ภาษาจำเพาะโดเมนสำหรับการตรวจจับการบุกรุกเครือข่าย

นายคณิน โชติวรรักษ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

DOMAIN SPECIFIC LANGUAGE FOR NETWORK INTRUSION DETECTION

Mr. Kanin Chotvorrarak

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2013

หัวข้อวิทยานิพนธ์	ภาษาจำเพาะโดเมนสำหรับการตรวจจับการบุกรุกเครือข่าย
โดย	นายคณิน โชติวรรักษ์
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร. ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร. บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(ศาสตราจารย์ ดร. บุญเสริม กิจศิริกุล)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(รองศาสตราจารย์ ดร. ญาใจ ลิ่มปิยะกรณ์)

.....กรรมการภายนอกมหาวิทยาลัย

(อาจารย์ ดร. ภาสกร อภิรักษ์วรพินิต)

คณิน โชติวรรักษ์ : ภาษาจำเพาะโดเมนสำหรับการตรวจจับการบุกรุกเครือข่าย. (DOMAIN SPECIFIC LANGUAGE FOR NETWORK INTRUSION DETECTION) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร. ญาใจ ลิ่มปิยะกรณ์, 84 หน้า.

งานวิจัยนี้ได้นำเสนอระบบตรวจจับการบุกรุกเครือข่าย ซึ่งจัดอยู่ในประเภทวิธีการตรวจจับแบบอิงลายเซ็นต์ ภาษาจำเพาะโดเมนอีสดีเอสแอลได้ถูกพัฒนาขึ้นเพื่อใช้ประกาศลายเซ็นต์การบุกรุก วากยสัมพันธ์ของกฎอีสดีเอสแอลได้ถูกกำหนดขึ้นบนพื้นฐานโครงสร้างโปรโตคอลทีซีพี/ไอพี และสัญญาณของการบุกรุกถูกกำหนดการเขียนให้อยู่ในรูปแบบของคุณสมบัติและค่าที่สามารถข้ามแพกเก็ตหรือชั้นโปรโตคอลทีซีพี/ไอพีได้ งานวิจัยได้พัฒนาระบบต้นแบบตรวจจับการบุกรุก ประกอบด้วยองค์ประกอบหลัก 3 ส่วน ได้แก่ 1) ตัวแจงส่วนอีสดีเอสแอล 2) ตัวเฝ้าระวังการจราจรเครือข่าย และ 3) ตัวตรวจจับการบุกรุกเครือข่าย ตัวแจงส่วนอีสดีเอสแอลสนับสนุนการวิเคราะห์คำในเงื่อนไขการบุกรุกของกฎในสคริปต์เพื่อเปลี่ยนเป็นเซตโครงสร้างของกฎสำหรับจับคู่กับแพกเก็ตการบุกรุกเครือข่ายที่เหมือนกัน ตัวเฝ้าระวังการจราจรเครือข่ายจะรับผิดชอบการดักจับแพกเก็ตเครือข่ายและเก็บไว้ในบัฟเฟอร์เพื่อตรวจสอบในขั้นตอนตรวจจับการบุกรุกเครือข่าย ซึ่งประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรมเพื่อค้นหาสถานะมุ่งร้ายบนการจราจรเครือข่าย ในงานวิจัยได้ทำการทดลองเบื้องต้นเพื่อศึกษาสมรรถนะของแนวทางที่นำเสนอ จากผลการทดลองพบว่าการประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรมเพื่อค้นหาสัญญาณการละเมิดความมั่นคงด้วยกฎเชิงประกาศเป็นแนวทางที่มีประสิทธิภาพและมีความเป็นไปได้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5570966621 : MAJOR SOFTWARE ENGINEERING
KEYWORDS: DOMAIN SPECIFIC LANGUAGE / INTRUSION DETECTION SYSTEM / GENETIC ALGORITHM / NETWORK SECURITY

KANIN CHOTVORRARAK: DOMAIN SPECIFIC LANGUAGE FOR NETWORK INTRUSION DETECTION. ADVISOR: ASSOC. PROF. DR. YACHAI LIMPIYAKORN, 84 pp.

This research presents a network intrusion detection system, which is categorized as a type of signature-based detection method. A domain specific language, called isDSL, is developed as a means of declaring intrusion signatures. The isDSL rule syntax is defined based on the structure of TCP/ IP stack, and the sign of attack is prescribed as a combination of properties and values that could span across the packets or TCP/IP layers. The prototype of intrusion detection system has been implemented. It consists of three major components: 1) isDSL parser, 2) Network traffic monitor, and 3) Network intrusion detector. The isDSL parser supports the parsing of the intrusion conditions prescribed in a rule script into a set of rule structures used for matching with the network intrusion packets. Traffic monitor is the engine responsible for capturing the network packets and storing them in the buffer for further inspection. The Network intrusion detector applies the genetic algorithm for searching malicious states on network traffics. Preliminary experiments were conducted to study the performance of the presented approach. The findings reported that the application of genetic algorithm for searching the signs of security breaches against declarative rules would be efficient and promising.

Department: Computer Engineering Student's Signature .....
Field of Study: Software Engineering Advisor's Signature .....
Academic Year: 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความอนุเคราะห์อย่างดียิ่งของรองศาสตราจารย์ ดร. ญาใจ ลิ่มปิยะกรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้สละเวลาให้ความรู้ ให้คำปรึกษา ตรวจสอบ ให้คำแนะนำแนวทางการวิจัย และสนับสนุน จนทำให้การวิจัยในครั้งนี้สำเร็จออกมาด้วยดี ข้าพเจ้าจึงขอกราบระลึกถึงพระคุณของรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ไว้ ณ ที่นี้

ขอขอบพระคุณ ศาสตราจารย์ ดร.บุญเสริม กิจศิริกุล และ ดร.ภาสกร อภิรักษ์วรพินิต กรรมการสอบวิทยานิพนธ์ ที่กรุณาเสียสละเวลา ให้คำแนะนำ ตรวจสอบ และแก้ไขวิทยานิพนธ์ฉบับนี้

ท้ายที่สุด ผู้เสนอวิทยานิพนธ์ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ และครอบครัว สำหรับกำลังใจที่มีค่ายิ่ง รวมถึงขอขอบพระคุณผู้บังคับบัญชาในสายงาน เพื่อนร่วมงาน และมิตรสหาย ที่คอยติดตามให้กำลังใจ ให้การสนับสนุนและความช่วยเหลือในด้านต่างๆ และท่านอื่นๆ ที่มิได้กล่าวชื่อไว้ ณ ที่นี้ที่มีส่วนช่วยให้วิทยานิพนธ์ของข้าพเจ้าสำเร็จไปได้ด้วยดี

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การขยายตัวของการใช้งานระบบเครือข่ายคอมพิวเตอร์ที่เพิ่มสูงขึ้นในปัจจุบันทำให้เกิดภัยคุกคาม (Threat) หรือการโจมตีทางเครือข่ายเพิ่มสูงขึ้นอย่างมาก โดยการรักษาความมั่นคงปลอดภัยของเครือข่าย (Network security) จะมีเครื่องมือที่ใช้ในการตรวจจับการบุกรุก หรือความผิดปกติที่เรียกว่า ระบบตรวจจับการบุกรุก (Intrusion Detection System- IDS) และระบบป้องกันการบุกรุก (Intrusion Prevention System) ซึ่งจะนำสัญญาณจราจร(Traffic) หรือข้อมูลแพกเก็ต(Packet)ที่ส่งไปมาในเครือข่ายมาประมวลผลเพื่อตรวจหาความผิดปกติหรือการบุกรุก โดยเมื่อระบบตรวจพบความผิดปกติ (Anomaly) ที่เกิดขึ้นกับเครือข่าย ก็จะทำการแจ้งเตือนผู้ใช้ หรือระงับความผิดปกตินั้น โดยการแบ่งประเภทของระบบตรวจจับการบุกรุกตามวิธีการตรวจจับสามารถแบ่งได้ 2 ประเภท คือ 1) ระบบที่ตรวจจับจากการใช้งานผิดปกติ (Misuse detection) ที่ใช้หลักการตรวจหารูปแบบการบุกรุกด้วยการวิเคราะห์พฤติกรรม หรือเหตุการณ์ที่เข้ามาในเครือข่าย มาใช้เพื่อเปรียบเทียบกับข้อมูลการบุกรุกหรือกฎซึ่งจัดเก็บไว้ในฐานข้อมูลระบบ ซึ่งหากเปรียบเทียบตรงกันแสดงว่าพฤติกรรมหรือเหตุการณ์นั้นเป็นการบุกรุกการตรวจจับแบบนี้จะมีข้อดีคือสามารถโปรแกรม หรือสร้างกฎที่ใช้ในการตรวจจับการบุกรุกได้และมีความแม่นยำสูง แต่มีข้อเสียคือไม่สามารถรองรับการโจมตีรูปแบบใหม่ที่ไม่เคยบันทึกหรือเก็บไว้ในฐานข้อมูลและใช้ระยะเวลาค่อนข้างมากในการเปรียบเทียบกฎที่ใช้ในประมวลผลและ 2) ระบบที่ตรวจจับจากข้อมูลการใช้งานปกติเพื่อตรวจหาความผิดปกติ (Anomaly detection) ซึ่งจะตรงข้ามกับแบบแรก คือจะเก็บข้อมูลการใช้งานที่ปกติ ในการตรวจหาความผิดปกติที่เกิดขึ้น ซึ่งส่วนใหญ่จะใช้ความรู้เกี่ยวกับการเรียนรู้ของเครื่อง (Machine Learning) เข้ามาช่วยในการเรียนรู้พฤติกรรม หรือการใช้งานที่ปกติ เพื่อใช้ในการตรวจจับความผิดปกติ หรือการบุกรุกที่เกิดขึ้นได้ วิธีนี้มีข้อดีคือสามารถรองรับการโจมตีรูปแบบใหม่ๆได้ และมีการประมวลผลที่รวดเร็วกว่าแบบแรก แต่มีข้อเสียคือต้องอาศัยข้อมูลที่ใช้ในการเรียนรู้สูง และหากเป็นรูปแบบการใช้งานที่มีความแปลกใหม่ ที่ไม่ใช่รูปแบบการบุกรุก อาจก่อให้เกิดการตรวจจับที่ผิดพลาด (Fault Positive Detection) โดยนอกจากการใช้ข้อมูลเครือข่ายมาใช้ในการวิเคราะห์เพื่อการการบุกรุกแล้ว ระบบตรวจจับการบุกรุกบางประเภทยังมีการใช้ข้อมูลจากระบบอย่างล็อกไฟล์ (Log file) ที่มีร่องรอยการใช้งานคำสั่งของผู้ใช้งาน และข้อมูลสมรรถนะเครือข่าย (Network performance) หรือสมรรถนะระบบ (System Performance) มาใช้ในการวิเคราะห์เพื่อตรวจจับการบุกรุกอีกด้วย

ภาษาจำเพาะโดเมน หรือ ดีเอสแอล (Domain Specific Language - DSL) คือ ภาษาขั้นสูงที่ถูกกำหนดรูปแบบขึ้นโดยมีโครงสร้างของตัวภาษาซับซ้อนน้อยที่สุด การออกแบบและใช้งานจะมีจุดประสงค์เพื่อแก้ไขปัญหาในโดเมนปัญหา (Problem domain) ที่สนใจ โดยคำนึงถึงสภาพแวดล้อมที่ภาษาถูกนำไปใช้งาน [1]

จากการศึกษาพบว่าระบบตรวจจับการบุกรุกเครือข่ายบางระบบมีการสร้างภาษาที่ใช้ในการเขียนเงื่อนไขการตรวจจับเช่นระบบ Snort [2] หรือระบบ Bro [3] ที่มีภาษาที่ใช้ในการเขียนกฎใน

การตรวจจับการบุกรุกเครือข่าย แต่ภาษาเหล่านั้นเป็นภาษาเชิงสคริปต์ที่เขียนเพื่ออธิบายเงื่อนไขที่ใช้ในการกรองแพกเก็ตที่สนใจ เพื่อแจ้งเตือนผู้ใช้งานให้ทราบถึงการบุกรุกเครือข่ายของแพกเก็ตที่ตรงกับเงื่อนไขโดยใช้หลักการจัดคู่รูปแบบ และนิพจน์ปรกติที่ตรงกับสคริปต์ที่เขียนขึ้นซึ่งอธิบายลักษณะของพฤติกรรมการบุกรุกได้จำกัดอยู่บนแพกเก็ตเดียว

งานวิจัยนี้จึงได้เสนอแนวทางการประยุกต์ใช้ภาษาจำเพาะโดเมนมาใช้ในการเขียนกฎสำหรับตรวจจับรูปแบบพฤติกรรมการบุกรุกเครือข่าย เพื่อให้ผู้เชี่ยวชาญสามารถสร้างกฎในการตรวจจับการบุกรุกแบบใหม่ได้ง่ายโดยอาศัยข้อมูลในแพกเก็ตโพรโทคอลทีซีพีไอพี (TCP/IP Protocol) โดยภาษาจำเพาะโดเมนที่สร้างขึ้นจะสามารถอธิบายรูปแบบพฤติกรรมการบุกรุกเครือข่ายที่เกิดจากกลุ่มของแพกเก็ต ซึ่งสามารถเขียนเงื่อนไขในเรื่องเวลา และลำดับของแพกเก็ตมาใช้ในการตรวจหาการบุกรุกพฤติกรรมการบุกรุกได้

1.2 วัตถุประสงค์ของการวิจัย

เพื่อสร้างภาษาจำเพาะโดเมนสำหรับการเขียนกฎ หรือรูปแบบพฤติกรรมการบุกรุก ซึ่งเป็นการตรวจจับการบุกรุกจากการใช้งานผิดปกติ เพื่อให้ผู้เชี่ยวชาญสามารถสร้างกฎในการตรวจจับการบุกรุกแบบใหม่ได้ โดยอาศัยข้อมูลในแพกเก็ตในโพรโทคอลทีซีพีไอพี โดยภาษาจำเพาะโดเมนที่สร้างขึ้นจะสามารถอธิบายรูปแบบพฤติกรรมการบุกรุกเครือข่ายที่เกิดจากกลุ่มของแพกเก็ต ซึ่งสามารถเขียนเงื่อนไขในเรื่องเวลา และลำดับของแพกเก็ตมาใช้ในการตรวจหาการบุกรุกเครือข่าย

1.3 ขอบเขตของการวิจัย

1. พัฒนาภาษาจำเพาะโดเมนและตัวแจงส่วนโดยมีความสามารถต่อไปนี้เป็นอย่างน้อย
   - การออกแบบภาษาจำเพาะโดเมนจะใช้ข้อมูลจากโพรโทคอลทีซีพีไอพี
   - ตัวแจงส่วนสามารถแจ้งเตือนได้เมื่อพบรูปแบบที่ผิดพลาด
2. พัฒนาส่วนเชื่อมต่อผู้ใช้เพื่อแก้ไขภาษาจำเพาะโดเมน โดยมีความสามารถต่อไปนี้เป็นอย่างน้อย
   - สามารถแก้ไขไฟล์ภาษาจำเพาะโดเมนได้
   - สามารถแจ้งเตือนได้เมื่อพบรูปแบบที่ผิดพลาด
3. ระบบตรวจจับการบุกรุกการบุกรุกจะใช้ข้อมูลจากแพกเก็ตที่ดักจับด้วยโปรแกรมดักจับแพกเก็ตมาใช้เพื่อประมวลผล

1.4 วิธีดำเนินงานวิจัย

1. ศึกษาและทำความเข้าใจทฤษฎีและงานวิจัยที่เกี่ยวข้อง ได้แก่ ภาษาจำเพาะโดเมน โพรโทคอลทีซีพี/ไอพี ระบบตรวจจับการบุกรุก รูปแบบการโจมตีและประเภทบุกรุกเครือข่าย
2. วิเคราะห์และกำหนดภาพรวมของงานวิจัย
3. เลือกเครื่องมือที่เหมาะสมในการสร้างระบบสนับสนุนเพื่อช่วยในการสร้างตัวแจงส่วนเพื่อการออกแบบภาษาจะเพาะโดเมน
4. พัฒนาตัวแจงส่วนและส่วนเชื่อมต่อผู้ใช้งานเพื่อแก้ไขภาษาจำเพาะโดเมน

