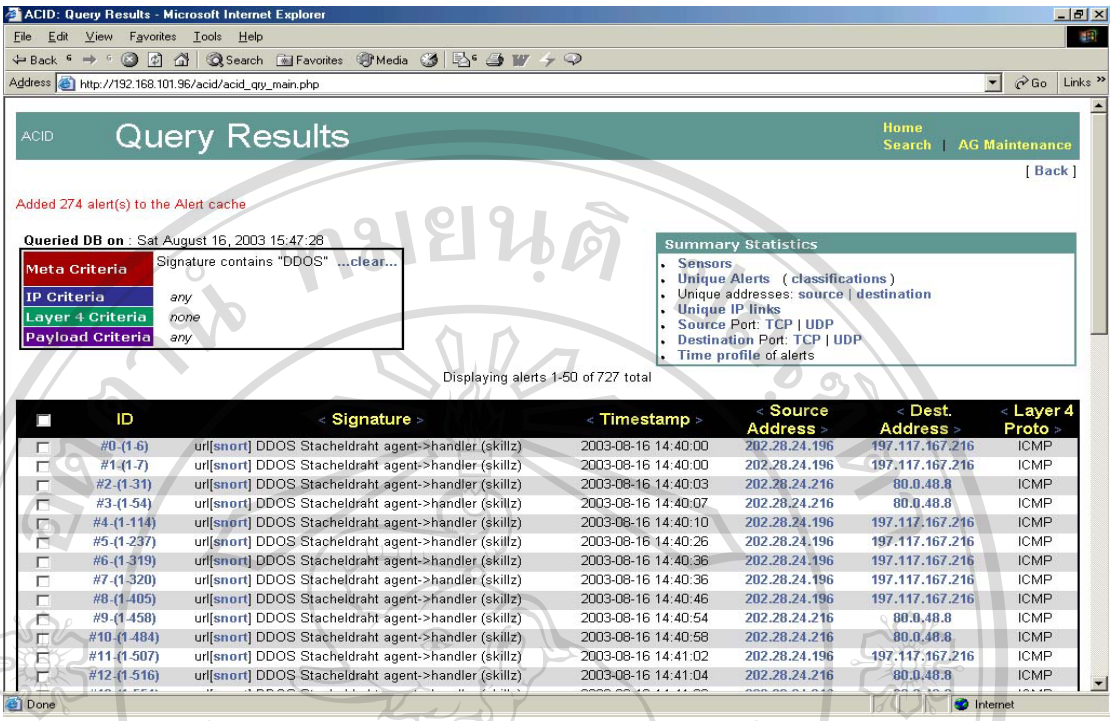
- Input Criteria Encoding Type ระบุชนิดของข้อมูลที่ต้องการค้นหาภายในส่วนเนื้อหาของแพ็กเก็ต



