บทที่ 3
การศึกษาและออกแบบระบบตรวจสอบการบุกรุกระบบเครือข่าย

3.1 ขอบเขตการศึกษา

ขอบเขตของการศึกษาค้นคว้าอิสระนี้ คือการออกแบบและสร้างระบบตรวจสอบผู้บุกรุกเครือข่ายคอมพิวเตอร์ภายในมหาวิทยาลัยราชภัฏเชียงราย โดยใช้ซอฟต์แวร์ที่ไม่ได้มีการเรียกเก็บค่าลิขสิทธิ์ โดยที่ระบบที่ทำการพัฒนาขึ้นมานั้นมีความสามารถขั้นต่ำดังต่อไปนี้

1. การจัดเก็บรูปแบบและตำแหน่งการบุกรุก หมายถึงทำการบันทึกเหตุการณ์ที่ได้มีการละเมินกฎการรักษาความปลอดภัยที่มีการกำหนดไว้แล้วเพื่อให้ผู้ดูแลระบบทำการตรวจสอบติดตาม

2. การแสดงรายงานเชิงสถิติ หมายถึงระบบสามารถรายงานเหตุการณ์ที่เกิดขึ้นเป็นข้อมูลเป็นรายงานเชิงสถิติ (วัน / เดือน / ปี)

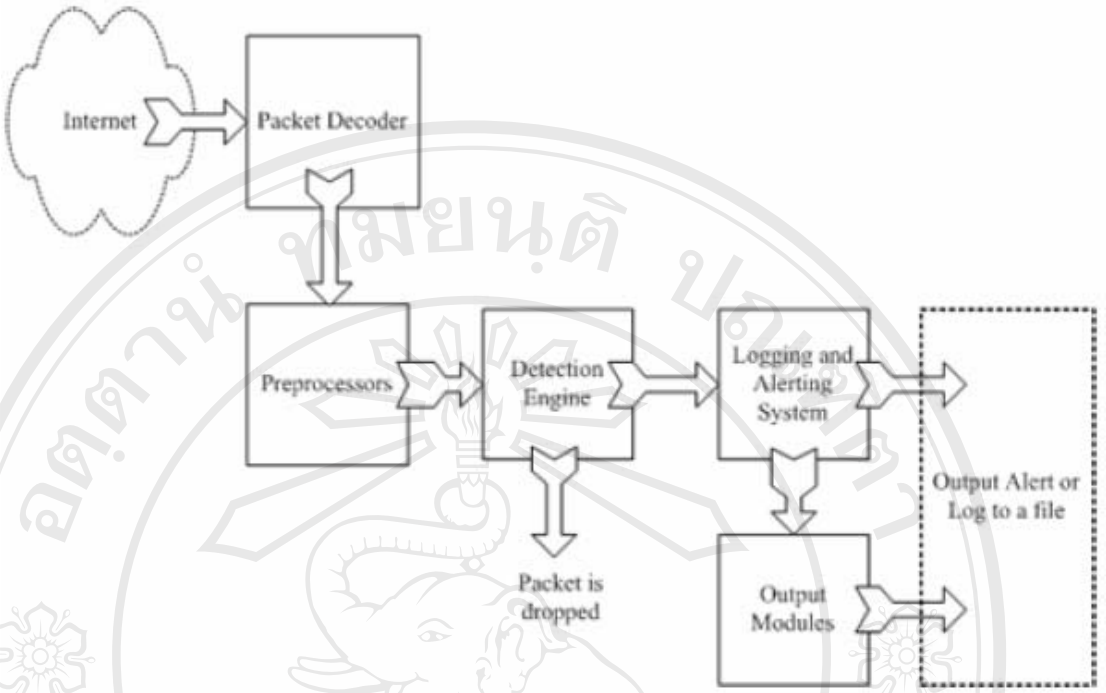
3. การรายงานผลเป็นภาษาไทย หมายถึงระบบรายงานผลเชิงสถิติมีส่วนโต้ตอบที่เป็นภาษาไทยเพื่อให้ง่ายต่อการใช้งาน

4. การจัดการเกี่ยวกับกฎเพื่อปรับแต่งเงื่อนไขในการตรวจสอบการบุกรุก หมายถึงมีระบบในการปรับแต่งแก้ไขกฎที่ใช้ในการตรวจสอบผู้บุกรุก

3.2 ขั้นตอนการศึกษา

3.2.1 ศึกษาหลักการทำงานของโปรโตคอล (Protocol) ทีซีพี/ไอพี (TCP/IP)

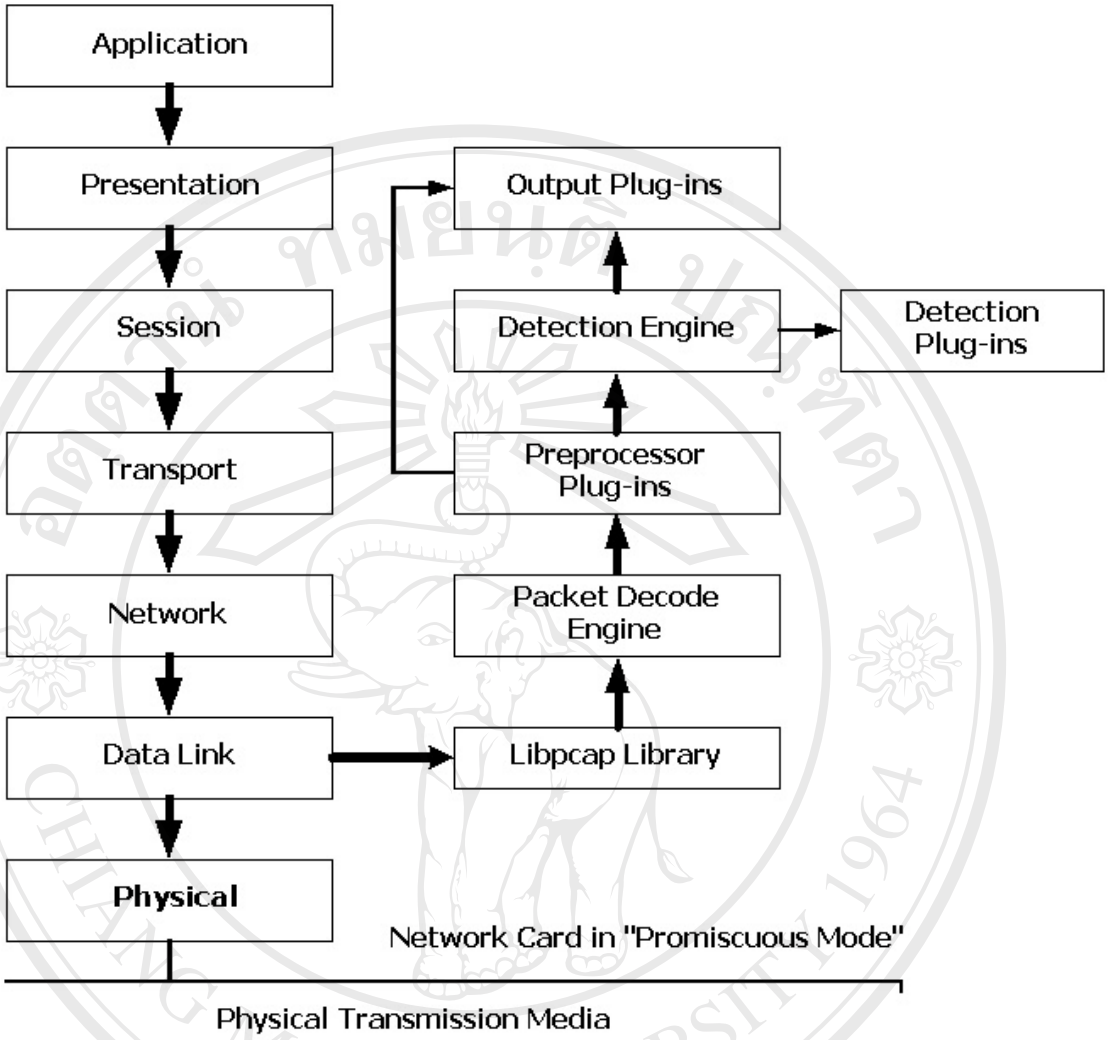
เป็นขั้นตอนศึกษาหลักการทำงานของชุดโปรโตคอล ทีซีพี/ไอพี ที่ใช้เป็นโปรโตคอลหลักในการติดต่อสื่อสารกันในระบบเครือข่ายอินเทอร์เน็ตโดยศึกษามุ่งเน้นไปที่วิธีการรับและส่งข้อมูล เพื่อให้ทราบหลักการติดต่อกันระหว่างเครื่องคอมพิวเตอร์ในเครือข่าย