5. ทดสอบการทำงานของภาษาจำเพาะโดแมนในการตรวจจับการบุกรุก และประเมินผลงานวิจัย
6. ตีพิมพ์ผลงานทางวิชาการ
7. สรุปผลการวิจัยและข้อเสนอแนะ และจัดทำวิทยานิพนธ์

1.5 ข้อตกลงเบื้องต้น

1. ภาษาจำเพาะโดเมนที่ใช้ในการเขียนกฎการบุกรุกจะอยู่บนพื้นฐานของชุดโพรโทคอลทีซีพี/ไอพี โดยเลือกมาเพียงบางโพรโทคอลเท่านั้น
2. ประเมินผลงานวิจัยจากการทดลองใช้ภาษาจำเพาะโดเมนที่สร้างขึ้นในการตรวจจับการบุกรุกเครือข่ายที่เกิดจากกลุ่มของแพกเก็ตที่มีคุณลักษณะตรงกับกฎที่เขียนขึ้นได้

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ภาษาจำเพาะโดเมน และตัวแจงส่วนที่ใช้ในการเขียนกฎที่ใช้ในการตรวจจับพฤติกรรมการบุกรุกเครือข่ายบนโพรโทคอลทีซีพี/ไอพี
2. ได้ระบบตรวจจับการบุกรุกเครือข่ายที่ประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรมในการค้นหารูปแบบพฤติกรรมการบุกรุกเครือข่ายที่ทำงานร่วมกับภาษาจำเพาะโดเมนที่สร้างขึ้น

1.7 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความวิชาการเรื่อง “isDSL-A Domain Specific Language for Intrusion Signature Declaration”, (ASEA), 2013 และบทความวิชาการเรื่อง “Domain Specific Language for Detecting Intrusion Signature with Genetic Search”, International Journal of Security and Its Application (IJSIA), Vol. 8 No. 2 March 2014

1.8 ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

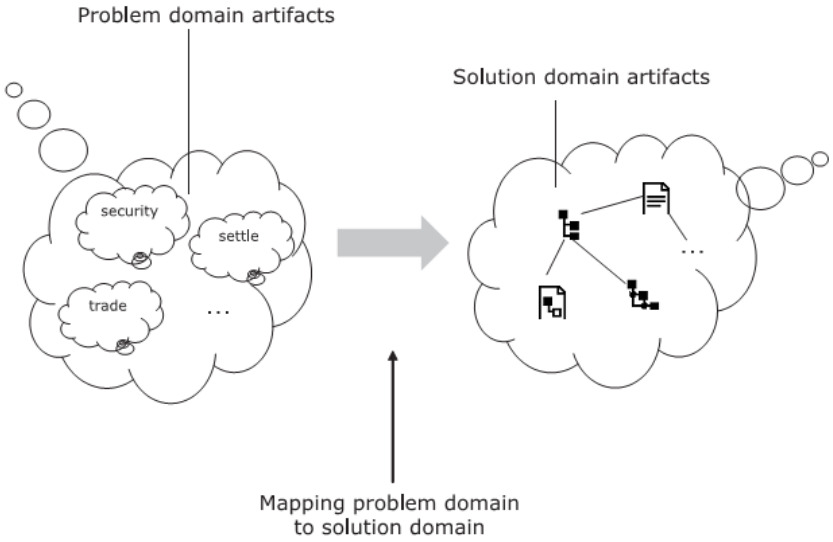
วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 6 บทดังต่อไปนี้ บทที่ 1 เป็นบทนำกล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ประโยชน์ที่คาดว่าจะได้รับและผลงานตีพิมพ์ บทที่ 2 กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้องบทที่ 3 กล่าวถึงวิธีดำเนินการวิจัย บทที่ 4 กล่าวถึงการออกแบบและพัฒนาระบบตามแนวทางการวิจัยที่นำเสนอ บทที่ 5 กล่าวถึงวิธีการประเมินและวัดผลการทดลองและบทที่ 6 สรุปผลการวิจัย ข้อเสนอแนะ และแนวทางสำหรับการวิจัยต่อในอนาคต

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

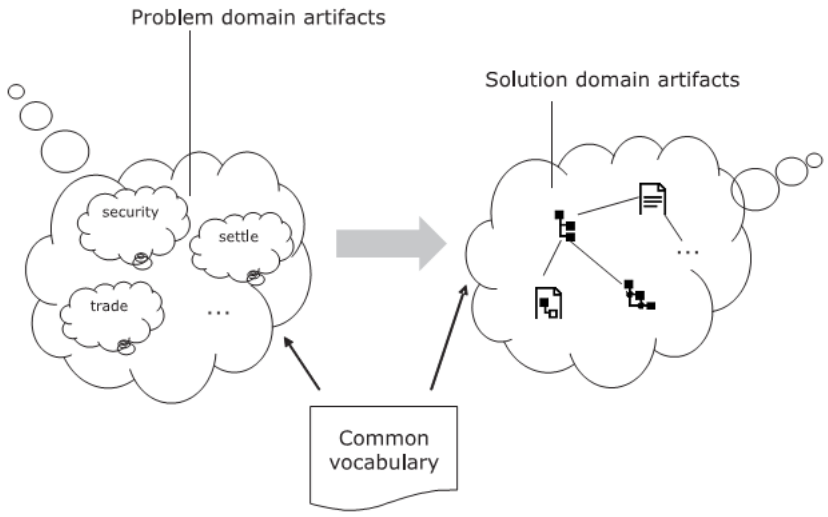
2.1.1 ภาษาจำเพาะโดเมน หรือ ดีเอสแอล (Domain Specific Language - DSL) [1]

กระบวนการในการแก้ปัญหาต่างๆ จะมีกระบวนการในการทำความเข้าใจปัญหา เพื่อที่จะสามารถหาวิธีการ หรือเครื่องมือที่จะแก้ปัญหานั้นๆ เราเรียกกระบวนการนี้ว่า การสร้างแบบจำลองโดเมน (Domain modeling) หรือที่เรียกกันทั่วไปว่า การวิเคราะห์โดเมน (Domain analysis) ซึ่งสามารถแบ่งได้เป็น 2 ส่วนคือ โดเมนปัญหา (Problem domain) และ โดเมนคำตอบ (Solution domain) โดยโดเมนปัญหาจะประกอบด้วย สิ่งที่เกี่ยวข้องกับปัญหานั้นๆ ผู้ที่เกี่ยวข้อง กระบวนการ รวมถึงข้อจำกัดของปัญหา และโดเมนคำตอบประกอบด้วย เทคนิค วิธีการ เครื่องมือ รวมถึงผู้พัฒนาเครื่องมือที่จะสามารถแก้ไขปัญหาได้ ซึ่งการศึกษาวิจัยได้มีความพยายามที่จะเชื่อมโยงโดเมนปัญหาและโดเมนคำตอบเข้าด้วยกันดังภาพที่ 1



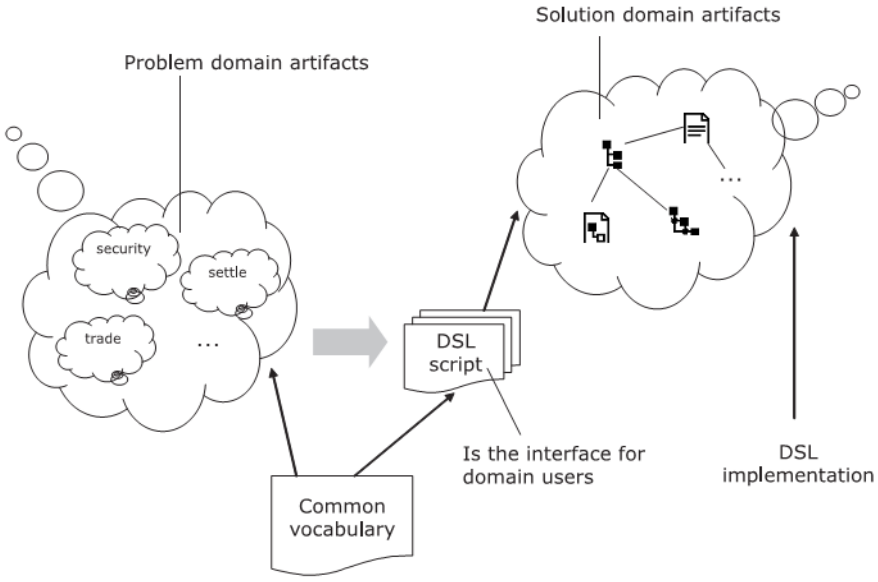
ภาพที่ 1 การเชื่อมต่อของโดเมนปัญหาและโดเมนคำตอบ [1]

ขั้นตอนในการเชื่อมโยงปัญหาจากโดเมนปัญหาไปยังเทคนิค หรือวิธีการในการแก้ปัญหาในโดเมนคำตอบนั้น จะต้องมีการสร้างคำศัพท์ร่วม (Common vocabulary) ที่ใช้ในการอธิบายสิ่งที่สนใจ (Entity) ที่มีความหมายเดียวกันระหว่างทั้งสองโดเมน เช่นคำสั่งซื้อหุ้น (Stock trade) ในโดเมนปัญหาของผู้ใช้ที่ต้องการจะซื้อขายหุ้น และฟังก์ชันการซื้อขายหุ้นในโดเมนคำตอบที่อยู่ในเครื่องมือที่ใช้ซื้อขายหุ้น ดังภาพที่ 2



ภาพที่ 2 คำศัพท์ร่วมของโดเมนปัญหาและโดเมนคำตอบ [1]

จากความพยายามที่จะเชื่อมโยงโดเมนปัญหา และโดเมนคำตอบเข้าด้วยกันนั้นทำให้เกิดภาษาจำเพาะโดเมนขึ้นเพื่อที่จะเป็นสิ่งที่ใช้ในการเชื่อมโยง สิ่งต่างๆจากโดเมนปัญหา ไปยังโดเมนคำตอบได้อย่างราบรื่น และถูกต้องตรงกัน



ภาพที่ 3 องค์ประกอบของภาษาจำเพาะโดเมน [1]

จากภาพที่ 3 ภาษาจำเพาะโดเมนประกอบด้วย 2 ส่วนหลักๆคือ ส่วนที่เป็นสคริปต์ของภาษาจำเพาะโดเมน (DSL script) ซึ่งเป็นช่องทางการเชื่อมต่อกับผู้ใช้ในการเขียน อ่านภาษาจำเพาะโดเมน และส่วนที่ทำให้เกิดผลของภาษาจำเพาะโดเมน (DSL implementation)

องค์ประกอบสำคัญของภาษาจำเพาะโดเมนประกอบด้วย

1. ภาษาการโปรแกรมคอมพิวเตอร์ (Computer Programming Language) : การทำงานของภาษาจำเพาะโดเมนจะต้องสามารถให้มนุษย์ทำความเข้าใจได้ง่าย และภาษาจำเพาะโดเมนจะต้องสามารถสั่งให้คอมพิวเตอร์ทำงานได้
2. ธรรมชาติของภาษา (Language Nature) : ลักษณะของภาษาจำเพาะโดเมนจะต้องมีความสอดคล้องกับโดเมนของปัญหาทั้งคำศัพท์ที่ใช้ และกระบวนการทำงานของภาษาเพื่อให้ผู้เชี่ยวชาญในโดเมนนั้นสามารถเข้าใจ และใช้งานได้
3. การสื่อความหมายที่ลึกซึ้งอย่างจำกัด (Limited expressiveness): ภาษาจำเพาะโดเมนแตกต่างจากภาษาโปรแกรมมิ่งในเรื่องของการรองรับความหลากหลายของโครงสร้างและความสามารถต่างๆของภาษาที่สามารถประยุกต์กับปัญหาต่างๆได้ แต่ภาษาจะจำกัดอยู่ในโดเมนปัญหาหนึ่งๆ ซึ่งไม่สามารถทดแทนทั้งระบบได้ แต่ใช้ในการแก้ปัญหากับบางส่วนของระบบเท่านั้น
4. โดเมนโฟกัส (Domain Focus): ภาษาที่ถูกจำกัดอยู่ภายในโดเมนนั้นจะมีประโยชน์ ถ้าสามารถระบุโดเมนที่สนใจได้อย่างชัดเจนจึงต้องระบุให้ชัดเจนถึงโดเมนที่ภาษานั้นๆดำรงอยู่

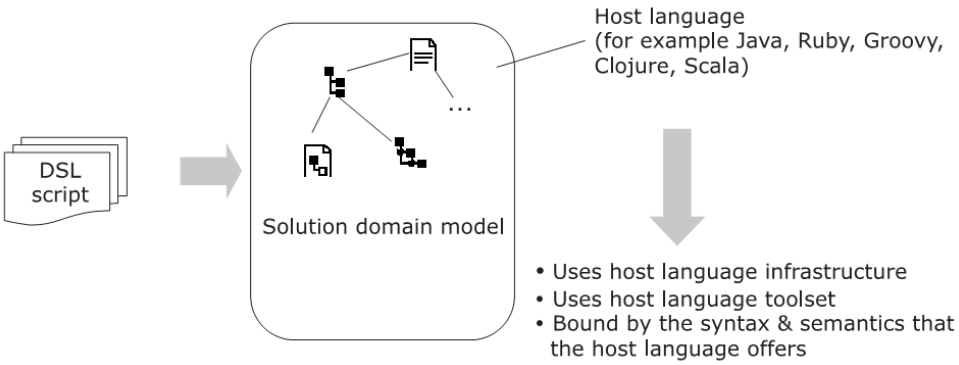
ในปัจจุบันภาษาจำเพาะโดเมนได้รับความนิยมอย่างมาก ซึ่งตัวภาษาจะถูกสร้างขึ้นโดยผู้เชี่ยวชาญในโดเมนปัญหานั้นๆ ทำให้ภาษาจำเพาะโดเมนสามารถเข้าใจ และแก้ไขได้ง่าย โดยไม่จำเป็นต้องมีความรู้ความเข้าใจด้านโปรแกรมมิง ส่งผลให้การติดต่อสื่อสารระหว่างผู้ที่อยู่ในโดเมนปัญหา และผู้ออกพัฒนาระบบเป็นไปอย่างราบรื่น และเข้าใจถูกต้องตรงกัน ในตารางที่ 1 แสดงตัวอย่างของการนำภาษาจำเพาะโดเมนไปใช้ในการแก้ปัญหาต่างๆ

ตารางที่ 1 ตัวอย่างภาษาจำเพาะโดเมนที่นำไปใช้ในการแก้ปัญหา

ภาษาจำเพาะโดเมน	**คำอธิบาย**
SQL	ภาษาที่ใช้ในการร้องขอ(Query) ฐานข้อมูลเชิงสัมพันธ์ (Relational database) และการจัดการข้อมูล
HTML	ภาษามาร์กอัปสำหรับพัฒนาเวป
CSS	ภาษารายละเอียดแผ่นลักษณะ (Style sheet)
YACC, Bison, ANTLR	ภาษาในการสร้างตัวแจงส่วน (Parser)
RSpec, Cucumber	ภาษาในการชี้นำการทดสอบด้วยพฤติกรรม (Behavior-driven testing) ในภาษารูบี (Ruby)
Ant, Rake, Make	ภาษาที่ใช้ในการสร้างระบบซอฟต์แวร์

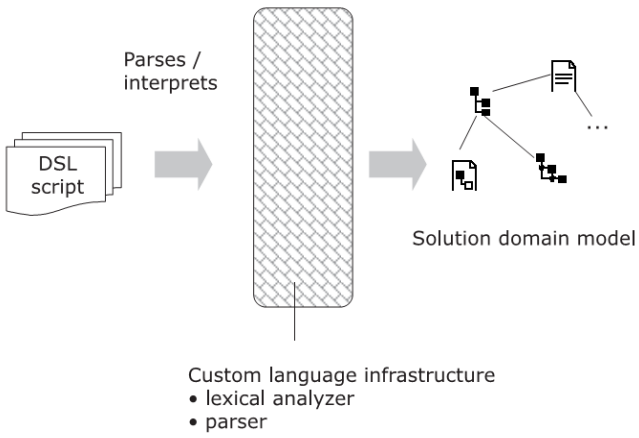
ภาษาจำเพาะโดเมนสามารถแบ่งออกเป็น 3 ประเภทคือ