รูปที่ 3.9 แสดงรายละเอียดพารามิเตอร์ต่างๆที่ใช้ในการค้นหา



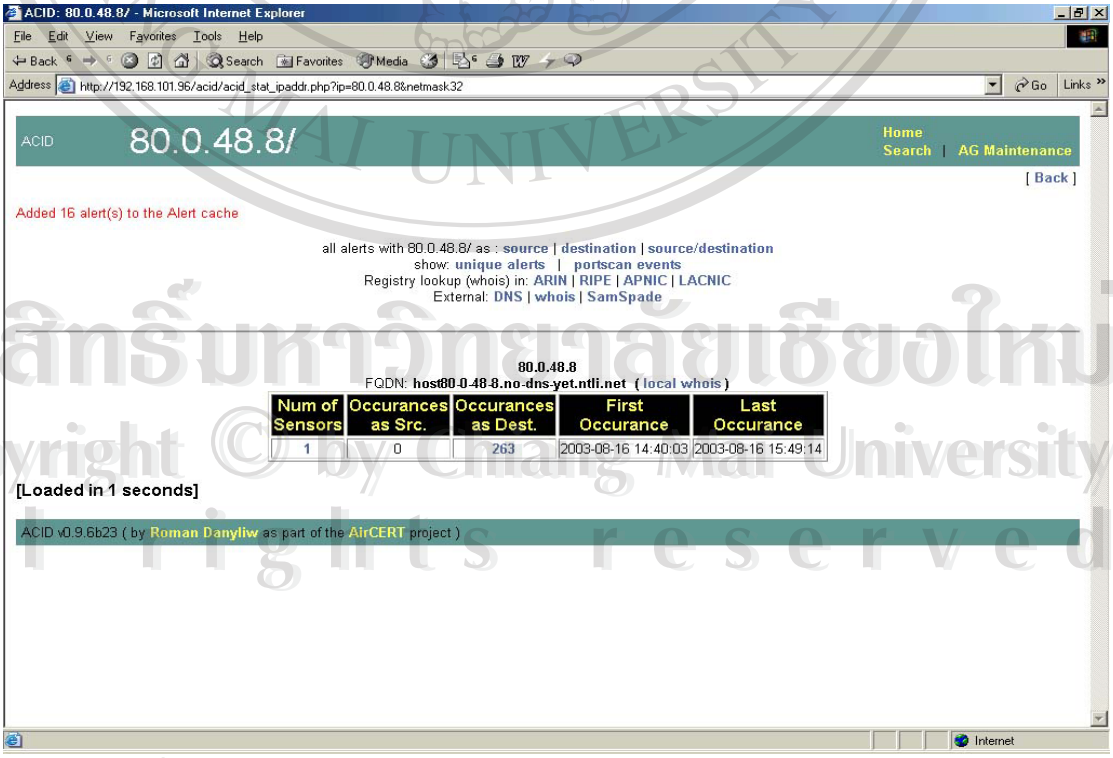
รูปที่ 3.10 ตัวอย่างแสดงการค้นหาการบุกรุกที่เป็นรูปแบบ DDOS



รูปที่ 3.11 ตัวอย่างแสดงผลลัพธ์การค้นหาการบุกรุกที่มีรูปแบบ DDOS

3.4.5 ส่วนบริการสืบค้น Whois

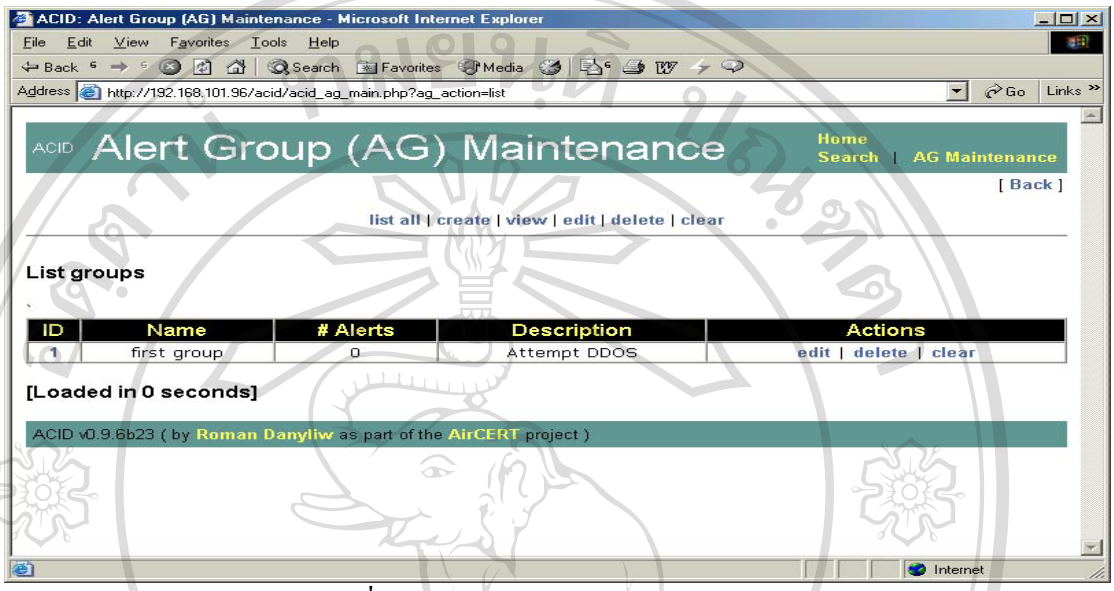
เป็นส่วนที่ให้บริการสืบค้นข้อมูลไอพีแอดเดรสผ่านบริการ Whois ต่างๆบนอินเทอร์เน็ต เช่น American Registry for Internet Numbers (ARIN) ที่ http://www.arin.net เป็นต้น