รูปที่ 3.1 ภาพแสดงการทำงานส่วนประกอบหลักของโปรแกรม Snort

3.2.2 ศึกษาหลักการทำงานของระบบการตรวจจับการบุกรุกแบบเครือข่าย

สำหรับโปรแกรมที่นำมาใช้เป็นระบบตรวจสอบผู้บุกรุกระบบเครือข่ายสำหรับการศึกษาระบบป้องกันและตรวจสอบผู้บุกรุกเครือข่ายนี้คือโปรแกรม Snort เวอร์ชัน 2.2.0 ซึ่งมีส่วนประกอบที่สำคัญ 4 ส่วนดังรูปที่ 3.1 และเมื่อนำมาเทียบกับการทำงานกับ OSI โมเดลจะเทียบได้ดังรูปที่ 3.2



รูปที่ 3.2 ภาพแสดงการทำงานของโปรแกรม Snort เทียบกับโมเดล OSI

องค์ประกอบทั้ง 4 ส่วนสามารถอธิบายได้ดังนี้

3.2.2.1 Packet Decode Engine

Packet Decode Engine ทำหน้าที่รับข้อมูลจากส่วนที่เชื่อมต่อกับระบบเครือข่ายผ่านทางไลบารี Libpcap แล้วทำการถอดรหัส (Decode) ข้อมูลเพื่อส่งไปยังส่วนอื่นต่อไป แพ็กเก็ตที่วิ่งอยู่บนระบบเครือข่ายจะถูกดักจับเข้าสู่ระบบตรวจสอบผู้บุกรุกผ่านทางการทำงานในส่วนนี้ โดยอาศัยกลไกดังนี้

1. เซ็ท Network Interface Card (NIC) ให้อยู่ในโหมด Promiscuous
2. ดักจับแพ็กเก็ตเข้าสู่ Network Interface Card ผ่านทางไลบารี Libpcap

เมื่อแพ็กเก็ตข้อมูลเดินทางจากระบบเครือข่ายเข้ามายัง NIC ที่ทำงานในโหมด Promiscuous จะส่งต่อไปยังส่วน Packet Decode Engine โดยผ่านทางไลบารี Libpcap เพื่อทำการถอดรหัสข้อมูลที่รับมาจากเลเยอร์ Data Link ซึ่งจะทำให้สามารถแยกประเภทของโปรโตคอล

ต่างๆในระดับเลเยอร์นี้ได้ เช่น Ethernet, 802.11 หรือ Token Ring เป็นต้น รวมถึงโปรโตคอลในระดับที่สูงขึ้นไปด้วย ได้แก่ IP, TCP และ UDP ในระหว่างกระบวนการถอดรหัสนี้ Decode Engine จะทำการลิงค์ข้อมูลดิบที่ได้มาไปยังโครงสร้างสำหรับการวิเคราะห์ต่อภายหลังในส่วนของ Preprocessors และ Detection Engine ต่อไป

3.2.2.2 Preprocessors

Preprocessor Plug-ins เป็นส่วนที่ทำหน้าที่รับแพ็กเก็ตข้อมูลที่ผ่านการถอดรหัสเข้ามาเพื่อเตรียมความพร้อมแพ็กเก็ตก่อนส่งไปตรวจสอบยังส่วน Detection Engine ต่อไป ก่อนที่แพ็กเก็ตจะถูกส่งไปตรวจสอบต่อยังส่วน Detection Engine จะต้องผ่านส่วน Preprocessor นี้ก่อนเพื่อตรวจสอบความถูกต้องเบื้องต้นของข้อมูลและจัดโครงสร้างข้อมูลให้อยู่ในรูปแบบที่พร้อมถูกตรวจสอบในส่วนต่อไป โดยสามารถแยกเป็นส่วนการทำงานย่อยได้ดังนี้

1. The _decode Family of Preprocessors

หน้าที่ประการหนึ่งของ Preprocessor คือการทำให้แพ็กเก็ตอยู่ในรูปแบบที่เป็นมาตรฐานก่อนส่งต่อไปยังส่วน Detection Engineได้แก่ Telnet_decode, Ftp_decode และ Rpc_decode

2. The Frag2 Preprocessor

การแฟร็กเมนเตชัน (Fragmentation) ถือเป็นเรื่องปกติที่จำเป็นในการส่งข้อมูลเพื่อติดต่อสื่อสารกันบน Internet Protocol (IP) เพราะเนื่องจากดาตาแกรม[1] (Datagram) ที่ถูกส่งจาก IP ถ้ามีขนาดใหญ่กว่า MTU[2] ในเลเยอร์ Physical นั้น โปรโตคอล IP จะทำการแตกข้อมูลดาตาแกรมให้เป็นส่วนเล็กๆก่อนส่งเข้าไปยังอินเตอร์เฟสของเครือข่ายนั้นๆ เมื่อถึงปลายทางแพ็กเก็ตที่ถูกแบ่งเป็นส่วนเล็กๆเหล่านี้ก็จะถูกประกอบกลับคืนมาอีกครั้ง เรียกว่าการรีแอสเซมเบิล (Reassemble)

การแบ่งแพ็กเก็ตออกเป็นส่วนย่อยๆ สามารถใช้เป็นวิธีหนึ่งในการเลี่ยงผ่านการตรวจสอบจากระบบตรวจสอบผู้บุกรุกเครือข่ายประเภท Pattern-based หรือ Signature-based ได้ด้วย เพราะการแบ่งแพ็กเก็ตเป็นส่วนๆจะทำให้เกิดความยุ่งยากในการตรวจสอบสำหรับระบบตรวจสอบผู้บุกรุกประเภทนี้ ซึ่งจะเปรียบเทียบแพ็กเก็ตแบบทีละรายๆกับรูปแบบการบุกรุกที่เตรียม

[1]ชื่อเรียกแพ็กเก็ตที่ถูกส่งจาก Internet Protocol ในชั้นเลเยอร์ Network

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

ไว้อ้างอิง 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 ซึ่งจะใช้การโจมตีที่เลียนแบบการบุกรุกที่มีในฐานข้อมูลรูปแบบการบุกรุกของโปรแกรม 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[3]ขึ้น เมื่อการเชื่อมต่อสำเร็จ การส่งข้อมูลจึงเริ่มขึ้นและเมื่อเสร็จสิ้นการส่งข้อมูล การเชื่อมต่อก็จะสิ้นสุดลง โดยในระหว่างการเชื่อมต่อที่เกิดขึ้น Stream4 Preprocessor จะสร้างตารางสำหรับเก็บเซสชันต่างๆที่เกิดขึ้น (และลบเซสชันนั้นออกจากตารางเมื่อการเชื่อมต่อสิ้นสุดลง) เพื่อใช้ในการตรวจสอบเซสชันทั้งหมดของการเชื่อมต่อนั้นๆ โดยเมื่อส่วน Detection Engine จะทำการตรวจสอบแพ็กเก็ต จะมีการตรวจสอบก่อนว่าแพ็ก

[3] ดูรูปที่ 2.13 หน้า 38

เก็ตนั้นเป็นส่วนหนึ่งที่เกิดจากเชื่อมต่อเซสชันด้วยหรือไม่ 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 กลับถ้าพอร์ตนั้นเปิดอยู่ และจะส่งแพ็กเก็ต SYN/RST กลับถ้าพอร์ตนั้นปิดอยู่ ฉะนั้นการส่งแพ็กเก็ต SYN ไปยังพอร์ตต่างๆที่เครื่องเป้าหมาย ก็จะสามารถทราบได้ว่าเครื่องนั้นเปิดพอร์ตไหนอยู่บ้าง โดยดูจากแพ็กเก็ตที่ตอบกลับมา

โปรแกรม Snort สามารถตรวจสอบการสแกนพอร์ตได้โดยใช้ Preprocessor ในส่วนนี้ ซึ่งในเวอร์ชัน 2.2.0 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 นั้น

เป็นส่วนหนึ่งของการเชื่อมต่อ 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

5. Preprocessors อื่นๆ

ตารางที่ 3.1 แสดง Preprocessors ประเภทอื่นๆ[4]