1. ภาษาจำเพาะโดเมนภายใน (Internal DSLs) : คือภาษาจำเพาะโดเมนที่ใช้โครงสร้างพื้นฐานของภาษาโปรแกรมมิงที่มีอยู่เดิมที่เรียกว่าภาษาโฮสต์ (Host language) ในการสร้างภาษาจำเพาะโดเมนเพื่อสื่อความหมายแทนภาษาโปรแกรมมิงเดิมซึ่งการทำงานจะอิงกับระบบเดิมของภาษาโฮสต์ ตัวอย่างของภาษาจำเพาะโดเมนภายในที่เป็นที่รู้จักคือ ภาษาเรียล (Rail) ที่ถูกพัฒนาขึ้นบนภาษารูบี (Ruby) เวลาที่ผู้ใช้เขียนภาษาเรียล จะเปรียบเสมือนกำลังเขียนภาษารูบี โดยทั่วไปภาษาจำเพาะโดเมนภายในจะถูกพัฒนาขึ้นเพื่อเป็นไลบรารี่ของภาษโฮสเพื่ออำนวยความสะดวกในการใช้งานในโดเมนหนึ่งๆดังภาพที่ 4 แสดงองค์ประกอบของภาษาจำเพาะโดเมนภายใน



ภาพที่ 4 องค์ประกอบของภาษาจำเพาะโดเมนภายใน [1]

2. ภาษาจำเพาะโดเมนภายนอก (External DSLs) : คือภาษาจำเพาะโดเมนที่พัฒนาขึ้นเพื่อสนับสนุนการทำงานของระบบที่มีอยู่ก่อน โดยอาจมีพื้นฐานโครงสร้างที่ไม่อิงกับภาษาที่มีอยู่ก่อน หรืออาจถูกพัฒนาขึ้นจากภาษาอื่นเช่น ภาษาเอ็กเอ็มแอล โดยจะมีโครงสร้างพื้นฐานที่ถูกพัฒนาแยกตัวออกมา ในการสร้างไวยกรณ์ และวากยสัมพันธ์ของภาษาจำเพาะโดเมน เช่น ตัววิเคราะห์คำศัพท์ (Lexical analysis) ตัวแจงส่วน (Parser) ตัวแปลภาษา(Interpreter) การคอมไพล์ (Compilation) เป็นต้น ดังภาพที่ 5 แสดงองค์ประกอบของภาษาจำเพาะโดเมนภายนอก



ภาพที่ 5 องค์ประกอบของภาษาจำเพาะโดเมนภายนอก [1]

3. ภาษาจำเพาะโดเมนที่ไม่มีต้นฉบับดังเดิม (Non-textual DSLs) : คือภาษาจำเพาะโดเมนที่สร้างขึ้นบนสิ่งแวดล้อม (Environment)ใหม่ ที่มีจุดประสงค์เพื่อการแก้ปัญหาสำหรับโดเมนหนึ่งโดยเฉพาะ โดยมีการสร้างสิ่งแวดล้อมของเพื่อการพัฒนา (Integrated Development Environment, IDE) เพื่ออำนวยความสะดวกในการแก้ไข และสนับสนุนการพัฒนาของโดเมนที่สนใจ

ข้อดีของการพัฒนาภาษาจำเพาะโดเมนมีดังนี้

1. การพัฒนาที่อาศัยภาษาจำเพาะโดเมนสามารถขยายตัวได้ง่าย โดยการปรับเปลี่ยน หรือเพิ่มเติมความสามารถของภาษา
2. ผู้เชี่ยวชาญในโดเมนมีความเข้าใจในภาษาจำเพาะโดเมนทำให้ผลิตผล (Productivity) สูงขึ้น
3. ภาษาจำเพาะโดเมนถูกออกแบบให้สนใจเฉพาะรูปแบบนามธรรมระดับสูง (High level of abstraction) ทำให้เพิ่มความสามารถในการเรียนรู้ภาษาเดิมของผู้ใช้

ข้อเสียของการพัฒนาภาษาจำเพาะโดเมนมีดังนี้

1. การพัฒนาภาษาจำเพาะโดเมนทำได้ยาก และใช้ทรัพยากรสูง ซึ่งทำให้งบประมาณในการพัฒนาอาจไม่คุ้มต่อทรัพยากรที่ต้องเสีย
2. การพัฒนาภาษาจำเพาะโดเมนจำเป็นต้องใช้ความรู้ในโดเมนที่จะพัฒนา รวมถึงความรู้ในด้านวิศวกรรมโดเมน และการสร้างภาษาจำเพาะโดเมน
3. การทำงานของภาษาจำเพาะโดเมนอาจนำไปสู่ข้อจำกัดเรื่องประสิทธิภาพการทำงานของระบบ

2.1.2 ระบบตรวจจับการบุกรุก (Intrusion Detection System, IDS) [4-6]

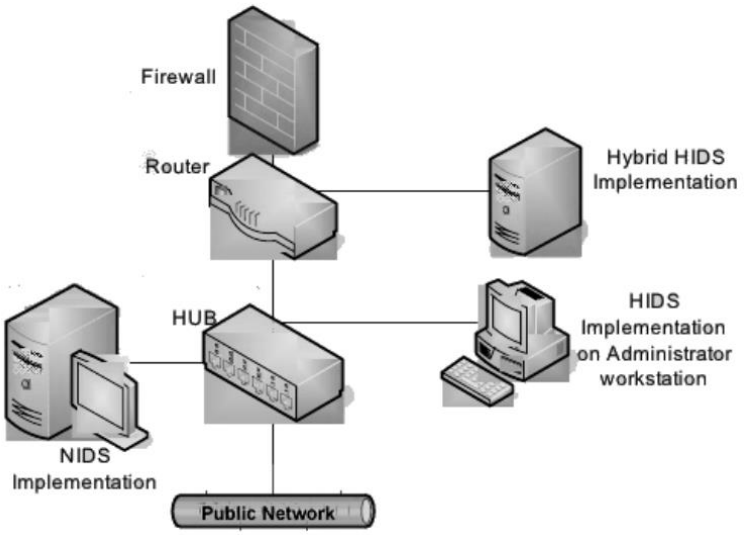
การตรวจจับการบุกรุก (Intrusion detection) คือกระบวนการในการเฝ้าดู และวิเคราะห์ข้อมูลโดยมีจุดประสงค์ที่จัดค้นหาข้อมูลที่ประสงค์ร้าย (malicious information) ในช่วง ค.ศ.1987 หลังจากอินเทอร์เนต เริ่มมีการแพร่หลายซึ่งระบบคอมพิวเตอร์ และเครือข่ายเป็นช่องทางที่สำคัญในการติดต่อสื่อสารในการทำธุรกิจส่งผลให้เกิดภัยคุกคามประเภทต่างๆเพิ่มมากขึ้น ตัวอย่างเช่นภัยคุกคามที่เกิดขึ้นกับข้อมูลที่สำคัญ อย่างข้อมูลการเงิน หรือระบบบริการ (system service) จนส่งผลให้เกิดความเสียหายต่อความถูกต้องของข้อมูล ความสามารถในการให้บริการของระบบได้ หรือความน่าเชื่อถือขององค์กรได้ ทำให้ Dorothy Denning เสนอ "แบบจำลองการตรวจจับการบุกรุก" (intrusion detection model) ซึ่งก่อให้เกิดแรงกระตุ้นในการวิจัยในการพัฒนาระบบตรวจจับการบุกรุกเพิ่มมากขึ้น โดยระบบตรวจจับการบุกรุกจะต้องเป็นระบบที่สามารถแจ้งเตือนผู้ใช้ หรือผู้ดูแลให้ทราบถึงการภัยคุกคามที่เกิดขึ้นกับระบบคอมพิวเตอร์ หรือเครือข่ายที่ดูแลได้ ซึ่งปัจจุบันมีทั้งระบบการตรวจจับการบุกรุกเชิงพาณิชย์ (commercial intrusion detection) และแบบเปิดต้นฉบับ (open source) เป็นที่รู้จักมากมาย เช่น Real secure, Trip wire, Snort, Shadow และ STAT เป็นต้น

ประเภทของระบบตรวจจับการบุกรุกที่แบ่งตามแพลตฟอร์มสามารถแบ่งได้เป็น 2 ประเภทใหญ่ๆคือ

1. ระบบตรวจจับการบุกรุกประเภทโฮสต์เบส (Host-based IDS) คือโปรแกรมหรือ ระบบตรวจจับการบุกรุกที่ถูกติดตั้งลงบนระบบคอทพิวเตอร์เพื่อตรวจจับข้อมูลนำเข้าและออกจากคอมพิวเตอร์ และตรวจจับการทำงานของกระบวนการ (Process) และไฟล์ของระบบ (System file) เพื่อแจ้งเตือน และบันทึกลงล็อกไฟล์ (Log file) ในกรณีที่เป็นการบุกรุก หรือเป็นภัยคุกคาม
2. ระบบตรวจจับการบุกรุกเครือข่าย (Network-based IDS) คือระบบที่ตรวจจับการจราจรภายในเครือข่ายว่ามีการใช้งานเครือข่ายที่เป็นการบุกรุกหรือประสงค์ร้ายหรือไม่ โดยระบบตรวจจับการบุกรุกประเภทนี้จะตรวจแพกเก็ตเครือข่ายที่ส่งไปมามาตรวจสอบเพื่อค้นหารูปแบบที่น่าสงสัย เพื่อแจ้งเตือนหรือระงับการบุกรุกนั้น

ความแตกต่างระหว่างระบบตรวจจับการบุกรุกประเภทโฮสต์เบส และระบบตรวจจับการบุกรุกเครือข่ายนั้นจะต่างกันที่ระบบตรวจจับการบุกรุกประเภทโฮสต์เบสจะเน้นไปที่การปกป้องระบบที่ถูกระบบตรวจจับการบุกรุกประเภทนี้ติดตั้งอยู่เครื่องใดเครื่อหนึ่งเท่านั้น ในขณะที่ระบบตรวจจับการบุกรุกเครือข่ายจะตรวจจับ และแจ้งเตือนปัญหา หรือการบุกรุกของสภาพแวดล้อมของเครือข่ายที่ติดตั้ง

นอกจากนี้ยังมีการออกแบบระบบตรวจจับการบุกรุกที่ผนวกเอาทั้ง 2 ประเภทที่แบ่งตามแพลตฟอร์ม คือ ระบบตรวจจับการบุกรุกประเภทโฮสต์เบส และระบบตรวจจับการบุกรุกเครือข่ายเข้าด้วยกันเรียกว่า ระบบตรวจจับการบุกรุกแบบลูกผสม (Hybrid intrusion detection) เพื่อให้สามารถนำข้อดีของทั้ง 2 รูปแบบไว้ด้วยกัน โดยระบบตรวจจับการบุกรุกแบบลูกผสมนั้นจะมีความยืดหยุ่น (Flexibility) ในการติดตั้งและเพิ่มระดับความปลอดภัย (Security level) ให้สูงขึ้นได้ ดังภาพที่ 6 แสดงระบบตรวจจับการบุกรุกประเภทต่างๆ



ภาพที่ 6 ระบบตรวจจับการบุกรุกประเภทต่างๆ [6]

ระบบตรวจจับการบุกรุกหากแบ่งด้วยวิธีการในการตรวจจับ สามารถแบ่งได้ 5 ประเภท คือ

1. การตรวจจับจากข้อมูลการใช้งานผิดปกติ (Misuse detection) : เป็นระบบตรวจจับการบุกรุกที่ใช้หลักการตรวจหารูปแบบการบุกรุกด้วยการวิเคราะห์พฤติกรรม หรือเหตุการณ์ที่เข้ามาในเครือข่าย มาใช้เพื่อเปรียบเทียบกับข้อมูลการบุกรุกหรือกฎซึ่งจัดเก็บไว้ในฐานข้อมูลระบบ ซึ่งหากเปรียบเทียบตรงกันแสดงว่าพฤติกรรมหรือเหตุการณ์นั้นเป็นการบุกรุกการตรวจจับแบบนี้จะมีข้อดีคือสามารถโปรแกรม หรือสร้างกฎที่ใช้ในการตรวจจับการบุกรุกได้และมีความแม่นยำสูง แต่มีข้อเสียคือไม่สามารถรองรับการโจมตีรูปแบบใหม่ที่ไม่เคยบันทึกหรือเก็บไว้ในฐานข้อมูลและใช้ระยะเวลาค่อนข้างมากในการเปรียบเทียบกฎที่ใช้ในประมวลผล
2. การตรวจจับจากข้อมูลการใช้งานปกติเพื่อตรวจหาความผิดปกติ (Anomaly-based detection) : จะมีการทำงานที่ตรงข้ามกับระบบที่ตรวจจับจากข้อมูลการใช้งานผิดปกติ โดยจะเก็บข้อมูลการใช้งานที่ปกติ ในการตรวจหาความผิดปกติที่เกิดขึ้น ซึ่งส่วนใหญ่จะใช้ความรู้เกี่ยวกับการเรียนรู้ของเครื่อง (Machine Learning) และการทำเหมืองข้อมูล (Data mining) เข้ามาช่วยในการเรียนรู้พฤติกรรม หรือการใช้งานที่ปกติ เพื่อใช้ในการตรวจจับความผิดปกติ หรือการบุกรุกที่เกิดขึ้นได้ วิธีนี้มีข้อดีคือสามารถรองรับการโจมตีรูปแบบใหม่ๆได้ และมีการประมวลผลที่รวดเร็วกว่าแบบแรก แต่มีข้อเสียคือต้องอาศัยข้อมูลที่ใช้ในการเรียนรู้สูง และหากเป็นรูปแบบการใช้งานที่มีความแปลกใหม่ ที่ไม่ใช่รูปแบบการบุกรุก อาจก่อให้เกิดการตรวจจับที่ผิดพลาดเชิงบวก (Fault Positive Detection)
3. การตรวจจับโดยจากข้อมูลที่ขัดแย้งกับโพรโทคอล (Protocol anomaly detection) : เป็นการตรวจจับการบุกรุกโดยอาศัยข้อมูลของแพกเก็ตเครือข่ายที่ขัดแย้งกับโพรโทคอลที่ใช้ในการติดต่อสื่อสาร ซึ่งการตรวจจับประเภทนี้จะอาศัยการจัดคู่รูปแบบ (Pattern matching) โดยกฎที่ใช้ในการตรวจสอบจะถูกกำหนดโดยโพรโทคอลหนึ่งๆที่ใช้ ข้อดีของวิธีการตรวจจับประเภทนี้คือ มีความยืดหยุ่นในการขยาย (scalability) หรือการเพิ่มกฎที่ใช้ในการตรวจจับ แต่มีข้อเสียคือหากการโจมตีเป็นรูปแบบใหม่ที่ไม่เคยเกิดขึ้นมาก่อนทำให้ต้องพัฒนากฎใหม่เพื่อที่จะวิเคราะห์รูปแบบการโจมตีนั้น
4. วิธีการตรวจจับจากสถานะ (Stateful detection method) : เป็นวิธีการตรวจจับการบุกรุกที่มีสถานะที่แน่นอนโดยอาศัยการจดจำสถานะของเหตุการณ์ หรือการโจมตีที่มีหลายขั้นตอน เช่นกระบวนการเชื่อมต่อของโพรโทคอล ทีซีพี/ไอพี จะมีการทำ Three-way Handshake ซึ่งมีสถานะที่แน่นอน หากระบบตรวจจับได้ว่ามีสถานะที่ผิดไปจากที่กำหนดก็จะแจ้งเตือนเป็นต้น
5. การตรวจจับด้วยวิธีการทางฮิวริสติก (Heuristic-based detection) : เป็นวิธีการตรวจจับที่มาจากการวิวัฒนาการทางด้านทางด้านสถิติที่จะใช้ขั้นตอนวิธี (Algorithm) ต่างๆในการทำนายหรือตัดสินว่าข้อมูลที่พบเป็นการบุกรุกหรือไม่ ซึ่งทำให้สามารถลด

เวลาที่ใช้ในการตรวจจับการบุกรุกที่มีความซ้ำซ้อนได้ อย่างไรก็ตามวิธีการตรวจจับนี้ มักก่อให้เกิดการตรวจจับที่ผิดพลาดเชิงบวกสูง และต้องการข้อมูลก่อนหน้าจำนวนมาก