รูปที่ 3.12 ตัวอย่างแสดงการใช้บริการ Local Whois ค้นหาข้อมูลไอพี 80.0.48.8

3.4.6 ส่วนจัดการกลุ่มเตือนภัย

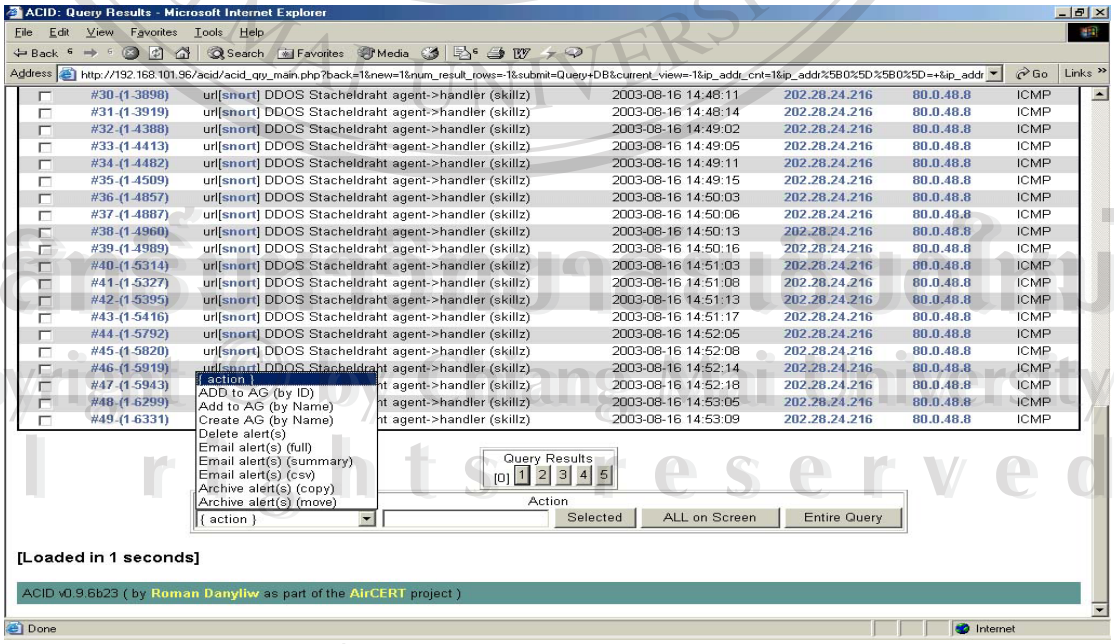
กลุ่มเตือนภัย หรือ Alert Group (AG) ในโปรแกรม ACID ใช้สำหรับจัดกลุ่มการเตือนภัยต่างๆ แบบเฉพาะเจาะจง