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

3.2.2.3 Detection Engine แพ็กเก็ตที่รับเข้ามาจากเครือข่าย หลังจากผ่านการทำงานในสองส่วนแรกแล้ว จะถูกเก็บเข้าสู่โครงสร้างข้อมูล (Data Structure) ซึ่งผ่านกระบวนการจัด

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

โครงสร้าง 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

1.) สมรรถนะของเครื่องที่โปรแกรม Snort ทำงานอยู่
2.) จำนวนของกฎที่ใช้ในการเปรียบเทียบข้อมูลการบุกรุก
3.) ความเร็วของช่องทางที่เชื่อมต่อระบบเครือข่ายของเครื่องที่โปรแกรม Snort ทำงานอยู่
4.) ความหนาแน่นของข้อมูลบนเครือข่าย

ฉะนั้นการออกแบบระบบตรวจสอบผู้บุกรุกบนเครือข่าย นอกจากต้องคำนึงถึงตำแหน่งการวางระบบแล้ว ยังต้องพิจารณาองค์ประกอบดังกล่าวด้วย

ระบบตรวจสอบแพ็กเก็ตในส่วนนี้จะสามารถจำแนกส่วนต่างๆของแพ็กเก็ต เพื่อใช้ในการนำไปเปรียบเทียบกับกฎต่างๆได้ โดยสามารถแยกส่วนต่างๆของแพ็กเก็ตที่ใช้ในการวิเคราะห์ได้เป็น

1.) IP Header
2.) Transport Layer Header (TCP Header และ UDP Header)
3.) Application Layer Header (เช่น DNS Header, FTP Header, SNMP Header เป็นต้น)

การใช้วิธีการตรวจสอบกฎทั้งหมดที่มีกับแพ็กเก็ตหนึ่งๆ 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

Action เป็นส่วนที่ใช้ระบุประเภทของ Action เมื่อข้อมูลภายในแพ็กเก็ตตรงตามเงื่อนไขของกฎ ประกอบด้วย

1) Pass เป็น Action ให้เพิกเฉยต่อแพ็กเก็ตนี้ และปล่อยให้การตรวจสอบดำเนินต่อไปกับกฎข้อที่เหลือ

2) Log เป็น Action ให้ทำการเก็บบันทึกข้อมูลของแพ็กเก็ตนั้น ตามที่กำหนดไว้ในไฟล์คอนฟิกุเรชันของตัวเซ็นเซอร์ เช่น เก็บบันทึกลงฐานข้อมูลต่างๆ เป็นต้น

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

4) Activate เป็น Action ที่ใช้เมื่อต้องการให้ตรวจสอบแพ็กเก็ตนี้ด้วยกฎข้ออื่น (ที่ใช้ Dynamic Action) อีก

5) Dynamic เป็น Action ที่ถูกใช้โดยกฎข้ออื่นที่ใช้ Activate Action เพื่อให้ทำการตรวจสอบแพ็กเก็ตจากกฎข้อนั้นด้วยกฎในข้อที่เป็น Dynamic Action อีก

Protocol เป็นส่วนที่ใช้ระบุชนิดโปรโตคอลของแพ็กเก็ตที่กฎจะใช้พิจารณาตรวจสอบ ประกอบด้วยโปรโตคอล IP, ICMP, TCP และ UDP

Address เป็นส่วนที่ใช้ระบุแอดเดรส ประกอบด้วยสองส่วนได้แก่

1) Source Address ใช้ระบุแหล่งที่มาของแพ็กเก็ต

2) Destination Address ใช้ระบุแหล่งปลายทางของแพ็กเก็ต

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

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

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

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

Port Number ใช้ระบุหมายเลขพอร์ตในแพ็กเก็ตจากต้นทางไปยังปลายทาง โดยสามารถระบุแบบเจาะจงหรือระบุเป็นช่วงหมายเลขพอร์ตก็ได้ เช่น ถ้าระบุ Source port 23 หมายถึงการตรวจสอบแพ็กเก็ตที่เริ่มต้นมาจาก Telnet เซิร์ฟเวอร์

วิธีระบุหมายเลขพอร์ตได้แก่

1) Port Ranges เช่น ตั้งแต่พอร์ต 1024-2048

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

2) 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";)

3) Negation Symbol เช่น ต้องการระบุพอร์ตทุกพอร์ตยกเว้นพอร์ตที่ระบุ

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

Direction ใช้ระบุทิศทางในการพิจารณาแหล่งที่มาและปลายทางของแพ็กเก็ต โดยสัญลักษณ์ที่ใช้ได้แก่

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

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

3) <> 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.) 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

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

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!";)

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

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

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

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

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”**;)

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";)

10.) Flow Control เป็นออปชันที่ใช้ในการกำหนดการเชื่อมต่อของแพ็กเก็ตบนโปรโตคอล TCP ใช้งานร่วมกับออปชันอื่น โดยการใช้งานจะอยู่ในรูปแบบ flow: <OPTION> โดยค่า OPTION ที่สามารถกำหนดได้มีดังตารางที่ 3.3

ตารางที่ 3.3 Flow Control ออปชัน

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.

ตัวอย่างเช่น

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

ใช้ในการตรวจสอบแพ็กเก็ตที่ถูกส่งจากเครื่องเซิร์ฟเวอร์ที่มีการเรียกดูไฟล์รหัสผ่านของระบบปฏิบัติการยูนิกซ์ไปยังเครื่องภายนอก

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)

1.) Fragmentation Bits ในส่วนหัวของแพ็กเก็ต IP ประกอบด้วย Flag bit สามตัว ซึ่งใช้ในการแฟร็กเมนเตชันและรีแอสแซมบลีแพ็กเก็ต ได้แก่

ก. R (Reserved Bit) ซึ่งถูกสงวนไว้ใช้สำหรับการใช้งานในอนาคต

ข. D (Don't Fragment Bit) ถ้าบิตนี้ถูกเซ็ทแสดงว่าแพ็กเก็ต IP นี้ไม่ควรถูกแฟร็กเมนต์

ค. 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.) Equivalent Source and Destination IP Option ใช้ในการตรวจสอบแพ็กเก็ตที่มีการเปลี่ยนหรือปลอมแปลงให้มีแหล่งต้นทางและปลายทางเป็นที่เดียวกัน ตัวอย่างเช่น

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

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

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

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

ซึ่งผู้บุกรุกสามารถใช้ออปชันเหล่านี้ในการสำรวจข้อมูลบนระบบ เครือข่าย ได้เช่น การใช้ออปชัน lsrr และ ssrr ช่วยให้ผู้บุกรุกสามารถตรวจสอบได้ว่ามีเส้นทางนั้นบนระบบเครือข่ายอยู่จริงหรือไม่ ตัวอย่างเช่น

alert ip any any -> any any (**ipopts: lsrr**; msg: “Loose source routing attempt”;)

4.) ID Option ใช้ในการตรวจสอบค่าหมายเลขแฟร็กเมนต์ในฟิลด์ Fragment ID ในส่วนหัวของแพ็กเก็ต IP โดยการระบุค่าหมายเลขที่ต้องการตรวจสอบ ในรูปแบบ id: <ID_NUMBER>

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

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

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

1.) Sequence Number Option ใช้ในการตรวจสอบหมายเลขลำดับของแพ็กเก็ตในฟิลด์ Sequence Number บนแพ็กเก็ต TCP ในรูปแบบ seq: <SEQUENCE_NUMBER>

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

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

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.