ประเภทของการตอบสนองของระบบตรวจจับการบุกรุก แบ่งได้สองประเภทคือ 1) การตอบสนองแบบตื่นตัว (Active responses) คือระบบตรวจจับการบุกรุกที่สามารถตรวจจับที่สามารถตอบสนองต่อการบุกรุกโดยอัตโนมัติได้ โดยปราศจากผู้ดูแล (Administrator) โดยสามารถเก็บข้อมูลของการบุกรุก เพื่อที่จะเลือกวิธีการตอบสนองกับการบุกรุกนั้นๆได้อย่างถูกต้อง เช่นระงับการโจมตีโดยการตัดการเชื่อมต่อ เป็นต้น 2) การตอบสนองแบบไม่ดิ้นรน (Passive responses) ระบบตรวจจับการบุกรุกที่เมื่อพบการบุกรุกจะทำการแจ้งเตือนไปยังผู้ดูแลระบบ หรือระบบที่รับผิดชอบในการจัดการการบุกรุกให้ทราบถึงการบุกรุก โดยปราศจากการโต้ตอบแบบอัตโนมัติจากระบบ ตรวจจับการบุกรุก

การแบ่งประเภทของระบบตรวจจับการบุกรุกโดยใช้เกณฑ์เวลาที่ใช้ในการทวนสอบข้อมูล (audit information) สามารถแบ่งได้ 2 ประเภทคือ

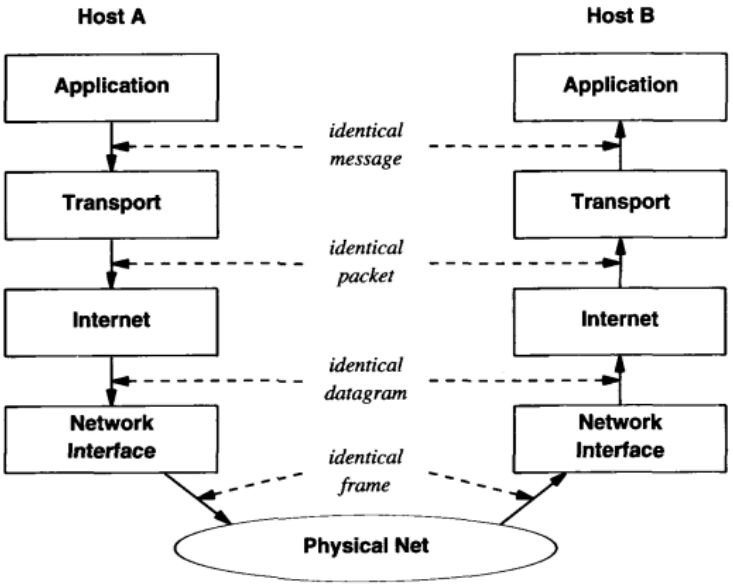
1. การทวนสอบวิเคราะห์ในเวลาจริง(Real-time audit analysis) : คือระบบตรวจจับวิเคราะห์ที่ประมวลผลข้อมูลที่ต่อเนื่อง มักใช้ระบบตรวจจับการบุกรุกเครือข่าย โดยทำการตรวจจับข้อมูลการบุกรุกพร้อมๆ กับการติดต่อสื่อสารซึ่งการตรวจสอบประเภทนี้ต้องการหน่วยประมวลผลที่มีความสามารถสูงมาก เพื่อให้สามารถวิเคราะห์ข้อมูลได้อย่างรวดเร็ว โดยอย่างน้อยต้องสามารถพบการบุกรุกก่อนเวลาที่การบุกรุกนั้นจะส่งผลต่อระบบที่เฝ้าระวังเพื่อที่จะสามารถลดความเสียหาย หรือระงับการบุกรุกได้ทันเวลา
2. การทวนสอบวิเคราะห์ภายหลังเหตุการณ์ (Post-event audit analysis) : หรือ ระบบตรวจจับการบุกรุกแบบช่วงเวลา (Interval-base IDS) คือระบบการตรวจจับการบุกรุกที่ส่วนของประมวลผลข้อมูลการบุกรุก และส่วนที่รวบรวมข้อมูลเพื่อทำการวิเคราะห์ทำงานไม่มีความต่อเนื่อง หรือทำงานเป็นช่วงเวลา โดยทั่วไปจะใช้กับการสื่อสารที่มีลักษณะการทำงานแบบเก็บรวบรวม (Store) และส่งต่อ (Forward) หรือใช้ในการวิเคราะห์ข้อมูลการบุกรุกที่มีความซ้ำซ้อนมากๆ ทำให้ต้องใช้เวลาในการประมวลผลที่นาน

2.1.3 โพรโทคอลทีซีพี/ไอพี (TCP/IP Protocol) [7]

โพรโทคอลทีซีพีเป็นโพรโทคอลที่ใช้ในการควบคุมรูปแบบการสื่อสารในเครือข่ายที่ถูกพัฒนาขึ้นโดยกระทรวงกลาโหมของสหรัฐอเมริกาหรือ DOD (Department of Defense) ซึ่งเริ่มแรกใช้ในการทหารเป็นหลัก และด้วยการเติบโตที่รวดเร็วของระบบเครือข่ายจึงทำให้โพรโตคอลทีซีพี/ไอพีเป็นที่แพร่หลาย โดยโพรโทคอลทีซีพี/ไอพีมีเป้าประสงค์คือ เพื่อให้มีความน่าเชื่อถือ (Reliability) ในการส่งข้อมูลจากต้นทางไปยังปลายทางได้อย่างถูกต้องสมบูรณ์ โดยมีวัตถุประสงค์ของโพรโตคอลทีซีพี/ไอพีดังนี้

1. เพื่อให้การติดต่อสื่อสารระหว่างระบบที่แตกต่างกันได้
2. เพื่อลดความผิดพลาดในการส่งข้อมูลจากต้นทางไปยังปลายทาง โดยมีการตรวจสอบข้อมูลทั้งผู้รับและผู้ส่ง

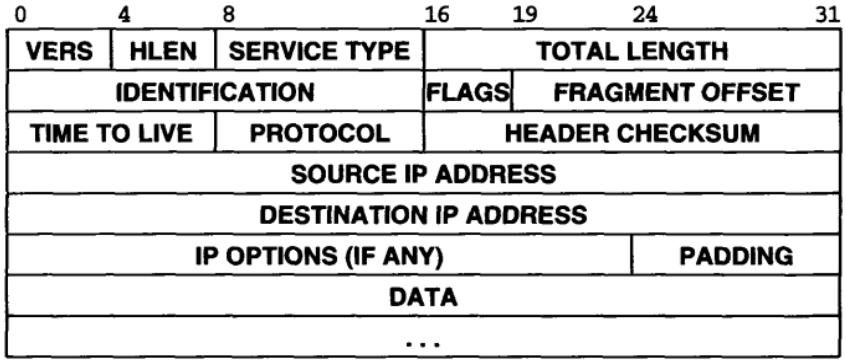
3. ช่วยในการแก้ไขปัญหากรณีที่เกิดปัญหาขึ้นกับการส่งข้อมูลจากต้นทางไปยังปลายทาง เพื่อไม่ให้ข้อมูลของผู้ส่งและผู้รับเกิดความเสียหาย
4. สามารถค้นหาเส้นทางที่ใช้ในการติดต่อจากผู้ส่งและผู้รับได้อย่างอัตโนมัติ
5. มีความคล่องตัวในการสื่อสารข้อมูลหลายรูปแบบทั้งที่ต้องการความเร็วในการส่งข้อมูล และต้องการความถูกต้องของข้อมูล



ภาพที่ 7 โครงสร้างแบบชั้นของโพรโทคอลทีซีพี/ไอพี [7]

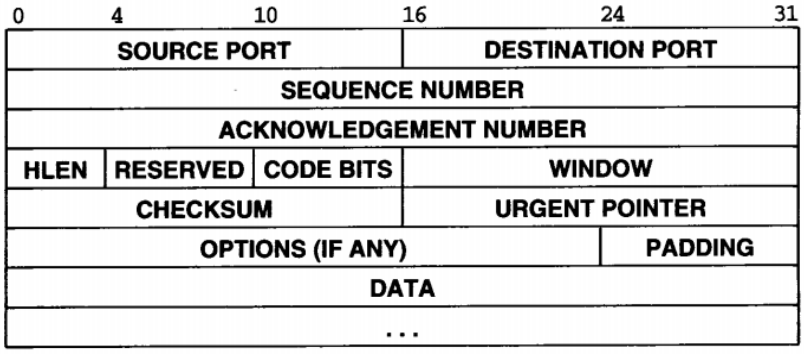
ชุดโพรโทคอลทีซีพี/ไอพี (TCP/IP Protocol suite) ถูกออกแบบด้วยหลักการของโครงสร้างแบบชั้น โดยการส่งข้อมูลจากใช้การหุ้มแค็ปซูล(Encapsulate) และการถอดแค็ปซูล (Decapsulate) ข้อมูลในแต่ละชั้นโดยสามารถแบ่งชั้นที่รับผิดชอบการทำงานออกแป็น 4ชั้น (ภาพที่ 7) คือ

1. ชั้นโฮสต์ทูโฮสต์ (Host to host layer) : คือชั้นที่รับผิดชอบการทำงานให้สามารถเชื่อมต่อจากเครื่องต้นทางไปยังเครื่องปลายทางได้ โดยจะอยู่ในเครือข่ายท้องถิ่น (Local network) เท่านั้น ซึ่งโพรโตคอลหลักที่ใช้ในการระบุตำแหน่งของเครื่องต่างๆ คือ อาร์พโพรโทคอล (ARP protocol) ซึ่งจะทำการค้นหาและระบุตำแหน่งของโหนดต่างๆ เพื่อให้สามารถส่งข้อมูลได้อย่างถูกต้อง
2. ชั้นอินเทอร์เนต (Internet layer) : คือชั้นที่รับผิดชอบในการเชื่อมต่อระหว่างเครือข่าย โดยการส่งข้อมูลที่ขนาดเล็กที่เรียกว่าไอพีดาตาแกรม (IP datagram) ในภาพที่ 8 ซึ่งเป็นการสื่อสารโดยในหมายเลขที่อยู่ไอพี (IP address) เป็นตัวกำหนดต้นทางและปลายทาง โดยสามารถส่งเป็นชุดข้อมูลที่มีจุดหมายปลายทางเดียวกันได้ แต่ชุดข้อมูลที่ส่งอาจถึงปลายทางไม่พร้อมกันเนื่องจากการส่งข้อมูลในชั้นนี้จะมีกระบวนการกระจายข้อมูลไปในเครือข่าย (Fragmentation) และกระบวนการรวบรวมที่ปลายทาง (Defragmentation) เพื่อในกรณีที่ข้อมูลที่ส่งมาไม่มีลำดับก็จะสามารถรวบรวมและเรียงตัวเหมือนข้อมูลต้นทางได้



ภาพที่ 8 โครงสร้างของไอพีดาตาแกรม [7]

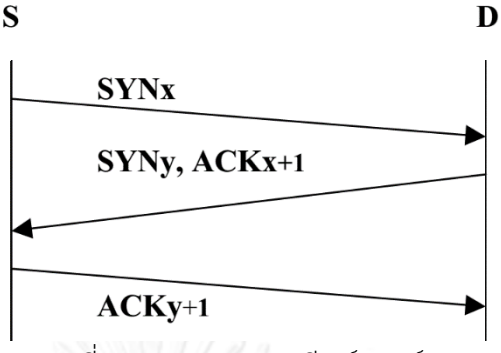
3. ชั้นทรานสปอร์ต (Transport layer) : คือชั้นที่รับผิดชอบในการควบคุมการเชื่อมต่อจากต้นทางไปยังปลายทาง โดยสามารถแบ่งโพรโทคอลออกเป็น 2 ลักษณะการเชื่อมต่อ 1) โพรโทคอลทีซีพี (TCP Protocol) เป็นโพรโทคอลที่มีการเชื่อมต่อแบบชี้นำ (Connection-oriented) ซึ่งภายในแพกเก็ต (Packet) ในส่วนหัวของทีซีพี (TCP header) ดังภาพที่ 9 จะมีข้อมูลที่ตรวจสอบลำดับการตอบ และรับระหว่างต้นทางและปลายทางที่เรียกว่า (Acknowledgement) เพื่อยืนยันว่าข้อมูลที่ส่งถึงที่หมายปลายทางอย่างถูกต้องสมบูรณ์ โดยในขณะเริ่มการเชื่อมต่อของโพรโทคอลทีซีพีไอพีจะมีการทำกระบวนการสร้างการเชื่อมต่อที่เรียว่าทรีเวย์แฮนด์แชค (3-way handshake) ระหว่างเครื่องต้นทางและปลายทางก่อนที่จะส่งข้อมูลดังภาพที่ 10 และ 2) โพรโตคอลยูดีพี (UDP Protocol) เป็นโพรโทคอลที่เน้นการสื่อสารที่ต้องการความเร็ว โดยจะมีการตรวจสอบข้อมูลความถูกต้อง แต่จะไม่มีการตอบกลับแต่อย่างใด เรียกการเชื่อมต่อแบบนี้ว่าหารเชื่อมต่อแบบไม่มีการชี้นำ (Connectionless) โดยทั่วไปจะใช้กับการสื่อสารต้องการส่งข้อมูลแบบต่อเนื่อง (Stream) ที่ต้องการความเร็วมากกว่าความถูกต้องของข้อมูล เช่น การสื่อสารแบบเสียงบนไอพี (Voice over IP) หรือ การถ่ายทอดวีดีโอทางอินเตอร์เนต



ภาพที่ 9 โครงสร้างของทีซีพีแพกเก็ต [7]

4. ชั้นการประยุกต์ (Application layer) : คือชั้นที่รับผิดชอบการเชื่อมต่อระหว่างแอพลิเคชันระหว่างเครื่องต้นทางและปลายทาง ตัวอย่างเช่น โพรโทคอลเอฟทีพี (FTP) ที่ควบคุมการส่งข้อมูลรหว่างระบบไฟล์ หรือโพรโทคอลเอสเอ็มทีพี (SMTP) ที่ดูแลการทำงานของ

ระบบจดหมายอิเล็กทรอนิกส์



ภาพที่ 10 กระบวนการทรีเวย์แฮนด์แชค

2.1.4 การบุกรุกบนโพรโทคอลทีซีพี/ไอพี (TCP/IP Attack) [8]

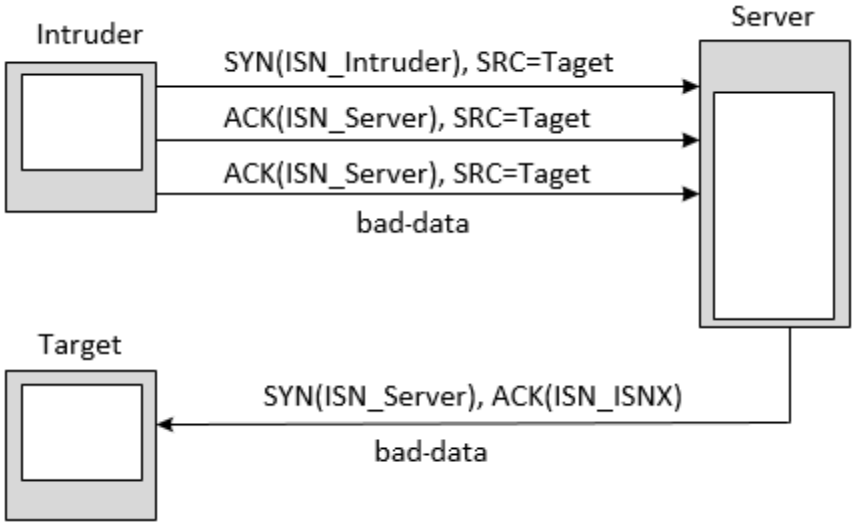
การโจมตีบนโพรโทคอลทีซีพี/ไอพีที่สำคัญมี 6 ประเภทดังนี้