รูปที่ 3.13 ตัวอย่างการใช้งาน Alert Group

3.4.7 ส่วนการจัดการข้อมูล

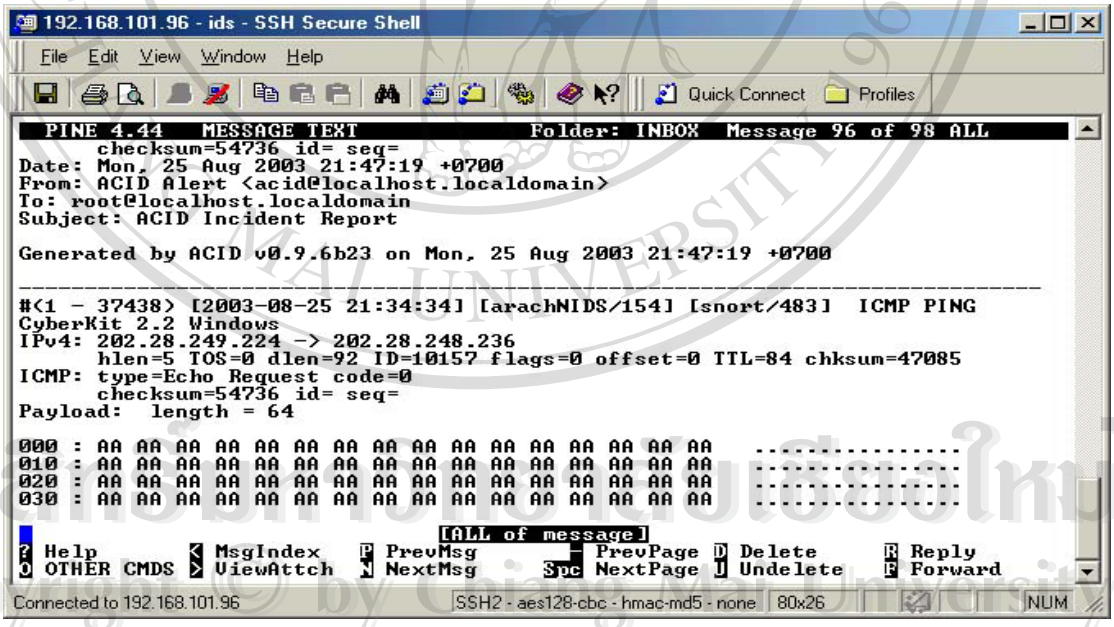
ได้แก่ส่วนล่างสุดในแต่ละหน้าที่แสดงรายละเอียดการบุกรุกแบบแยกประเภทตามรูปแบบการบุกรุกและที่แสดงแบบเฉพาะรูปแบบการบุกรุก



รูปที่ 3.14 ตัวอย่างการใช้งานส่วนจัดการข้อมูล

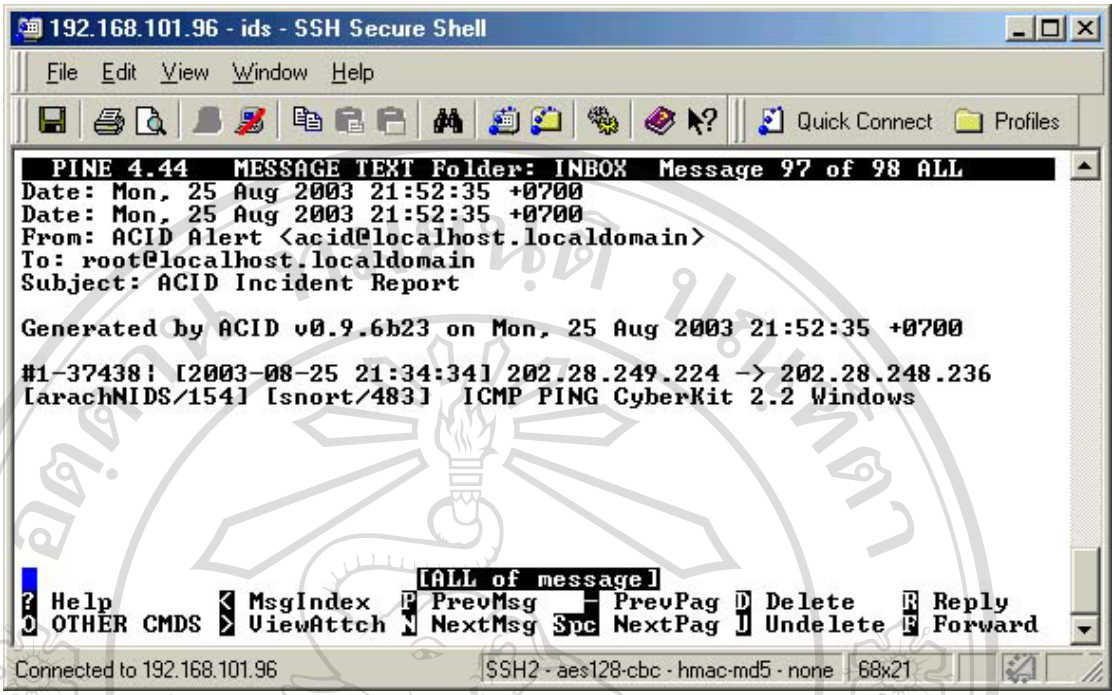
ประกอบด้วยปุ่มด้านซ้ายสุดใช้ระบุแอ็คชัน (Action) ต่างๆที่ใช้จัดการกับข้อมูลการบุกรุก ได้แก่