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

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";)

2.) ICMP Sequence Option ใช้ในการตรวจสอบค่าในฟิลด์ Sequence Number ในส่วนหัวของแพ็กเก็ต ICMP ตัวอย่างเช่น

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

3.) ICMP itype Option ใช้ในการตรวจสอบค่าในฟิลด์ Type ในส่วนหัวของแพ็กเก็ต ICMP เพื่อตรวจสอบการบุกรุกโดยการใช้ที่ระบุในฟิลด์นี้ ตัวอย่างเช่น

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

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

1.) Snort ID Option ค่า SID เป็นค่าที่ใช้ในการจำแนกประเภทของกฎ ซึ่งถูกใช้ทั้งในส่วนของโมดูล Output และ Logging ในการอ้างอิงถึงกฎ โดยค่า SID แบ่งได้สามช่วงดังนี้

ตารางที่ 3.6 ช่วงค่า Snort ID

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

ตัวอย่างเช่น

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

2.) Rule Revision Number ใช้แสดงค่ารีวิชัน (Revision Number) ในกรณีที่มีการแก้ไขกฎข้อนี้จากรูปแบบดังเดิม ตัวอย่างเช่น

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

3.) Severity Identifier Option ใช้ระบุค่าลำดับความร้ายแรงของกฎ ซึ่งมีอยู่ สามระดับ ได้แก่ 1, 2 และ 3 โดยที่ค่า 1 หมายถึงร้ายแรงที่สุด ตัวอย่างเช่น

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

4.) Classification Identifier Option ใช้ในการระบุประเภทความร้ายแรงในการบุกรุกสำหรับกฎต่างๆในโปรแกรม Snort ซึ่งแบ่งประเภทตามลำดับความเสี่ยงได้ดังนี้

ตารางที่ 3.7 ประเภทความเสี่ยงสูง (ค่า Priority 1)

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 3.8 ประเภทความเสี่ยงปานกลาง (ค่า Priority 2)

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 3.9 ประเภทความเสี่ยงต่ำ (ค่า Priority 3)

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

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.) 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

1.) Messages Option ใช้ในการระบุข้อความที่ใช้ในการแจ้งเตือนภัยหรือที่ปรากฏในล็อกไฟล์ที่บันทึกกฎข้อนี้ ตัวอย่างเช่น

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

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/

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

ตารางที่ 3.10 อาร์กูเมนต์ที่ใช้ร่วมกับคีย์เวิร์ด tag

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

ตัวอย่างเช่น

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

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";)

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.) Real-Time Countermeasures Option เป็นออปชันที่สำคัญมาก ใช้ในการหยุดยั้งการบุกรุกที่ตรวจพบเจอแบบอัตโนมัต โดยการส่งแพ็กเก็ตตอบสนองกลับไปยังแหล่งที่มาของแพ็กเก็ตนั้น โดยมีการตอบสนองในรูปแบบต่างๆ ดังตาราง

ตารางที่ 3.11 อาร์กูเมนต์ที่ใช้ร่วมกับคีย์เวิร์ด resp

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.

ตัวอย่างเช่น

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

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

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

3.2.2.4 Output Plug-ins ทำหน้าที่สำคัญในการแสดงผลลัพธ์ของโปรแกรม Snort เก็บบันทึกข้อมูลการ บุกรุก และส่งสัญญาณเตือนภัย ซึ่งส่วนปลั๊กอิน Output นี้จะถูกเรียกใช้โดยส่วนอื่นๆในโปรแกรม Snort ทั้งสิ้น ได้แก่

1. ส่วน Packet Decode Engine ใช้ปลั๊กอิน Output ในการออกผลลัพธ์ของข้อมูลแพ็กเก็ตที่รับเข้ามาในรูปแบบของ TCPDump และ ASCII decode

2. ส่วน Preprocessor ใช้ปลั๊กอิน Output ในการส่งสัญญาณเตือนภัย เช่นในส่วนของ PortScan2 Preprocessor

3. ส่วน Detection Engine ใช้ปลั๊กอิน Output ในการส่งสัญญาณเตือนภัยและเก็บบันทึกข้อมูล

ซึ่งขึ้นอยู่กับการกำหนดในไฟล์คอนฟิกุเรชันของโปรแกรม Snort โดยสามารถกำหนดเอาท์พุตในรูปแบบต่างๆได้ดังนี้

1.) บันทึกลงไฟล์ /var/log/snort/alerts ในรูปแบบของเท็กซ์ไฟล์

2.) ส่งสัญญาณเตือนภัยแบบ SNMP

3.) ส่งข้อมูลต่อไปให้โปรแกรม Syslog เพื่อบันทึกลงล็อกไฟล์ของระบบ

4.) บันทึกลงฐานข้อมูล เช่น MySql, Postgresql, Oracle, MS-Sql และ UnixODBC ในกรณีที่ใช้ฐานข้อมูลอื่น เช่น DB2, Informix หรืออื่นๆ สามารถใช้ UnixODBC เป็นตัวกลางในการเชื่อมต่อได้

5.) แปลงผลลัพธ์ให้อยู่ในรูปแบบของ eXtensible Markup Language (XML)

6.) ส่งสัญญาณเตือนภัยไปยังอุปกรณ์ที่ทำหน้าที่รักษาความปลอดภัยเช่น เราเตอร์ หรือไฟร์วอลล์

7.) ส่งสัญญาณเตือนภัยแบบ Server Message Block (SMB)

3.2.3 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 เป็นการกำหนดค่าเน็ตเวิร์คสำหรับภายนอกเป็นทุกๆไอพีแอดเดรส
- var DNS_SERVERS [10.4.28.10, 10.4.0.2] เป็นการกำหนดไอพีแอดเดรสสำหรับเซิร์ฟเวอร์ DNS เป็นต้น

2. การกำหนดค่า Preprocessor

Preprocessor จะถูกเรียกทำงานในส่วน Detection Engine ของโปรแกรม Snort ซึ่งมีรูปแบบในการกำหนดได้แก่

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 ใช้ระบุตำแหน่งไฟล์กฎตัวอย่างการกำหนดเช่น

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/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

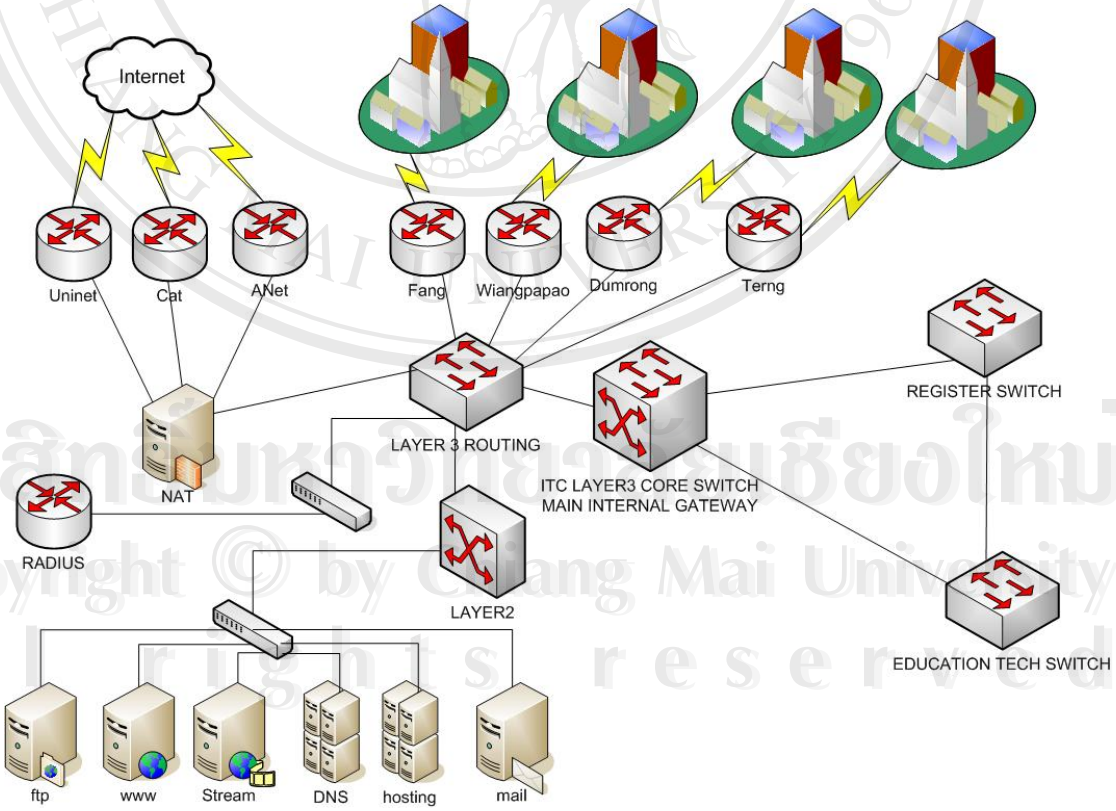
include $RULE_PATH/porn.rules

include $RULE_PATH/info.rules

include $RULE_PATH/icmp-info.rules

include $RULE_PATH/virus.rules

3.2.4 ศึกษาโครงสร้างของระบบเครือข่ายภายในของมหาวิทยาลัยราชภัฏเชียงราย



รูปที่ 3.5 แสดงโครงสร้างเครือข่ายมหาวิทยาลัยราชภัฏเชียงราย

เครือข่ายมหาวิทยาลัยราชภัฏเชียงรายมีโครงข่ายหลัก (Back bone) เป็นแบบกิกาบิต อีเธอร์เน็ต (Gigabit Ethernet) สามารถรับส่งข้อมูลสูงสุดที่ 1000M ต่อวินาที โดยแบ่งเป็นสามส่วนที่เชื่อมต่อกันได้แก่ อาคารสำนักบริการเทคโนโลยีสารสนเทศ, อาคารสำทะเบียนและวัดผล, อาคารโปรแกรมวิชาเทคโนโลยีและนวัตรกรรมการศึกษา โดยทั้งสามส่วนทำการเชื่อมต่อด้วยเส้นใยแก้วนำแสง (Fiber Optic) แบบ Single Mode ทั้งสามส่วนจะทำหน้าที่เชื่อมโยงเครือข่ายของอาคารสำนักงานที่อยู่ใกล้เคียง ด้วยเส้นใยแก้วนำแสงแบบ Multi Mode โดยมีเส้นทางในการออกสู่เครือข่ายอินเทอร์เน็ตหลักอยู่ที่อาคารสำนักบริการเทคโนโลยีสารสนเทศ