1. การโจมตีเพื่อการระงับการให้บริการ (Denial of service attack) เป็นการโจมตีที่ทำเพื่อระงับการให้เป้าหมายไม่สามารถให้บริการได้เช่น การโจมตีโจมตีโดยใช้การเปิดการเชื่อมต่อกับเซิร์ฟเวอร์ที่ให้บริการค้างไว้ โดยการส่งแพกเก็ต Syn ปริมาณมากไปยังเซิร์ฟเวอร์เป้าหมาย และไม่ตอบแพกเก็ต Syn/Ack ที่ส่งมาจากเซิร์ฟเวอร์เป้าหมาย โดยในโพรโทคอลทีซีพีไอพี จะมีเครื่องที่ให้บริการจะมีช่วงเวลาในการรอ Ack แพกเก็ตจากเครื่องที่ร้องขอก่อนที่จะลบแพกเก็ต Syn ออกจากบัฟเฟอร์อย่างน้อย 75 วินาที ทำให้ไม่สามารถให้บริการได้เครื่องที่ร้องขออื่นๆได้ เนื่องจากบัฟเฟอร์ที่เก็บการเชื่อมต่อไม่ว่าง ซึ่งการป้องกันการโจมตีประเภทนี้จะให้การกรองหรือดักจับแพกเก็ตที่เข้ามาบ่อยๆ จำนวนมากเพื่อป้องกันหรือตัดการเชื่อมต่อกับการร้องขอนั้น
2. การโจมตีโดยการดักจับแพกเก็ต (Packet sniffing attack) คือกระบวนการที่ผู้บุกรุกดักจับแพกเก็ตในเครือข่ายเพื่อนำข้อมูลไปใช้ ซึ่งไปการทำลายการรักษาความลับของข้อมูล (Data confidentiality) หรืออาจทำให้เกิดการบุกรุกแบบคนกลาง (man-in-the-middle attack) ที่ผู้โจมตีที่เป็นคนกลางนำข้อมูลที่ดักจับได้ไปปรับแก้ไขหรือปลอมแปลงข้อมูลแพกเก็ตแล้วส่งต่อไปยังปลายทางจึงเป็นการทำลายความถูกต้องตรงกันของข้อมูล (Data integrity) ระหว่างผู้ส่งและผู้รับข้อมูลเช่นการดักจับจดหมายอิเล็กทรอนิกส์ (Email) หรือข้อมูลทางการเงินเพื่อปลอมแปลงแล้วส่งต่อไปยังผู้รับส่งผลให้เกิดความเสียหายเป็นต้น ซึ่งการป้องกันการโจมตีประเภทนี้สามารถทำได้โดยการเข้ารหัสลับ (encrypt) ข้อมูลที่จะส่งในเครือข่าย โดยขั้นตอนวิธีที่นิยมในปัจจุบันเช่น MD4, MD5, และ SSH เป็นต้นเพื่อเป็นการรักษาความลับของข้อมูลหรือใช้การยืนยันตัวบุคคล (Authentication) ของทั้งผู้รับและผู้ส่งที่มีสิทธิ์ในการเข้าถึงข้อมูลได้ ตัวอย่างระบบการยืนยันตัวบุคคลเช่น Kerberos อย่างไรก็ตามองค์กรหลายองค์กรที่ใช้ระบบการบืนยันตัวบุคลลก็ยังมีความเสี่ยงในการรักษาความลับของข้อมูลตัวอย่างเช่น การเชื่อมต่อผ่านโพรโทคอล Telnet ซึ่งมีการเข้ารหัสลับข้อมูลของรหัสผ่านที่ใช้ในการยืนยันตัวบุคคล อย่างไรก็ตามผู้โจมตีสามารถดักจับข้อมูลที่เข้ารหัสลับ และนำไป

ถอดรหัสลับ (decrypt)ได้หากรหัสผ่านที่ใช้ไม่แข็งแกร่งพอซึ่งเป็นความผิดพลาดของมนุษย์ (Human error)

3. การโจมตีแบบปลอมแปลง (Spoofing attack) เป็นการโจมตีที่สามารถเกิดขึ้นกับฮาร์ดแวร์ และซอฟต์แวร์ที่ใช้ในการเชื่อมต่อทำให้การส่งข้อมูลจากต้นทางไปยังปลายทางผิดพลาด โดยการโจมตีประเภทนี้ตรวจจับได้ยากมากสำหรับการใช้งานเครือข่ายในปัจจุบัน เช่นการเชื่อมต่อแบบเซิร์ฟเวอร์ไคลแอนต์ (Server-Client) ในโพรโทคอลทีซีพี ที่มีหมายเลขลับดับในการเชื่อมต่อ (Sequence number) ของแพกเก็ตที่ส่งเพื่อป้องการความซ้ำซ้อนและความผิดพลาดในการส่งข้อมูล ซึ่งระหว่างการเริ่มต้นการเชื่อมต่อของโพรโทคอลทีซีพี จะมีกระบวนการกระบวนการทรีเวย์แฮนด์แชคเพื่อสร้างการเซสชั่น (Session) การเชื่อมต่อระหว่างเครื่องที่ให้บริการและเครื่องปลายทาง ซึ่งผู้บุกรุกที่สามารถล่วงรู้หมายเลขลำดับที่ใช้ในการเชื่อมต่อสามารถสร้างแพกเก็ตปลอมโดยใช้หมายเลขการเชื่อมต่อนั้นในการส่งข้อมูลเพื่อหลอกผู้บุกรุกได้รับเซสชันการเชื่อมต่อไป โดยทั่วไปเรียกการโจมตีแบบนี้ว่า การช่วงชิงเซสชัน (Session hijacking)
4. การโจมตีตารางโปรเซส (Process table attack) ซึ่งจัดอยู่ในการรูปแบบการโจมตีเพื่อการระงับการให้บริการประเภทหนึ่งแต่สาเหตุเกิดจากการสร้างโปรเซสการทำงานในตารางโปรเซสมากเกิน ทำให้ใช้งานทรัพยากรเช่น หน่วยประมวลผลหรือพื้นที่หน่วยความจำของเครื่องที่ให้บริการเต็ม ส่งผลให้ให้ไม่สามารถใหบริการได้ โดยการโจมตีประเภทนี้เกิดจากลักษณะการเชื่อมต่อที่เครื่องที่ให้บริการต้องมีการสร้างโปรเซสที่จัดการประมวลผลข้อมูลการร้องขอก่อนแล้วตอบกลับไปยังเครื่องที่ทำการร้องขอ ตัวอย่างเช่นในการทำงานขอไปยังหมายเลขพอร์ต 79 หรือโพรโทคอลฟิงเกอร์ (finger protocol) ในทีซีพี/ไอพีที่จะมีการสร้างโปรเซสการทำงานที่รอรับคำสั่งจากเครื่องที่ร้องขอในระบบปฏิบัติการยูนิกซ์ (Unix) แล้วสร้างโปรเซสตามคำสั่งที่มีการร้องขอนำไปเก็บในตารางโปรเซสเป็นต้น โดยการป้องกันการโจมตีประเภทนี้สามารถทำได้โดยการตรวจสอบจำนวนช่องที่ว่างสำหรับเก็บโปรเซสในตารางโปรเซส หรือการกำหนดค่าเวลาสิ้นสุด (timeout) ของการเชื่อมต่อ
5. การโจมตีโดยกระบวนการสร้างหมายเลขลำดับของโพรโทคอลทีซีพี (TCP sequence number generation attack) ซึ่งโดยทั่วไปหมายเลขลำดับในโพรโทคอลทีซีพีจะใช้ในการกำหนดลำดับการส่งข้อมูลในทุกๆไบต์ (Byte) ทั้งของเครื่องต้นทางและเครื่องปลายทางเพื่อให้ทั้งสองฝั่งสามารถใช้หมายเลขลำดับของทั้ง 2 ฝั่งในการตรวจสอบความถูกต้องของลำดับข้อมูลที่ได้รับ ซึ่งในกระบวนการการทรีเวย์แฮนด์แชคเพื่อเริ่มต้นการเชื่อมต่อจะมีการสร้างหมายเลขลำดับเริ่มต้น (initial sequence number) ISN ที่จะใช้ในการเชื่อมต่อของเครื่องต้นทางโดยการสุ่มค่า ISN ของเครื่องต้นทางซึ่งจะถูกส่งไปในแพกเก็ต Syn ไปยังเครื่องปลายทางจากนั้นเครื่องปลายทางจะตอบกลับด้วยแพกเก็ต Syn/Ack ไปยังเครื่องต้นทางโดยระบุค่า ISN เครื่องปลายทางที่จะใช้ในการเชื่อมต่อ ซึ่งจากรูปแบบการทำงานดังกล่าวผู้โจมตีสามารถโจมตีเครื่องเป้าหมายได้ โดยการสร้างแพกเก็ต Syn ที่มีการระบุหมายเลข ISN ของเครื่องผู้บุกรุกแต่ระบุหมายเลขไอพีแอสเด

รสเป็นของเครื่องเป้าหมายไปยังเครื่องเซิร์ฟเวอร์ที่ให้บริการ ทำให้เครื่องเซิร์ฟเวอร์ส่งแพกเก็ต Syn/Ack ที่ระบุหมายเลข ISN ของเครื่องเซิร์ฟเวอร์ไปยังเครื่องเป้าหมาย ซึ่งผู้โจมตีจะไม่ทราบถึงหมายเลข ISN ของเครื่องเซิร์ฟเวอร์นั้น แต่ผู้โจมตีจะทำการสร้าง Ack แพกเก็ตที่มีข้อมูล ISN ที่ถูกสร้างขึ้น (โดยการสุ่มเพื่อให้ตรงกับ ISN ของเครื่องเซิร์ฟเวอร์ที่ให้บริการที่ส่งไปยังเครื่องเป้าหมาย)จำนวนมากพร้อมกับส่งข้อมูลไม่ดีไปยังเครื่องเซิร์ฟเวอร์โดยระบุหมายเลขไอพีแอสเดรสเป็นของเครื่องเป้าหมาย ซึ่งหาก ISN ของเครื่องเซิร์ฟเวอร์ที่ส่งมาในแพกเก็ต Ack ตรงกับISN ของเครื่องเซิร์ฟเวอร์ที่ส่งไปยังเครื่องเป้าหมาย จะทำให้เครื่องเซิร์ฟเวอร์รับข้อมูลไม่ดีที่อาจเป็นคำสั่งที่ก่อให้เกิดความเสียหายและตอบข้อมูลผลลัพธ์ไปยังเครื่องเป้าหมายในแพกเก็ต Syn/Ack ของเครื่องเซิร์ฟเวอร์ที่ให้บริการดังภาพที่ 11



ภาพที่ 11 การโจมตีโดยกระบวนการสร้างหมายเลขลำดับ

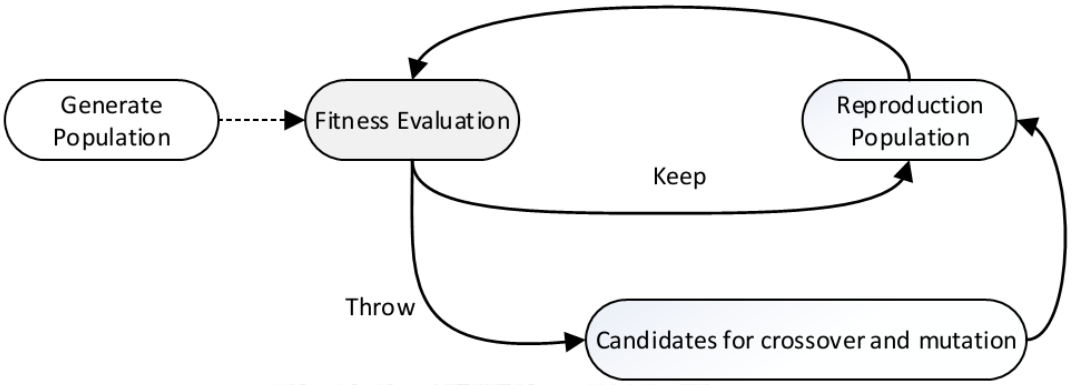
6. การโจมตีแบบไอพีฮาร์ฟแสกน (IP half scan attack) เป็นการโจมตีที่ผู้บุกรุกต้องการทราบข้อมูลของเครื่องเป้าหมาย โดยเครื่องเป้าหมายไม่สามารถบันทึกล็อกของการเชื่อมต่อไว้ได้ (เพราะไม่เกิดการเชื่อมต่อจริง) เนื่องจากผู้บุกรุกอาศัยช่วงโหว่ของกระบวนการทรีเวย์แฮนด์แชคในโพรโทคอลทีซีพี/ไอพี โดยการส่งเพียงแพกเก็ต Syn ไปยังพอร์ตต่างๆของเครื่องเป้าหมาย และส่งแพกเก็ต Rst เพื่อรีเซตหรือยกเลิกการเชื่อมต่อนั้น โดยหากบางพอร์ตของเครื่องเป้าหมายมีการเปิดให้บริการ ก็จะมีการตอบกลับไปยังผู้โจมตีก่อนที่จะมีการยกเลิกการเชื่อมต่อนั้น

2.1.5 ขั้นตอนวิธีเชิงพันธุกรรม (Genetic algorithm) [9]

ขั้นตอนวิธีเชิงพันธุกรรม เป็นการเรียนรู้ที่จำลองวิวัฒนาการของสิ่งมีชีวิต (Biological Evolution) ที่มีกฎการคัดเลือกโดยธรรมชาติ (Natural selection) ซึ่งเลือกสิ่งมีชีวิตที่เหมาะสม (Fitness) หรือมีแนวโน้มของการสืบลูกหลานมากกว่าก็จะสามารถดำรงอยู่รอดเป็นรุ่นถัดไป นอกจากนี้ในทางชีววิทยาเซลล์ของสิ่งมีชีวิตจะประกอบด้วยโครโมโซม (Chromosome) จำนวนหนึ่ง ซึ่งแต่ละโครโมโซมจะได้รับการถ่ายทอดรหัสพันธุกรรมที่มีลักษณะของการจับคู่กันจากของรุ่นก่อน

หน้าโดยมีการไขว้เปลี่ยน (Crossover) รหัสพันธุกรรมระหว่างคู่โครโมโซม หรืออาจเกิดการเปลี่ยนรหัสพันธุกรรมแบบที่ไม่เคยเกิดขึ้นมาก่อนหรือที่เรียกว่าการกลายพันธ์ (Mutation)

การประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรมสามารถใช้ในการแก้ปัญหาการหาค่าที่ดีที่สุด (Optimization problem) หรือการหาคำตอบที่ดีที่สุด (Optimal solution) โดยการเข้ารหัสโครโมโซมสำหรับปัญหาเกิดขึ้นเพื่อหาค่าคำตอบหรือค่าความเหมาะสมที่ดีที่สุดพร้อมกับตัวแปรต่างๆ ที่ทำให้เกิดคำตอบที่เหมาะสมหรือค่ารหัสพันธุกรรมในโครโมโซมซึ่งมีค่าความเหมาะสมดีที่สุด โดยกระบวนการของขั้นตอนวิธีเชิงพันธุกรรมมีขั้นตอนดังภาพที่ 12



ภาพที่ 12 ขั้นตอนทั่วไปของขั้นตอนวิธีเชิงพันธุกรรม

ขั้นตอนวิธีเชิงพันธุกรรมเป็นกระบวนการวนซ้ำโดยเริ่มจากการสร้างสมติฐานต่างๆ หรือประชากรรุ่นแรกจำนวน p โครโมโซม จากนั้นประชากรจะเข้าสู่การประเมินค่าความเหมาะสม (Fitness Evaluation) โดยเมื่อผ่านการประเมินแล้ว ค่าความเหมาะสมจะเป็นตัวบ่งบอกความน่าจะเป็นขอกระบวนการคัดเลือกโดยธรรมชาติ ซึ่งสมมติฐานที่มีค่าความเหมาะสมสูงจะมีความน่าจะเป็นที่จะถูกคัดเลือกเป็นประชากรรุ่นถัดไปสูงกว่าสมมติฐานอื่น แต่ไม่จำเป็นต้องถูกเลือกเสมอ และด้วยอัตราส่วนการแทนที่ (Replacement ratio) ที่กำหนดไว้จำนวนสมาชิกในประชากรจำนวน rp จะถูกนำเลือกด้วยความน่าจะเป็นไปทำกระบวนการไขว้เปลี่ยนและการกลายพันธุ์ให้กลายเป็นสมมติฐานที่มีลักษณะแต่กต่างจากเดิมซึ่งจะถูกนำไปรวมกับสมาชิกจำนวน (1-r)p ตัวประกอบกันเป็นสมาชิกในรุ่นถัดไป