- Add to AG (by ID) เป็นการเพิ่มข้อมูลการบุกรุกนั้นเข้ากลุ่มเตือนภัย AG โดยระบุหมายเลข ID ของกลุ่ม AG
- Add to AG (by Name) เป็นการเพิ่มข้อมูลการบุกรุกนั้นเข้ากลุ่มเตือนภัย AG โดยระบุชื่อกลุ่ม AG
- Create AG (by Name) เป็นการเพิ่มกลุ่มเตือนภัย AG โดยระบุชื่อกลุ่ม AG
- Delete Alert(s) เป็นการลบเหตุการณ์เตือนภัย
- Email Alert(s) (Full) เป็นการส่งอีเมล์โดยระบุรายละเอียดข้อมูลการบุกรุกแบบเต็มรูปแบบ
- Email Alert(s) (Summary) เป็นการส่งอีเมล์โดยระบุรายละเอียดข้อมูลการบุกรุกแบบสรุป
- Email Alert(s) (CVE) เป็นการส่งอีเมล์โดยระบุรายละเอียดข้อมูลการบุกรุกในรูปแบบของ CVE[8]

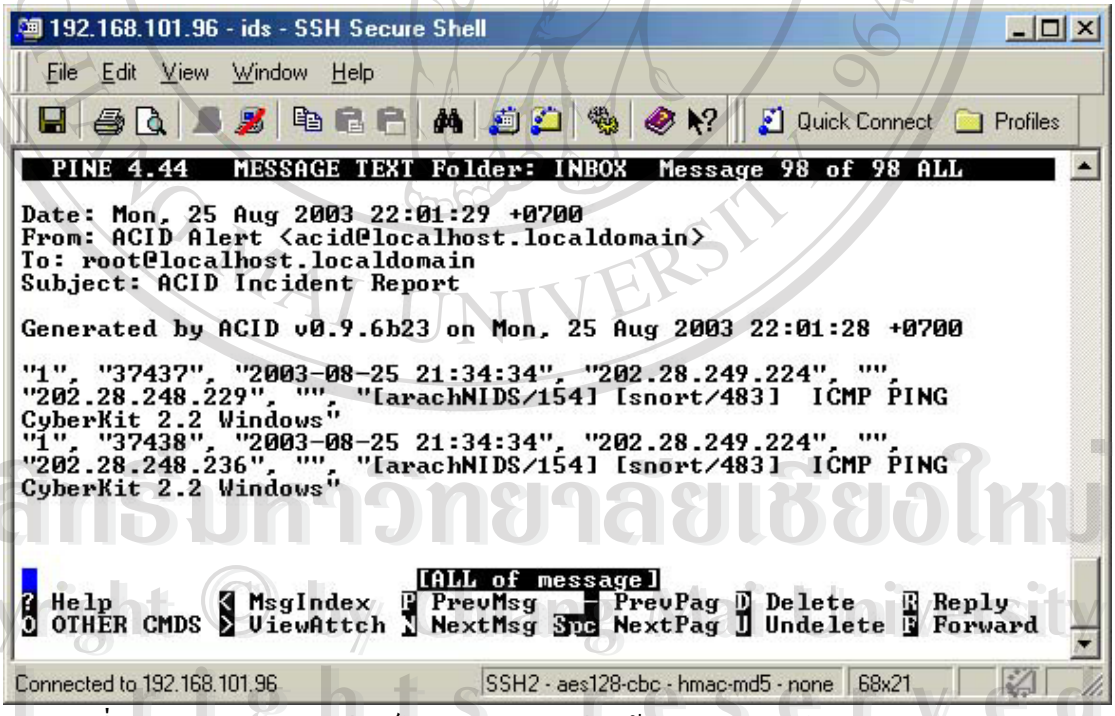


รูปที่ 3.15 ตัวอย่างการส่งอีเมล์โดยระบุรายละเอียดข้อมูลการบุกรุกแบบเต็มรูปแบบ

[8] Common Vulnerabilities and Exposures (CVE®) is: A list of standardized names for vulnerabilities and other information security exposures — CVE aims to standardize the names for all publicly known vulnerabilities and security exposures.



รูปที่ 3.16 ตัวอย่างการส่งอีเมล์โดยระบุรายละเอียดข้อมูลการบุกรุกแบบสรุป



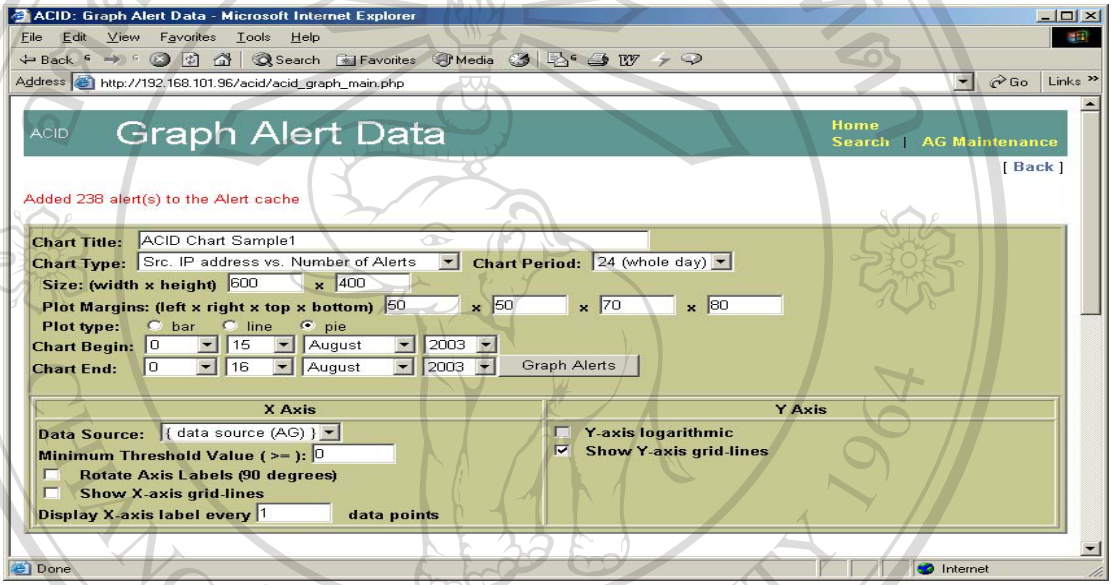
รูปที่ 3.17 ตัวอย่างการส่งอีเมล์โดยระบุรายละเอียดข้อมูลการบุกรุกในรูปแบบของ CVE