ปัจจุบันมหาวิทยาลัยราชภัฏเชียงรายทำการเชื่อมต่อกับผู้ให้บริการอินเทอร์เน็ต (internet Services Provider) อยู่ 3 แห่งได้แก่ Uninet มีความกว้างของช่องสัญญาณ 2 Mbps, บริษัท ทศท คอร์ปอเรชั่นจำกัด (มหาชน) มีความกว้างของช่องสัญญาณ 2 Mbps และบริษัท A-net มีความกว้างของช่องสัญญาณรวม 2 Mbps

การออกสู่เครือข่ายอินเทอร์เน็ตจะต้องอาศัยเครื่องที่ทำหน้าที่เปลี่ยนแปลงเลขไอพีแอดเดรส หรือที่นิยมเรียกกันว่า NAT (Network Address Translator)

3.3 ออกแบบระบบตรวจจับการบุกรุก ภายในเครือข่ายมหาวิทยาลัยราชภัฏเชียงราย

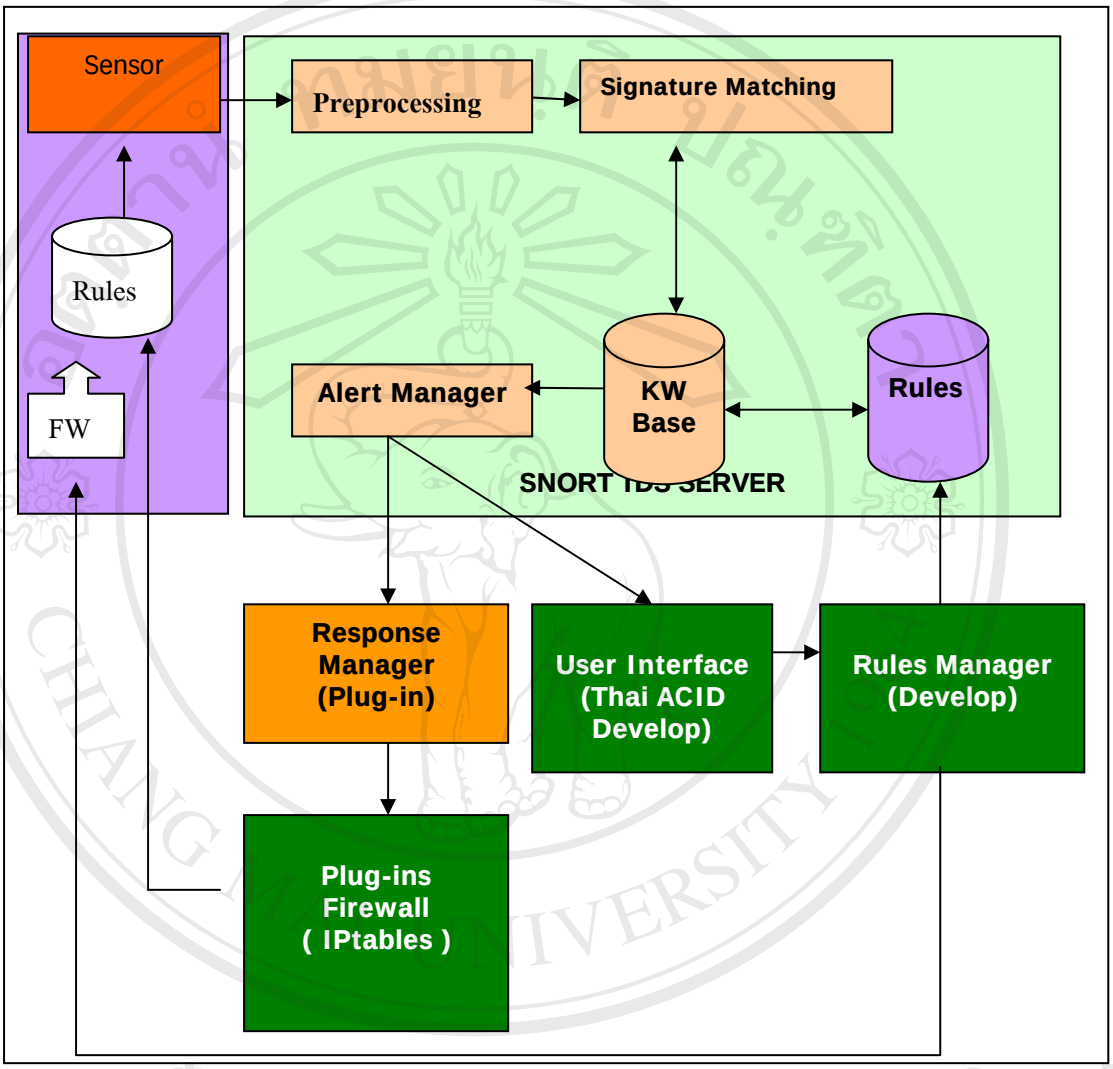
3.3.1 การออกแบบโครงสร้างภาพรวมของระบบ

Snort นับว่าเป็นระบบตรวจสอบผู้บุกรุกที่ได้รับความนิยมในการใช้งานในหมู่ผู้ดูแลระบบเป็นจำนวนมาก เนื่องจากมีข้อดีหลายประการ อาทิเช่น

- เป็นซอฟต์แวร์ที่ไม่ต้องเสียค่าใช้จ่ายในการทำงาน
- มีกฎเพื่อใช้ในการตรวจสอบที่หลากหลาย
- สามารถติดตั้งโดยในระบบปฏิบัติการที่หลายหลาย
- มีหลายโหมดในการใช้งาน
- เป็นที่ยอมรับและมีการใช้งานในเชิงพานิชย์

แต่อย่างไรก็ตามในการประยุกต์ใช้งานนั้นยังมีขั้นตอนที่ยุ่งยากสำหรับผู้ดูแลระบบที่ยังขาดประสบการณ์และการปรับแต่งเรื่องกฎยังทำได้ลำบากอยู่

สำหรับการออกแบบระบบการตรวจจับการบุกรุกในเครือข่ายของมหาวิทยาลัยราชภัฏเชียงรายนั้นมีโครงสร้างดังนี้



รูปที่ 3.6 แสดงโครงสร้างการทำงานของระบบการตรวจจับการบุกรุกเครือข่าย

จากรูปที่ 3.6 สามารถอธิบายดังนี้

1). Sensor: ทำหน้าที่เป็นตัวตรวจจับเหตุการณ์ต่างๆ ที่เกิดขึ้นในหรือเครือข่าย

2). Pre-processing: ทำหน้าที่จัดรูปแบบของข้อมูลที่รับเข้ามาเพื่อให้นำไปประมวลผลในขั้นถัดไป

3). Signature matching: ทำหน้าที่วิเคราะห์ข้อมูล จากพฤติกรรมการบุกรุกที่มีรูปแบบหรือมีการทำซ้ำๆ ซึ่งส่วนนี้ต้องมีข้อมูลมากพอที่จะวิเคราะห์ได้ว่าพฤติกรรมที่ปรากฏในระบบเป็นรูปแบบของการบุกรุกหรือไม่

4). Knowledge Base: ทำหน้าที่เก็บข้อมูลเกี่ยวกับพฤติกรรมของการบุกรุก โดยข้อมูลนี้ถูกใช้โดยส่วนของ signature matching