ข้อดีของขั้นตอนวิธีเชิงพันธุกรรมมีดังนี้

1. สามารถค้นหาคำตอบของสมมติฐานไปพร้อมๆกับการค้นหาสมมติฐานนั้นๆ
2. มีความยืดหยุ่นสามารถปรับใช้กับปัญหาได้หลากหลายรูปแบบ ขึ้นอยู่กับการเข้ารหัสพันธุกรรมของปัญหา
3. ขั้นตอนวิธีเชิงพันธุกรรมสามารถค้นหาคำตอบของปัญหาโดยหาจากสมมติฐานจำนวนมากทำให้มีโอกาสที่จะหลุดจากค่ามากที่สุดท้องถิ่นได้ง่ายกว่าการค้นหาแบบโดยทั่วไปที่ค้นหาคำตอบแบบทีละสมมติฐาน

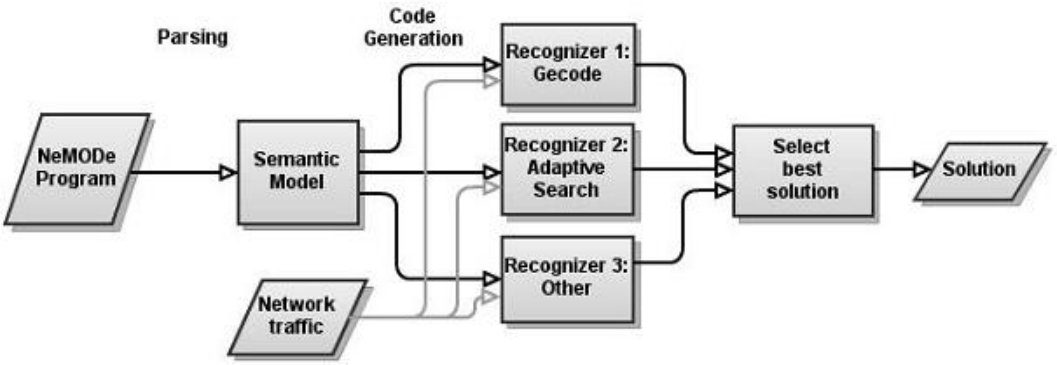
ข้อจำกัดของขั้นตอนวิธีเชิงพันธุกรรมมีดังนี้

1. ความยากในการเข้ารหัสปัญหาด้วยโครโมโซม ซึ่งสามารถหาคำตอบที่ดีที่สุดด้วยฟังก์ชันความเหมาะสม
2. ไม่มีการรับประกันคำตอบที่ดีที่สุด เนื่องจากขั้นตอนวิธีเชิงพันธุกรรมอาศัยการค้นหาค่าที่ดีที่สุดซึ่งอาจติดปัญหาค่าดีที่สุดท้องถิ่น (Local optima)
3. ใช้เวลาในการประมวลผลมาก เนื่องจากเมื่อฟังก์ชันความเหมาะสมประกอบด้วยหลายขั้นตอนในการประเมินค่า

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 Using Constraints for Intrusion Detection: the NeMoDe system [10]

เป็นงานวิจัยที่นำเสนอการประยุกต์ใช้ Constraint programming ในการค้นหารูปแบบพฤติกรรมการบุกรุกเครือข่าย โดยสามารถเขียนเงื่อนไขข้อจำกัด (Constraint) ในรูปของภาษาจำเพาะโดเมนเพื่อให้ระบบตรวจจับการบุกรุก NeMoDe สามารถตรวจจับการบุกรุกที่มีลักษณะลำดับเวลาการมาถึงของแพกเก็ตเครือข่ายได้ ซึ่งการค้นหาพฤติกรรมการบุกรุกเครือข่ายของ NeMoDe อาศัยวิธีการค้นหาแบบ Adaptive search และ Gecode โดยมีการเปรียบเทียบข้อดีข้อเสียของการค้นหาทั้ง 2 รูปแบบ โดยในภาพที่ 13 แสดงสถาปัตยกรรมระบบของ NeMoDe



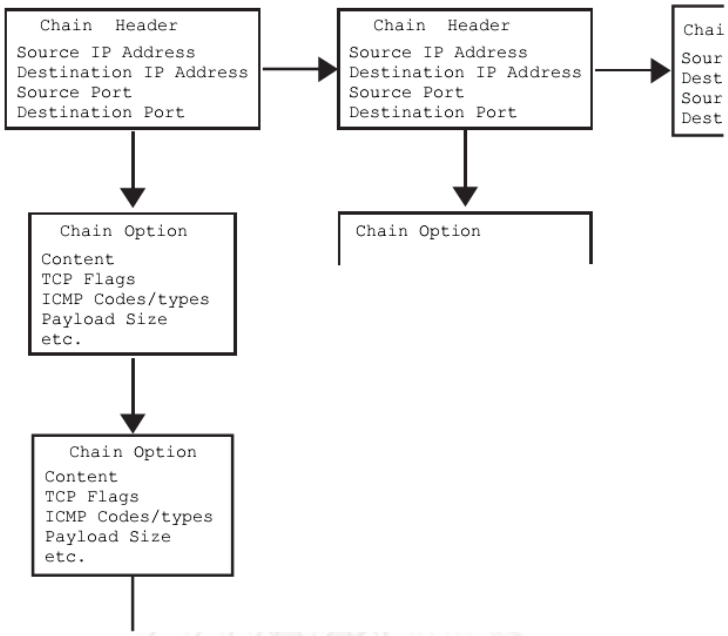
ภาพที่ 13 สถาปัตยกรรมระบบของ NeMoDe [10]

โดยภาษาจำเพาะโดเมนที่งานวิจัยนี้นำเสนอจะอาศัยหลักการทำงานของการโปรแกรมข้อจำกัด (Constraint programming) เพื่อสร้างโค้ดที่จะใช้งานกับระบบค้นหาการบุกรุกเบื้องหลังได้

2.2.2 Snort-Lightweight Intrusion Detection for Networks [2]

เป็นงานวิจัยที่นำเสนอคุณสมบัติและข้อดีของผลิตภัณฑ์ระบบตรวจจับการบุกรุกเครือข่าย Snort ที่นำเสนอระบบตรวจจับการบุกรุกเครือข่ายที่มีแนวความคิดที่จะลดความซ้ำซ้อนและค่าใช้จ่ายในการติดตั้งระบบตรวจจับการบุกรุกเครือข่ายสำหรับองค์กร และนำเสนอสถาปัตยกรรมทั้ง 3 ส่วนของ snort ที่ประกอบด้วยระบบถอดรหัสข้อมูลเครือข่าย ระบบตรวจจับการบุกรุก และระบบแจ้งเตือนผู้ใช้ในกรณีพบการบุกรุกนอกจากนี้งานวิจัยมีการแสดงลักษณะของกฎ รวมถึงโครงสร้าง

การค้นหารูปแบบพฤติกรรมที่เป็นการบุกรุกเครือข่ายแบบลูกโซ่ของ snort ดังภาพที่ 14 ตัวอย่างกฎที่ใช้ใน snort และภาพที่ 15



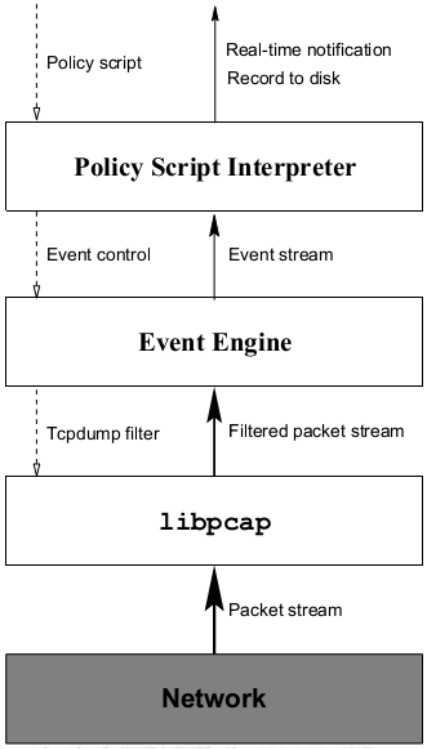
ภาพที่ 14 โครงสร้างการค้นหารูปแบบพฤติกรรมที่เป็นการบุกรุกเครือข่ายแบบลูกโซ่ [2]

```
alert tcp any any -> any 80 (msg:"CGI-nph-tst-cgi";
                content:"cgi-bin/nph-test-cgi?"; flags: PA;)
alert tcp any any -> any 80 (msg:"CGI-test-cgi";
                content:"cgi-bin/test-cgi?"; flags: PA;)
alert tcp any any -> any 80 (msg:"CGI-perl.exe";
                content:"cgi-bin/perl.exe?"; flags: PA;)
alert tcp any any -> any 80 (msg:"CGI-phf";
                content:"cgi-bin/phf?"; flags: PA;)
```

ภาพที่ 15 ตัวอย่างกฎการตรวจจับการบุกรุกของ snort [2]

2.2.3 Bro: A System for Detecting Network Intruders in Real-Time [3]

เป็นงานวิจัยเสนอแนวความคิดในการพัฒนาระบบตรวจจับการบุกรุกเครือข่ายที่สามารถรองรับการทำงานในเครือข่ายขนาดใหญ่และสามารถทำงานในเวลาจริงได้ โดยการทำงานจะมีการกำหนดนโยบายของระบบที่สร้างจากภาษาโบร (Bro language) รวมถึงโครงสร้างการทำงานของระบบโบร ดังภาพที่ 16 นอกจากนี้งานวิจัยได้นำเสนอปัญหาและข้อจำกัดของการพัฒนารวมถึงช่องโหว่ในการทำงานของระบบตรวจจับการบุกรุกในสภาวะดังกล่าว ซึ่งตัวอย่างปัญหาที่พบในการทำงาน



ภาพที่ 16 โครงสร้างของระบบโบร

2.2.4 Detection of Performance Deviations in the Load Testing of Large Scale Systems [11]

เป็นงานวิจัยที่เกี่ยวข้องกับการนำตัวนับสมรรถนะของระบบ (Performance counter) มาใช้ในการวิเคราะห์ว่าระบบนั้นเป็นไปตามข้อตกลงระดับการบริการ หรือ เอสแอลเอ (Service Level Agreements - SLA) หรือไม่ โดยในงานวิจัยนี้ได้มีการระบุตัวแปรนับสมรรถนะของระบบไว้ดังภาพที่ 17

No	Performance Counter Variables	10-Fold Selection	Count= 5	% Freq>20
1	% CPU Time	10	✓	✓
2	% Disk Time	10	✓	✓
3	Disk sec/Write	1		
4	Available Bytes	2		
5	Pages/sec	5	✓	✓
6	Database Cache Request/Sec	4		✓
7	Network Interface Bytes Total/sec	5	✓	✓
8	% CPU Idle Time	6	✓	✓
9	Datagram Rec/sec	1		
⋮	⋮	⋮	⋮	⋮
18	Avg. Disk Read Queue Length	0		

ภาพที่ 17 ตัวแปรนับสมรรถนะของระบบ

2.2.5 Intrusion Detection with Neural Network [12]

เป็นงานวิจัยที่สร้างระบบตรวจจับการบุกรุกโดยใช้โครงข่ายประสาทเทียมมาใช้ในการเรียนรู้และจดจำพฤติกรรมการใช้งานระบบ เช่น ระยะเวลาหรือความถี่ในการใช้งานคำสั่งในระบบ โดยใช้ข้อมูลจากบันทึกการใช้งานมาใช้ในการฝึกโครงข่ายประสาทเทียมเพื่อให้สามารถจดจำ และแยกแยะการบุกรุกได้ งานวิจัยที่นำเสนอได้ประยุกต์แนวคิดการใช้ข้อมูลจากระบบ มาช่วยตรวจจับรูปแบบการบุกรุกเครือข่ายโดยการนำข้อมูลบันทึกคำสั่งที่ผู้ใช้งานใช้มาใช้ในการวิเคราะห์เพื่อจำแนกการบุกรุกตามภาพที่ 18

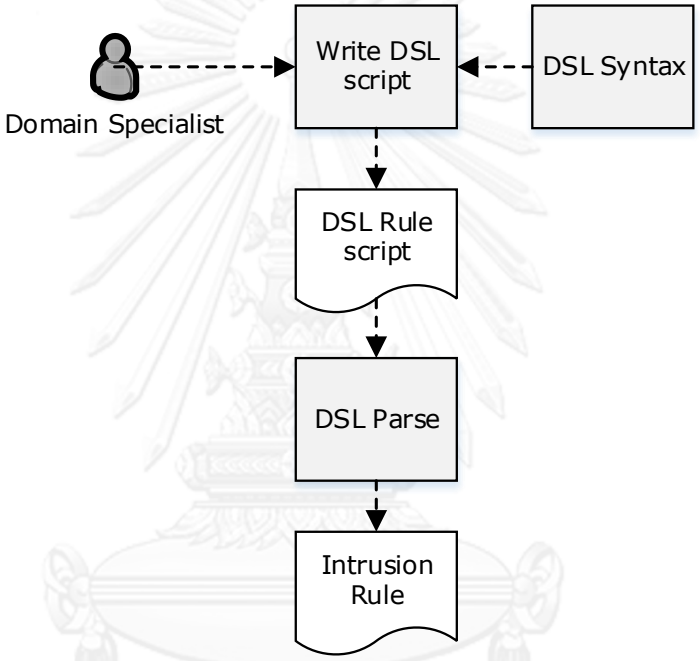
as	awk	bc	bibtex	calendar	cat	chmod	comsat	cp	cpp
cut	cvs	date	df	diff	du	dvips	egrep	elm	emacs
expr	fgrep	filter	find	finger	fmt	from	ftp	gcc	gdb
ghostview	gmake	grep	gs	gzip	hostname	id	ifconfig	ispell	last
ld	less	look	lpq	lpr	lprm	ls	machine	mail	make
man	mesg	metamail	mkdir	more	movemail	mpage	mt	mv	netscape
netstat	nm	objdump	perl	pgp	ping	ps	pwd	rcp	resize
rm	rsh	sed	sendmail	sh	sort	strip	stty	tail	tar
tcsh	tee	test	tgif	top	tput	tr	tty	uname	vacation
vi	virtex	w	wc	whereis	xbiff++	xcalc	xdvi	xhost	xterm

ภาพที่ 18 รายการคำสั่งที่ใช้ในการวิเคราะห์

บทที่ 3
วิธีดำเนินงานวิจัย

3.1 แนวคิดในการพัฒนา

งานวิจัยนี้เสนอภาษาจำเพาะโดเมนเพื่อการกำหนดรูปแบบพฤติกรรมการบุกรุก (Intrusion signature domain specific language) หรืออีสดีเอสแอล (isDSL) เพื่อให้ผู้เชี่ยวชาญสามารถสร้างกฎในการตรวจจับการบุกรุกเครือข่ายแบบใหม่ได้ง่าย โดยอาศัยข้อมูลในแพกเก็ตโพรโทคอลทีซีพีไอพี โดยมีการทำงานดังรูป



ภาพที่ 19 ภาพรวมการดำเนินการและการใช้งานภาษาจำเพาะโดเมน

จากภาพที่ 19 สามารถอธิบายกระบวนการการใช้งานภาษาจำเพาะโดเมนในการเขียนกฎในการตรวจจับรูปแบบการบุกรุกในเครือข่ายได้ โดยเริ่มต้นจากไฟล์ภาษาจำเพาะโดเมนจะถูกเขียนขึ้นโดยผู้เชี่ยวชาญที่สามารถเข้าไป เขียนเพิ่ม หรือแก้ไขได้ทันทีเมื่อต้องการสร้างกฎหรือพฤติกรรมของรูปแบบการบุกรุกและเมื่อกฎที่ได้ผ่านการตรวจสอบความถูกต้องแล้วจึงนำเข้าสู่ระบบและข้อมูลและกฎที่อยู่ในภาษาจำเพาะโดเมนด้วยตัวแจงส่วนที่สร้างขึ้น ซึ่งจะใช้ในการทำงานของระบบตรวจจับการบุกรุกตามกฎที่ได้นิยามไว้โดยผู้เชี่ยวชาญ