- Archive Alert(s) (Copy) เป็นการสำเนาข้อมูลการบุกรุกสู่ฐานข้อมูลอื่น
- Archive Alert(s) (Move) เป็นการย้ายข้อมูลการบุกรุกสู่ฐานข้อมูลอื่น

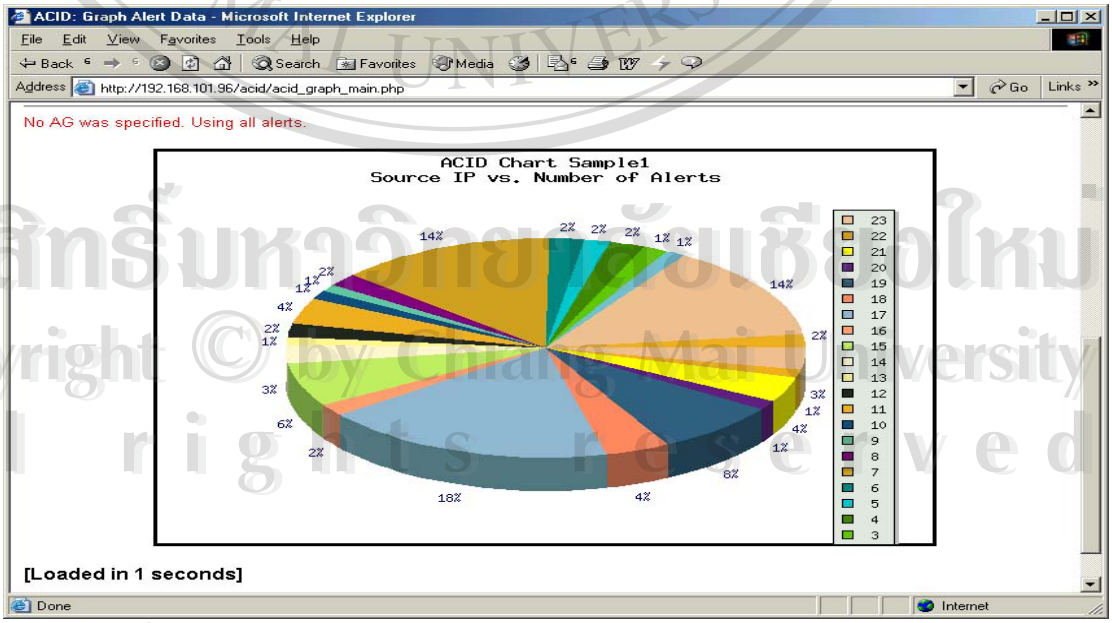
สำหรับสามปุ่มด้านขวามือ ได้แก่ปุ่ม Selected ใช้สำหรับเลือกเฉพาะเหตุการณ์ที่มีการทำเครื่องหมายถูกหน้าเหตุการณ์เท่านั้น ปุ่ม ALL on Screen ใช้สำหรับเลือกเหตุการณ์ทั้งหมดที่ปรากฏบนหน้าจอ (ปกติจะแสดงผล 50 เหตุการณ์ต่อหนึ่งหน้าจอ) และปุ่ม Entire Query ใช้สำหรับเลือกเหตุการณ์ทั้งหมด

3.4.8 ส่วนแสดงผลในรูปแบบกราฟิก (Graph Alert Data)

ใช้สำหรับแสดงผลในรูปแบบกราฟิก โดยการกำหนดช่วงเวลา กลุ่มเตือนภัย AG หมายเลขพอร์ตต้นทาง-ปลายทาง และหมายเลขไอพีแอดเดรส



รูปที่ 3.18 ตัวอย่างการใช้งานส่วนแสดงผลในรูปแบบกราฟิก



รูปที่ 3.19 ตัวอย่างการแสดงผลลัพธ์การใช้งานส่วนแสดงผลในรูปแบบกราฟิก