5). Alert Manager: ทำหน้าที่เป็นตัวตัดสินใจว่าจะเหตุการณ์ที่เกิดขึ้นในระบบควรจะต้องเตือนหรือไม่

6). User Interface: ทำหน้าที่โต้ตอบกับผู้ใช้ในการควบคุมการทำงานหรือการแจ้งเตือนผลลัพธ์ที่ได้จาก ส่วนอื่นๆ ออกมา นอกจากนี้ user interface ยังเป็นส่วนโต้ตอบระหว่างผู้ใช้กับระบบเพื่อเปลี่ยนแปลงหรือ update ข้อมูลใน knowledge base โดยผ่านทาง Rules Manager

7). Response Manager: รับข้อมูลจาก alert manager เพื่อนำมาตัดสินใจว่าจะโต้ตอบกับการบุกรุกอย่างไรซึ่งเมื่อมีการตัดสินใจจะทำการสร้าง Rules ของ Firewall เพื่อไป Update Firewall เครื่องที่อยู่ในระบบเครือข่าย

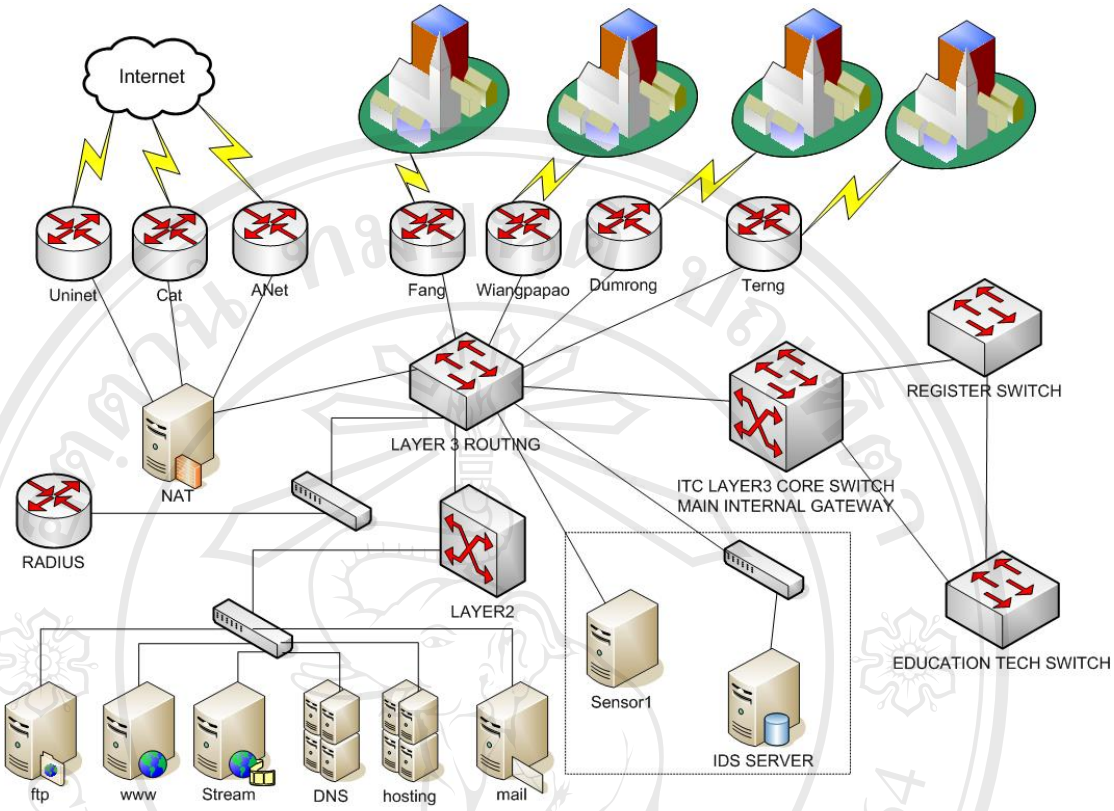
3.3.2 การจัดวางตำแหน่งของระบบในเครือข่ายภายในมหาวิทยาลัยราชภัฏเชียงราย

แนวคิดในการออกแบบระบบตรวจจับการบุกรุกเครือข่ายของมหาวิทยาลัยราชภัฏเชียงรายมาจากปัจจัย 3 ประการคือ

1. โครงสร้างพื้นฐานแบ่งเป็นสามพื้นที่ในการให้บริการดังนั้นในการตรวจควรมีเซ็นเซอร์[5]อย่างน้อย สามจุด
2. เพื่อให้การจัดการเป็นแบบรวมศูนย์จึงแยกฐานข้อมูลออกมาจากเครื่องเซ็นเซอร์เพื่อความง่ายในการรวบรวมข้อมูล
3. ต้องทำการวางเซ็นเซอร์ในจุดที่มีข้อมูลไหลผ่าน

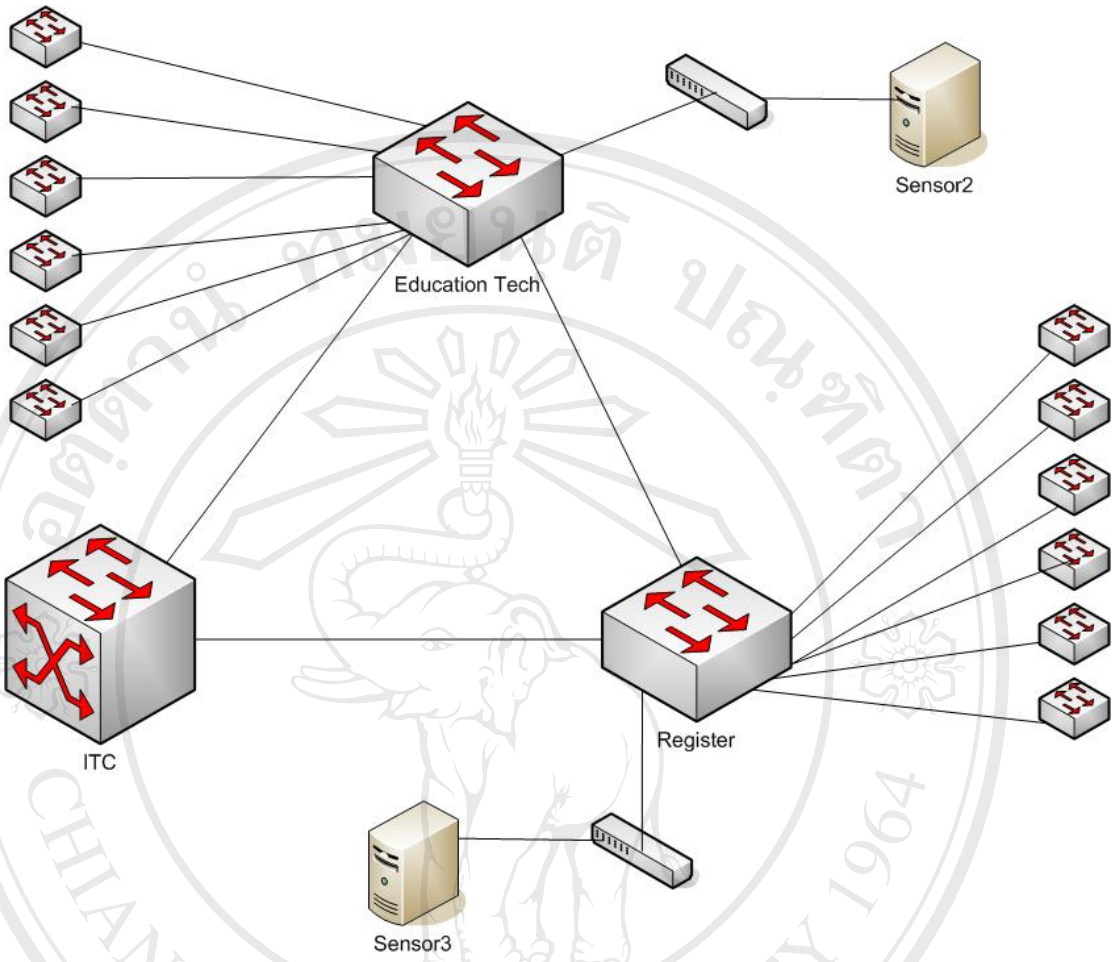
ดังนั้นจึงออกแบบการวางตำแหน่งของเซ็นเซอร์ได้ดังรูปที่ 3.7 และรูปที่ 3.8