3.2 แนวคิดในการพัฒนาวากยสัมพันธ์ของอีสดีเอสแอล

การแจงส่วน (Parsing) เป็นกระบวนการเพื่อการได้มาของสตริงที่ถูกสร้างออกมาจากวากยสัมพันธ์ของภาษา ซึ่งการสร้างวากยสัมพันธ์ของอีสดีเอสแอลใช้แนวความคิดตามลักษณะของชั้นของทีซีพี/ไอพี โดยในการสร้างวากยสัมพันธ์คือจะแบ่งการเขียนเงื่อนไขของค่าคุณลักษณะของอี

สดีเอสแอลออกเป็นชั้นๆตามชั้นของทีซีพี/ไอพี เพื่อให้ผู้เขี่ยวชาญด้านเครือข่ายคอมพิวเตอร์สามารถเข้าใจได้ง่าย โดยในอีสดีเอสแอลได้กำหนดขอบเขตของวากยสัมพันธ์ในในชั้นต่างๆ ดังนี้ดังต่อไปนี้

1. ส่วนหัวของอีเทอร์เน็ตเฟรม (Ethernet frame header)
2. ส่วนหัวของอาร์พแพกเก็ต (Arp packet header)
3. ส่วนหัวของไอพีดาตาแกรม (IP datagram header)
4. ส่วนหัวของทีซีพีแพกเก็ต (TCP packet header)
5. ส่วนหัวของยูดีพีแพกเก็ต (UDP packet header)
6. ส่วนหัวของดีเอนเอสแพกเก็ต (DNS packet header)

วากยสัมพันธ์ของอีสดีเอสแอลจะจะอ้างอิงตามค่าคุณลักษณะของแพกเก็ต ซึ่งในแต่ละชั้นจะมีคุณลักษณะดังตารางที่ 2, ตารางที่ 3, ตารางที่ 4, ตารางที่ 5, ตารางที่ 6,และตารางที่ 7

ตารางที่ 2 คุณลักษณะภายในอีเทอร์เน็ตเฟรม

คุณลักษณะ	**คำอธิบาย**
Source Mac address	หมายเลขของการ์ดเครือข่ายของเครื่องต้นทาง (ขนาด 48 บิต)
Destination Mac Address	หมายเลขของการ์ดเครือข่ายของเครื่องปลายทาง (ขนาด 48 บิต)
Protocol type	หมายเลขระบุโพรโทคอลที่ใช้ (ขนาด 8 บิต)

ตารางที่ 3 คุณลักษณะภายในอาร์พแพกเก็ต

คุณลักษณะ	**คำอธิบาย**
Arp Operation	คำสั่งของอาร์พโพรโทคอล (ค่า 1 หมายถึงการร้องขอ, ค่า 2 หมายถึงการตอบกลับ)
Source Hardware address	หมายเลขระบุที่อยู่ของเครื่องต้นทาง (ขนาด 48 บิต)
Destination Hardware address	หมายเลขระบุที่อยู่ของเครื่องปลายทาง (ขนาด 48 บิต)
Source Protocol address	หมายเลขที่อยู่โพรโทคอลของต้นทาง (ขนาด 32 บิต)
Destination Protocol address	หมายเลขที่อยู่โพรโทคอลของปลายทาง (ขนาด 32 บิต)

ตารางที่ 4 คุณลักษณะภายในไอพีดาตาแกรม

คุณลักษณะ	**คำอธิบาย**
Source IP address	หมายเลขไอพีของต้นทาง (ขนาด 32 บิต)
Destination IP address	หมายเลขไอพีของปลายทาง (ขนาด 32 บิต)
Time to Live	เลขระจำนวนครั้งในการส่งต่อระหว่างโนหดที่มากที่สุด โดยในการส่งต่อแต่ละโหนดค่าจะลดลงครั้งละ 1 เมื่อมีค่าเป็นข้อมูลจะถูกยกเลิก หรือไม่ส่งต่อ (ขนาด 8 บิต)
Fragmentation Flag	ค่าที่ระบุการแยกดาตาแกรมในเครือข่าย (ขนาด 3 บิต)
Protocol field	หมายเลขระบุโพรโทคอลที่ใช้ในดาตาแกรม (ขนาด 8 บิต)
Total length	ค่าความยาวสูงสุดของดาตาแกรม (ขนาด 16 บิต)
Version	เวอร์ชันของไอพีดาตาแกรม (ขนาด 4 บิต)
Identification	หมายเลขระบุไอดีของดาตาแกรม (ขนาด 16 บิต)

ตารางที่ 5 คุณลักษณะภายในทีซีพีแพกเก็ต

คุณลักษณะ	**คำอธิบาย**
Source Port	หมายเลขทีซีพีพอร์ตของผู้ส่ง (ขนาด 16 บิต)
Destination Port	หมายเลขทีซีพีพอร์ตของผู้รับ (ขนาด 16 บิต)
Acknowledgement number	หมายเลขระบุเลขที่การตอบกลับ (ขนาด 32 บิต) เพื่อใช้ในการตรวจสอบกับค่าหมายเลขลำดับการส่งข้อมูลของผู้รับ
Sequence Number	หมายเลขระบุลำดับการส่งข้อมูลของผู้รับและผู้ส่งในการส่งข้อมูลแต่ะครั้ง (ขนาด 32 บิต)
Next sequence number	หมายเลขระบุหมายเลขลำดับของข้อมูลถัดไป (ขนาด 32 บิต)
Window size	ขนาดของหน้าต่างที่ส่งข้อมูล (ขนาด 16 บิต)
FIN flag	แฟล็กที่แสดงว่าข้อมูลในฝั่งผู้ส่งหมดแล้ว (ขนาด 1 บิต)
SYN flag	แฟล็กที่แสดงว่าข้อมูลที่ส่งเป็นข้อมูลเริ่มต้นการเชื่อมต่อเพื่อประสานหมายเลขลำดับให้สอดคล้อง (ขนาด 1 บิต)
RST flag	แฟล็กที่แสดงการยกเลิกการเชื่อมต่อ (ขนาด 1 บิต)

PSH flag	แฟล็กที่ใช้ถามการเพื่อ push ไปยังบัฟเฟอร์ของแอปพลิเคชันปลายทาง (ขนาด 1 บิต)
ACK flag	แฟล็กที่แสดงการรับทราบจากผู้รับ (ขนาด 1 บิต)
ECE flag	แฟล็กที่แสดงว่าเป็นแพกเก็ต echo (ขนาด 1 บิต)
CWR flag	แฟล็กที่สื่อสารในกระบวนการลดความแออัดในเครือข่าย (ขนาด 1 บิต)
URG flag	แฟล็กที่แสดงข้อมูลเร่งด่วน หรือมีข้อมูลพิเศษ (ขนาด 1 บิต)
NS flag	แฟล็กที่แสดงว่ามีการป้องกันหรือปกปิดข้อมูล (ขนาด 1 บิต)
Reserved field	พื้นที่ที่สงวนไว้สำหรับอนาคต (ขนาด 3 บิต)

ตารางที่ 6 คุณลักษณะภายในยูดีพีแพกเก็ต

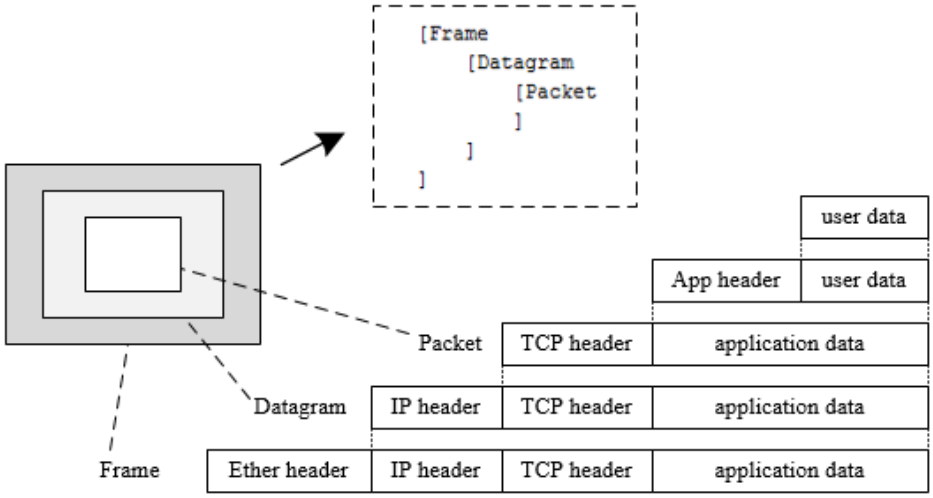
คุณลักษณะ	**คำอธิบาย**
Source port	หมายเลขพอร์ตยูดีพีของต้นทาง (ขนาด 16 บิต)
Destination port	หมายเลขพอร์ตยูดีพีของปลายทาง (ขนาด 16 บิต)
Packet Length	ความยาวของยูดีพีแพกเก็ต (ขนาด 16 บิต)

ตารางที่ 7 คุณลักษณะภายในดีเอนเอสแพกเก็ต

คุณลักษณะ	**คำอธิบาย**
DNS identification	หมายเลขไอดีของดีเอนเอสแพกเก็ต
DNS operation	คำสั่งของดีเอนเอสแพกเก็ต

การออกแบบวากยสัมพันธ์ของอีสดีเอสแอลเป็นไปตามโครงสร้างแบบลำดับชั้นของโพรโทคอลทีซีพี/ไอพี ซึ่งอ้างอิงตามการหลักการห่อหุ้มแคปซูล และการแกะแคปซูลดังภาพที่ 20 เพื่อให้ผู้เชี่ยวชาญสามารถเข้าใจได้ง่าย นอกจากนี้การนิยามเทอร์มินอล (Terminal) เพื่อสื่อความหมายแทนคุณลักษณะของแพกเก็ตในวากยสัมพันธ์ของอีสดีเอสแอลได้มีการรวมคุณลักษณะ Source Hardware address และ Destination Hardware address ในส่วนหัวของชั้นอาร์พแพกเก็ตเข้าไปอยู่ในส่วนหัวของชั้นอีเทอร์เน็ตเฟรม และคุณลักษณะ Source Protocol address และ

Destination Protocol address ในส่วนหัวของชั้นอาร์พแพกเก็ตเข้าไปในไอพีดาตาแกรมเพื่อลดความซ้ำซ้อนของเทอร์มินอลซึ่งแสดงในตารางที่ 8



ภาพที่ 20 โครงสร้างแบบลำดับชั้นของโพรโทคอลทีซีพี/ไอพี

ตารางที่ 8 การนิยามเทอร์มินอลของอีสดีเอสแอลที่แบ่งตามลำดับชั้น

เทอร์มินอล	**ลำดับชั้นของคุณลักษณะ**	**แทนความหมาย**
srcMac	อีเทอร์เน็ตเฟรม	Source Mac address
desMac	อีเทอร์เน็ตเฟรม	Destination Mac Address
protocolType	อีเทอร์เน็ตเฟรม	Protocol type
arpOp	อีเทอร์เน็ตเฟรม, อาร์พแพกเก็ต	Arp Operation
srcIP	ไอพีดาตาแกรม	Source IP address
desIP	ไอพีดาตาแกรม	Destination IP address
ttl	ไอพีดาตาแกรม	Time to Live
fragment	ไอพีดาตาแกรม	Fragmentation Flag
protocol	ไอพีดาตาแกรม	Protocol field
length	ไอพีดาตาแกรม, ยูดีพีแพกเก็ต	Total length
version	ไอพีดาตาแกรม	Version
id	ไอพีดาตาแกรม	Identification
srcPort	ทีซีพีแพกเก็ต, ยูดีพีแพกเก็ต	Source Port
desPort	ทีซีพีแพกเก็ต, ยูดีพีแพกเก็ต	Destination port

ackNumber	ทีซีพีแพกเก็ต	Acknowledgement number
seqNumber	ทีซีพีแพกเก็ต	Sequence Number
nextSeqNunber	ทีซีพีแพกเก็ต	Next sequence number
winSize	ทีซีพีแพกเก็ต	Window size
fin	ทีซีพีแพกเก็ต	FIN flag
syn	ทีซีพีแพกเก็ต	SYN flag
rst	ทีซีพีแพกเก็ต	RST flag
psh	ทีซีพีแพกเก็ต	PSH flag
ack	ทีซีพีแพกเก็ต	ACK flag
ece	ทีซีพีแพกเก็ต	ECE flag
cwr	ทีซีพีแพกเก็ต	CWR flag
urg	ทีซีพีแพกเก็ต	URG flag
res	ทีซีพีแพกเก็ต	Reserved field
dnsID	ยูดีพีแพกเก็ต, ดีเอนเอสแพกเก็ต	DNS identification
dnsOp	ยูดีพีแพกเก็ต, ดีเอนเอสแพกเก็ต	DNS Operation

โครงสร้างในการเขียนอีสดีเอสแอลจะถูกแบ่งด้วยเทอมชั้นของทีซีพี/ไอพี โดยมีเทอร์มินอลที่ใช้ในการแบ่งชั้นดังตารางที่ 9

ตารางที่ 9 เทอร์มินอลที่ใช้ในการแบ่งชั้นของทีซีพี/ไอพี

โหนด	**ความหมาย**	**ลำดับชั้นของเทอมที่สูงกว่า**
Rule	กฎของอีสดีเอสแอล	ไม่มี
Frame	อีเทอร์เน็ตเฟรม	กฎของอีสดีเอสแอล
Datagram	ดาตาแกรม	อีเทอร์เน็ตเฟรม
TCP_Packet	ทีซีพีแพกเก็ต	ทีซีพีแพกเก็ต
UDP_Packet	ยูดีพีแพกเก็ต	ยูดีพีแพกเก็ต
DNS_Packet	ดีเอ็นเอสแพกเก็ต	ดีเอ็นเอสแพกเก็ต

ซึ่งชั้นที่ต่ำกว่าจะถูกห่อหุ้มด้วยเทอมของชั้นที่สูงกว่าและมีอักขระ “[” และ “]” เป็นอักขระที่ใช้บอกขอบเขตของเทอร์มินอลคุณลักษณะในชั้นนั้นๆ ตัวอย่างในภาพที่ 21

```
[Frame
     [Datagram
     ]
]
```

ภาพที่ 21 ตัวอย่างการเขียนเทอมและขอบเขตในแต่ละชั้น

ความหมายของภาพที่ 21 คือโหนด Frame ซึ่งหมายถึงชั้นอีเทอร์เน็ตเฟรมเป็นชั้นที่ห่อหุ้มโหนด Datagram ซึ่งหมายถึงชั้นดาตาแกรม ในแต่ละชั้นจะสามารถเขียนเงื่อนไขของคุณลักษณะที่อยู่ภายในลำดับชั้นของคุณลักษณะนั้นๆตามโหนดที่กำหนด เช่น ในชั้นดาตาแกรมจะสามารถเขียนเงื่อนไขโดยใช้คุณลักษณะต่างๆที่อยู่ในชั้นดาตาแกรมได้ เช่น srcIP, desIP, version เป็นต้น

การออกแบบวากยสัมพันธ์ของโหนดในอีสดีเอสแอลจะออกแบบให้มีโครงสร้างโดยรวมของกฎดังภาพที่ 22

```
[Rule + “RuleName” + IdentificationList
       [Layer 1
              Condition1-1,
              Condition1-2,
              Condition1-3,
              …
             [Layer 2
                  Condition2-1,
                  Condition2-2,
                  Condition2-3,
                  …
                  [Layer N
               …
                  ]
             ]
      ]
]
```

ภาพที่ 22 โครงสร้างวากยสัมพันธ์โดยรวมของกฎ

โครงสร้างกฎเริ่มต้นด้วยโหนด Rule ตามด้วยชื่อของกฎ (RuleName) และตัวแปรต่างๆที่ประกาศใช้ (IdentificationList) โดยตัวแปรจะถูกใช้ในการเขียนเงื่อนไข (Condition) ของค่าคุณลักษณะในชั้นนั้น ซึ่งในแต่ละชั้นจะสามารถเขียนเงื่อนไขของคุณลักษณะที่อยู่ภายในลำดับชั้นตามโหนดที่กำหนด เช่น Condition1-1, Condition1-2 และ Condition1-3 อยู่ในชั้น Frame จะสามารถเขียนเงื่อนไขได้ดังตัวอย่างในภาพที่ 23

```
[Rule “Frame Attack” (a,b,c)
    [Frame
        a.desMac  = “FF:FF:FF:FF FF:FF”,
        b.srcMac != “F0:AC:19:28:55:BC”,
        c.desMac  = b.desMac
    ]
]
```