[5] เซ็นเซอร์ (Sensor) คือเครื่องคอมพิวเตอร์ที่ทำการติดตั้งโปรแกรมสนอร์ท (Snort) แต่มิได้จับเก็บข้อมูลการบุกรุกไว้ในตัวเครื่อง



รูปที่ 3.7 แสดงตำแหน่งของเซ็นเซอร์ในเครือข่ายของสำนักบริการเทคโนโลยีสารสนเทศ

จากรูปที่ 3.7 เป็นการวางตำแหน่งของเซ็นเซอร์เพื่อตรวจจับการบุกรุกในเครือข่ายของสำนักบริการเทคโนโลยีสารสนเทศ โดยเลือกวางในตำแหน่งของ Layer3 Routing Switch เพื่อทำการตรวจจับข้อมูลที่รับส่งมาจาก Core switch และเครือข่ายภายใน และได้มีการวาง IDS Server ซึ่งทำหน้าที่เก็บข้อมูลการบุกรุกที่ได้จากเซ็นเซอร์

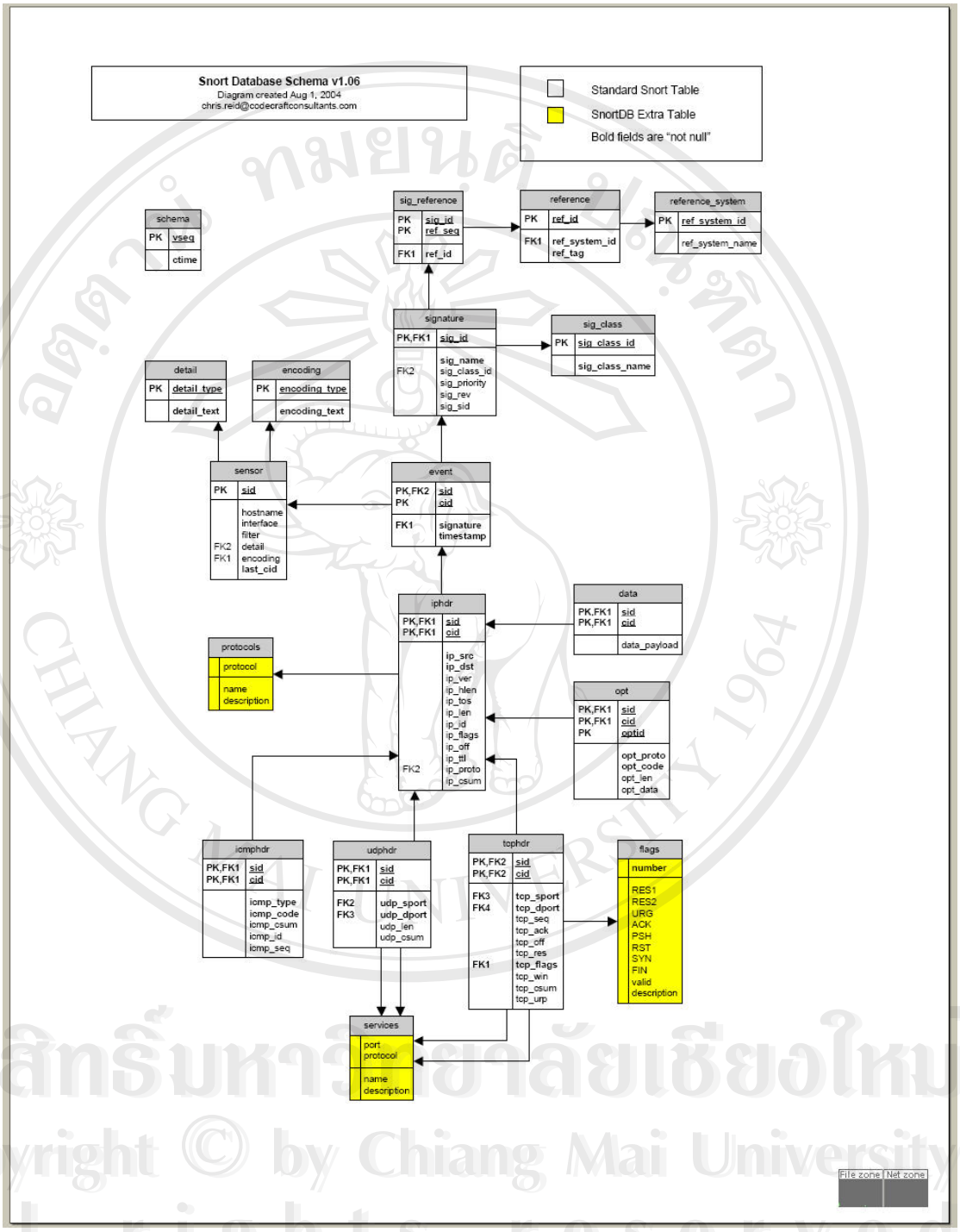


รูปที่ 3.8 แสดงตำแหน่งของ Sensor ในเครือข่ายสำนักทะเบียนและประมวลผลและอาคารเทคโนโลยีและนวัตกรรมการศึกษา

จากรูปที่ 3.8 ตำแหน่งของจะมีการเชื่อมต่อเข้าไปในอุปกรณ์สลับสัญญาณเพื่อรับข้อมูลที่รับส่งภายในเครือข่าย

3.3.3 การพัฒนาส่วนที่ติดต่อจัดการเกี่ยวกับเรื่องกฎและการแสดงผลทางด้านสถิติ

ในการออกแบบฐานข้อมูลเพื่อใช้สำหรับการจัดการเรื่องกฎของ snort นั้นผู้เขียนได้ออกแบบฐานข้อมูลเพิ่มขึ้นจากฐานข้อมูลเดิมของสนอร์ทที่มีใช้งานอยู่แล้วแต่ ไม่ได้จัดการเรื่องกฎเกณฑ์ต่างโดยฐานข้อมูลของสนอร์ทนั้นได้มุ่งจัดเก็บข้อมูลการบุกรุกจากเครื่องเซ็นเซอร์เพียงอย่างเดียว โดยโครงสร้างของฐานข้อมูลสนอร์ทนั้นแสดงได้ดังรูปที่ 3.9



รูปที่ 3.9 แสดงโครงสร้างของฐานข้อมูลที่เซ็นเซอร์ใช้จัดเก็บ

จากรูปที่ 3.9 นั้นจะเป็นโครงสร้างของฐานข้อมูลที่เซ็นเซอร์ใช้ ซึ่งประกอบไปด้วยตารางดังต่อไปนี้

ตารางที่ 3.12 แสดงตารางของสนอร์ทในการจัดเก็บข้อมูลการบุกรุก

ตาราง	รายละเอียด
data	เก็บข้อมูล payload ของการบุกรุก
detail	ข้อมูลโหมดการจัดเก็บ (Full , Fast)
encoding	เก็บประเภทการจัดเก็บ (hex,base64,ascii)
event	เก็บจัดเก็บข้อมูลที่เกี่ยวข้องกับเหตุการณ์ (ช่วงเวลาที่เกิด,ประเภทของเหตุการณ์)
icmphdr	เก็บข้อมูลส่วนหัวของโปรโตคอล icmp
iphdr	เก็บข้อมูลส่วนหัวของโปรโตคอล ip
opt	เก็บข้อมูลออปชั่นเพิ่มเติม
reference	เก็บข้อมูลการอ้างอิงไปยังเวบไซต์ที่เกี่ยวข้อง
reference_system	เก็บข้อมูลประเภทการอ้างอิง (url,cve,bugtraq,nessus)
schema	เก็บข้อมูลเวอร์ชันของโครงสร้างฐานข้อมูล
sensor	เก็บข้อมูลของเซ็นเซอร์
sig_class	เก็บข้อมูลประเภทของรูปแบบการบุกรุก
sig_reference	เก็บข้อมูลการอ้างอิงที่เกี่ยวข้องกับประเภทของการบุกรุก
signature	เก็บข้อมูลรูปแบบการบุกรุกที่ตรวจพบ
tcphdr	เก็บข้อมูลส่วนหัวของโปรโตคอล tcp
udphdr	เก็บข้อมูลส่วนหัวของโปรโตคอล udp

ในการออกแบบระบบเพื่อให้สามารถปรับแต่งกฎเกณฑ์ต่างๆ ให้สามารถแก้ไขได้อย่างสะดวกมากยิ่งขึ้นใน ได้ออกแบบฐานข้อมูลเพิ่มเติมจากเดิมดังต่อไปนี้

ตารางที่ 3.13 แสดงตารางที่ได้ออกแบบเพิ่มเติมสำหรับจัดการเรื่องกฎการตรวจสอบ

ตาราง	รายละเอียด
acid_ag	เก็บข้อมูลประเภทกลุ่มการแจ้งเตือน
acid_ag_alert	เก็บข้อมูลกลุ่มการแจ้งเตือน
acid_event	เก็บข้อมูลเหตุการณ์

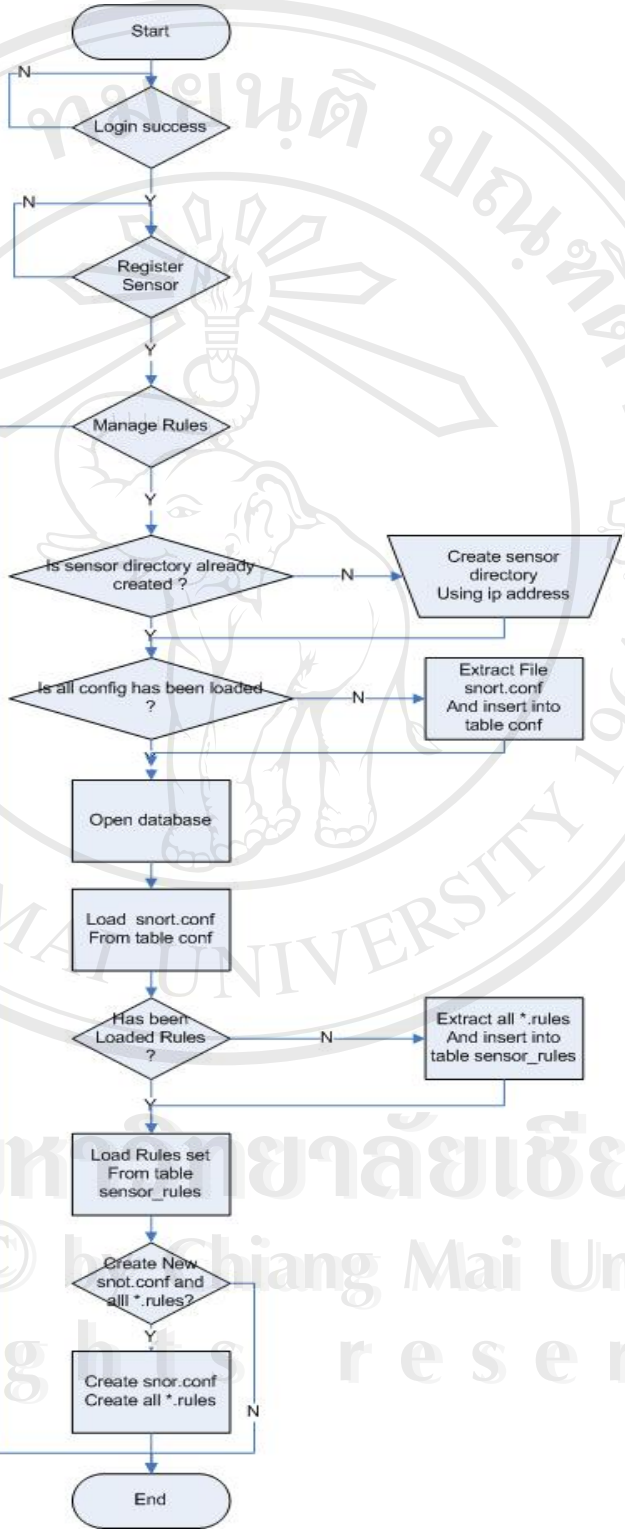
acid_ip_cache	เก็บข้อมูลไอพีแอดเดรสที่เกี่ยวข้องกับการละเมิดกฎ
conf	เก็บข้อมูลไฟล์ snort.conf
pre_back_orifice	เก็บข้อมูล preprocessor backOrifice
pre_flow	เก็บข้อมูล preprocessor flow
pre_flow_portscan	เก็บข้อมูล preprocessor flow_portscan
pre_frag2	เก็บข้อมูล preprocessor frag2
pre_performance_monitor	เก็บข้อมูล preprocessor performance monitor
pre_portscan	เก็บข้อมูล preprocessor portscan
pre_portscan2	เก็บข้อมูล preprocessor portscan2
pre_portscan_ingnorehost	เก็บข้อมูล preprocessor portscan ingnorehost
pre_rpc_decode	เก็บข้อมูล preprocessor RPC Decode
pre_sfportscan	เก็บข้อมูล preprocessor sfportscan
pre_stream4_reassembly	เก็บข้อมูล preprocessor stream4 reassembly
pre_stream4_stateful	เก็บข้อมูล preprocessor stream4 stateful
pre_telnet_decode	เก็บข้อมูล preprocessor telnet decode
rids_sensor	เก็บข้อมูลเซ็นเซอร์
sensor_rules	เก็บข้อมูลกฎของเซ็นเซอร์ทั้งหมด

โครงสร้างที่ปรากฏในตารางที่ที่ 3.13 นั้นได้พัฒนาเพิ่มเติมเพื่อจัดการกับกฎที่ใช้ในการตรวจสอบของ snort ที่มีอยู่ โดยรายละเอียดโครงสร้างทั้งหมดจะอยู่ในภาคผนวก ค

3.3.3.1 การพัฒนาระบบแสดงผลข้อมูลด้านสถิติ

สำหรับการแสดงผลข้อมูลการบุกรุกเครือข่ายนั้นได้เลือกใช้ระบบ ACID (Analysis Console Intrusion Detection System) ซึ่งเป็นระบบสำหรับรายงานผลการแจ้งเตือนจากโปรแกรมสนอร์ทที่ทำหน้าที่เป็นเซ็นเซอร์ตรวจจับ โดย ACID ได้รับการพัฒนาโดยภาษา PHP ซึ่งสามารถดาวน์โหลดได้ฟรีจาก http://sf.net ซึ่งเป็นระบบที่สามารถใช้งานได้ฟรีเช่นเดียวกับสนอร์ท ในการแสดงผลให้เป็นภาษาไทยผู้เขียนได้ทำการแก้ไขไฟล์ทุกไฟล์ที่นามสกุล .php และ .inc ให้เป็นภาษาไทยเพื่อให้ง่ายต่อการใช้งานสำหรับผู้ดูแลระบบ

3.3.3.2 การพัฒนาระบบจัดการเรื่องกฎการตรวจสอบ ได้วางโครงสร้างการทำงาน ดังนี้



รูปที่ 3.10 แสดงแผนภาพการไหลของข้อมูล (Flow-chart) ของการจัดการเรื่องกฎ

จากรูปที่ 3.10 นั้นจะเป็นขั้นตอนที่ใช้ในการจัดการเรื่องกฎการตรวจสอบ โดยใช้ภาษา PHP ในการพัฒนา โดยขั้นแรกสุดนั้นต้อง สร้างไดเรกทอรี่สำหรับเก็บข้อมูลเรื่องของกฎจากเครื่องเซ็นเซอร์ โดยมีไฟล์ที่เป็นหัวใจหลักคือ snort.conf จะต้องอยู่ในไดเรกทอรี่นี้ โดยชื่อของไดเรกทอรี่นั้นจะเป็นหมายเลขไอพีของเครื่องเซ็นเซอร์ ระบบจะทำการเปิดไฟล์ snort.conf และทุกไฟล์ที่นามสกุล .rules เพื่อทำการแปลงข้อมูลจากเท็กไฟล์ไปเป็นข้อมูลในฐานข้อมูล MySQL สำหรับการจัดการในอนาคต

3.4 สถานที่ที่ใช้ในการดำเนินการศึกษาและรวบรวมข้อมูล

มหาวิทยาลัยราชภัฏเชียงราย

3.5 ระยะเวลาในการศึกษา

การศึกษาครั้งนี้ใช้เวลาตั้งแต่เดือนพฤศจิกายน 2547 – พฤษภาคม 2548 รวมระยะเวลา 7 เดือน โดยทำการติดตั้งระบบตรวจจับการบุกรุกเครือข่ายในเดือน พฤษภาคม 2548