ภาพที่ 23 ตัวอย่างการเขียนกฎในชั้น Frame

จากภาพที่ 23 แสดงกฎชื่อ “Frame Attack” และประกาศตัวแปรที่ใช้คือ ตัวแปร a , ตัวแปร b และตัวแปร c โดยการออกแบบวากยสัมพันธ์ของเงื่อนไขในอีสดีเอสแอลจะมีรูปแบบคือ เขียนตัวแปร ตามด้วยอักขระ “.” แล้วตามด้วยเทอร์มินอลของคุณลักษณะที่จะตรวจสอบนั้น โดยอาจเขียนโหนดของค่าคุณลักษณะที่จะตรวจสอบกับค่าของคุณลักษณะนั้น เช่น a.desIP = “127.0.0.1” หรือเขียนเพื่อเป็นเงื่อนไขเพื่อเปรียบเทียบกับคุณลักษณะต่างตัวแปรกัน เช่น a.srcIP = b.desIP เป็นต้น โดยการเขียนเงื่อนไขจะใช้สัญลักษณะเครื่องหมายเปรียบเทียบคือสัญลักษณ์ “=” และสัญลักษณ์ “!=”

การสร้างตัวแปรในอีสดีเอสแอลมีเพื่อเพิ่มความขีดสามารถของภาษาให้สามารถอธิบายความสัมพันธ์ของคุณลักษณะที่อยู่ระหว่างระหว่างแพกเก็ต (Across packets) ได้ ซึ่งทำให้อีสดีเอสแอลสามารถใช้ในการอธิบายรูปแบบการบุกรุกที่เกิดจากกลุ่มของแพกเก็ตได้ โดยความหมายของตัวแปรในอีสดีเอสแอลจะแทนความหมายของแพกเก็ตหนึ่งๆที่ตรวจสอบเช่น ตัวแปร a หมายถึงการกำหนดแพกเก็ตที่สนใจชื่อ “a” , ตัวแปร b หมายถึงการกำหนดแพกเก็ตที่สนใจชื่อ “b” เช่นเดียวกับคุณลักษณะที่ตามหลังตัวแปรจะแทนความหมายของค่าคุณลักษณะของแพกเก็ตที่สนใจนั้น เช่น a.srcIP = “127.0.0.1” หมายถึงแพกเก็ตที่สนใจชื่อ a มีค่าคุณลักษณะไอพีต้นทาง เท่ากับค่า “127.0.0.1” , b.srcIP != c.desIP หมายถึง หมายถึงแพกเก็ตที่สนใจชื่อ “b” มีค่าคุณลักษณะไอพีต้นทางไม่เท่ากับค่าคุณลักษณะไอพีปลายทางของแพกเก็ตที่สนใจชื่อ “c” เป็นต้น นอกจากนี้อีสดีเอสแอลได้กำหนดสัญลักษณ์เครื่องหมายเฉพาะสำหรับการเปรียบเทียบค่าคุณลักษณะไอพีแอดเดรสในขั้นไอพีดาตาแกรมโดยเฉพาะคือสัญลักษณ์ “in” และสัญลักษณ์ “!in” เพื่อใช้ในการอธิบายหมายเลขไอพีแอดเดรสว่าอยู่ในช่วงค่าของหมายเลขแอดเดรสเครือข่ายหรือไม่ โดยค่าของคุณลักษณะที่เปรียบเทียบสามารถกำหนด subnet mask ของหมายเลขแอดเดรสเครือข่ายหลังจากหมายเลขแอดเดรสเครือข่ายได้เช่น “192.168.0.0/24” หรือ “172.10.0.0/16” โดยตัวอย่างของเงื่อนไขประเภทนี้ เช่น a.desIP in “192.168.0.0/24” หมายถึงเงื่อนไขที่หมายเลขไอพีแอดเดรสปลายทางของแพกเก็ตตัวแปร a อยู่ในช่วงหมายเลขเครือข่าย “192.168.0.0”

เนื่องจากการกำหนดโหนดที่ใช้ในการเขียนกฎและเงื่อนไขของไอเอสดีเอสแอลจะถูกแบ่งเป็นเงื่อนไขในชั้นของทีซีพี/ไอพี และเงื่อนไขที่เป็นฟังก์ชันควบคุม โดยมีโหนดต่างๆดังตารางที่ 10

ตารางที่ 10 โหนดที่ใช้ในการสร้างกฎและเงื่อนไขของอีสดีเอสแอล

โหนด	**ความหมาย**
rule	โหนดรากของกฎ
ruleName	โหนดชื่อกฎ
paramList	โหนดรายการตัวแปร
idList	โหนดที่ใช้สร้างรายการตัวแปรแบบวนซ้ำ
conList	โหนดเงื่อนไขทั้งหมด โดยมีทั้งเงื่อนไขในแต่ละชั้น และเงื่อนไขแบบฟังก์ชัน

tcpStackCon	โหนดเงื่อนไขในชั้นทีซีพี/ไอพี
frameCon	โหนดเงื่อนไขในชั้นอีเทอร์เน็ตเฟรม
datagramCon	โหนดเงื่อนไขของชั้นต่างๆภายใต้ชั้นดาตาแกรม
tcpPacketCon	โหนดเงื่อนไขของชั้นต่างๆภายใต้ชั้นทีซีพีแพกเก็ต
udpPacketCon	โหนดเงื่อนไขของชั้นต่างๆภายใต้ชั้นยูดีพีแพกเก็ต
dnsPacketCon	โหนดเงื่อนไขของชั้นต่างๆภายใต้ชั้นดีเอ็นเอสแพกเก็ต
frameConList	โหนดเงื่อนไขในชั้นอีเทอร์เน็ตเฟรม
datagramConList	โหนดเงื่อนไขในชั้นดาตาแกรม
tcpPacketConList	โหนดเงื่อนไขในชั้นทีซีพีแพกเก็ต
udpPacketConList	โหนดเงื่อนไขในชั้นยูดีพีแพกเก็ต
dnsPacketConList	โหนดเงื่อนไขในชั้นดีเอ็นเอสแพกเก็ต
frameConStmt	โหนดของประโยคเงื่อนไขในชั้นอีเทอร์เน็ตเฟรม
datagramConStmt	โหนดของประโยคเงื่อนไขในชั้นดาตาแกรม
tcpPacketConStmt	โหนดของประโยคเงื่อนไขในชั้นทีซีพีแพกเก็ต
udpPacketConStmt	โหนดของประโยคเงื่อนไขในชั้นยูดีพีแพกเก็ต
dnsPacketConStmt	โหนดของประโยคเงื่อนไขในชั้นดีเอ็นเอสแพกเก็ต
packetComparer	โหนดของประโยคเงื่อนไขการเปรียบเทียบแพกเก็ต
biOp	เครื่องหมายที่ใช้ในการเปรียบเทียบค่าคุณลักษณะของแพกเก็ต
inOp	เครื่องหมายที่ใช้ในการเปรียบเทียบค่าคุณลักษณะไอพีแอดเดรส
controlFuncStmt	โหนดของประโยคเงื่อนไขในรูปแบบฟังก์ชัน

ในภาพที่ 24 แสดงตัวอย่างการเขียนวากยสัมพันธ์ของโหนด rule, ruleName, paramList และ idList ซึ่งจะเห็นได้ว่าบางโหนดจะถูกใช้ในการสร้างโหนดอื่น เช่นโหนด paramList จะมีโหนด idList เป็นโหนดย่อย

```
rule:= [ + ruleName + paramList + conList + ]
ruleName := Rule + name;
paramList.Rule := ( + idList + )
idList := [identifier]+
```

ภาพที่ 24 นิยามวากยสัมพันธ์ของโหนด rule, ruleName, paramList และ idList

วากยสัมพันธ์ของโหนด idList ที่ต้องการแจงส่วนตัวแปรของอีสดีเอสแอลซึ่งสามารถประกาศตั้งแต่หนึ่งตัวแปรขึ้นไป จึงใช้ลักษณของการวนซ้ำโดยเริ่มจากหนึ่งของโหนด identifier คือ [identifier]+

ตัวอย่างความสัมพันธ์ของโหนดที่ใช้ในการสร้างเงื่อนไขในแต่ละชั้นโพรโทคอลทีซีพี/ไอพี เช่นในชั้นไอพีดาตาแกรมมีโหนด datagramCon ทหหน้าที่เป็นโหนดที่รวบรวมเงื่อนไขของชั้นต่างๆ ภายใต้ชั้นดาตาแกรม และ datagramConList เป็นโหนดที่รวบรวมโหนดประโยคเงื่อนไข หรือโนหด datagramConStmt ตั้งแต่ 0 โหนดขึ้นไป ดังแสดงในภาพที่ 25

```
datagramCon := datagramConList | datagramConList + [ + TCP_Packet +
tcpPacketCon + ] | datagramConList + [ + TCP_Packet + tcpPacketCon + ] + [ +
UDP_Packet + udpPacketCon + ] | datagramConList + [ + UDP_Packet +
udpPacketCon + ]

datagramConList:= (datagramConStmt)*

datagramConStmt := d_IpCon | d_TtlCon | d_FragmentCon | d_ProtocolCon |
d_LenghtCon | d_VersionCon | d_IdCon;
```

ภาพที่ 25 วากยสัมพันธ์ของโหนด datagramCon , datagramConList และ datagramConStmt

ในงานวิจัยนี้มีการกำหนดฟังก์ชันควบคุม (Control function) ของอีสดีเอสแอล 2 ฟังก์ชัน ได้แก่ 1) ฟังก์ชันนับ (Count function) คือฟังก์ชันนับจะใช้ในการนับจำนวนแพกเก็ตที่มีคุณลักษณะที่ซ้ำกับแพกเก็ตที่แทนด้วยตัวแปรนำเข้าของฟังก์ชัน เปรียบเทียบกับค่าจำนวนแพกเก็ตที่เป็นเงื่อนไข และจะให้ค่าผลลัพธ์เป็นบูลีนที่บอกว่าจำนวนแพกเก็ตที่มีคุณลักษณะตรงกับคุณลักษณะของแพกเก็ตที่แทนด้วยตัวแปรนำเข้านั้นเท่ากับค่าจำนวนแพกเก็ตที่เป็นเงื่อนไขหรือไม่ เช่น Count(a,100) หมายถึงนับจำนวนแพกเก็ตที่มีคุณลักษณะตรงกับคุณลักษณะของแพกเก็ต a ว่ามีค่าเท่ากับ100 หรือไม่และ 2) ฟังก์ชันลำดับ (Sequence function) คือฟังก์ชันที่ทำหน้าที่ตรวจสอบลำดับก่อนหลังของแพกเก็ตที่แทนด้วยตัวแปรนำเข้าโดยเรียงลำดับจากลำดับของตัวแปรนำเข้าทั้งหมด ซึ่งค่าผลลัพธ์จะเป็นค่าบูลีนที่เปรียบเทียบลำดับว่าตรงกับลำดับของตัวแปรนำเข้าทั้งหมดหรือไม่ เช่น Sequence(b,a,c) หมายถึงแพกเก็ต b มีลำดับมาก่อน แพกเก็ต a และแพกเก็ต c เป็นต้น โดยโหนดที่ทำหน้าที่รวบรวมเงื่อนไขที่มีลักษณะแบบฟังก์ชันควบคุมนี้คือโหนด controlFuncStmt ตามภาพที่ 26 แสดงการนิยามวากยสัมพันธ์ของโหนด controlFuncStmt

```
controlFuncStmt := countFunction | seqFunction | packetComparer
countFunction := Count + ( + identifier + comma + number + )
seqFunction := Sequence + ( + seqParamList + );
seqParamList := (identifier)+
```

ภาพที่ 26 นิยามวากยสัมพันธ์ของโหนด controlFuncStmt

โดยประโยคฟังก์ชัดควบคุมจะดำรงอยู่ในขอบเขตของโหนด Rule ซึ่งอยู่นอกลำดับชั้นทีซีพี/ไอพีดังภาพที่ 27

```
[Rule + "RuleName" + IdentificationList
       [Layer 1
              Condition1-1,
              Condition1-2,
      ]
       controlFuncStmt1
       controlFuncStmt2
       controlFuncStmt3
]
```

ภาพที่ 27 นิยามวากยสัมพันธ์ของโหนด controlFuncStmt

ตัวอย่างการใช้งานฟังก์ชันควบคุม Count() ของกฎ “Syn flooding” ตามภาพที่ 28

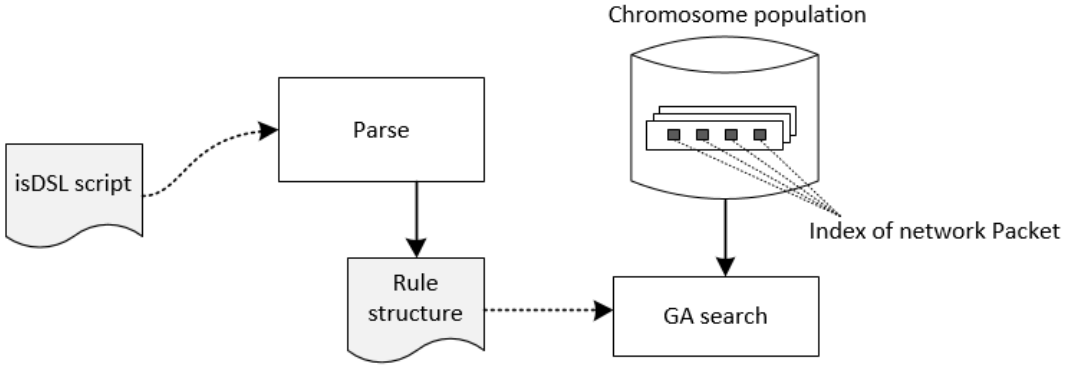
```
[Rule  "Syn flooding" (a)
    [TCP_Packet
         a.syn = 1
    ]
    Count(a,30)
]
```

ภาพที่ 28 กฎการตรวจจับการบุกรุกเครือข่าย “Syn flooding”

เงื่อนไข Count(a,30) ในภาพหมายถึงนับจำนวนแพกเก็ตที่มีคุณลักษณะ flag syn ในชั้นทีซีพีที่มีค่าเท่ากับ 1 นั้นมีจำนวนเท่ากับ 30 แพกเก็ตหรือไม่

3.3 แนวคิดในการพัฒนาระบบตรวจจับการบุกรุกของอีสดีเอสแอล

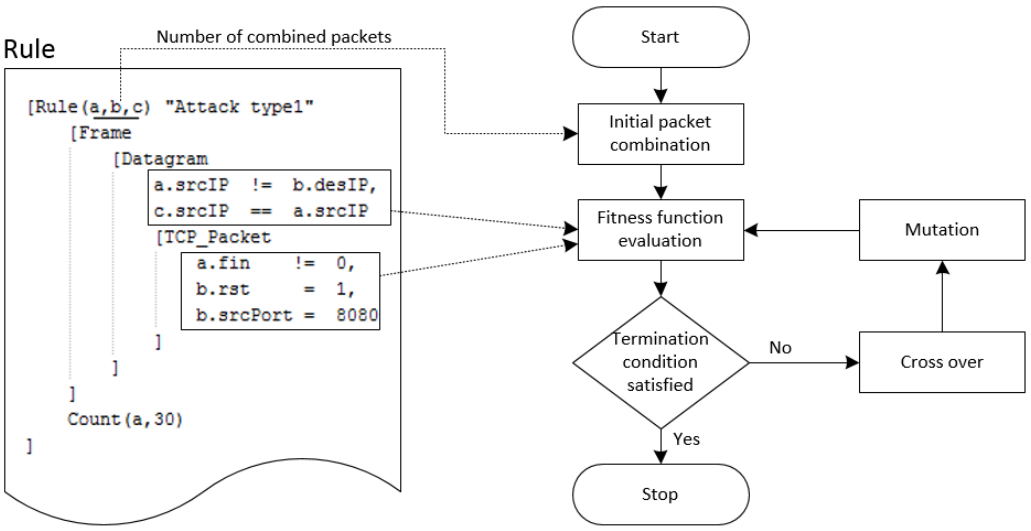
การออกแบบระบบตรวจจับการบุกรุกของอีสดีเอสแอล มีแนวคิดในการประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรมในการค้นหากลุ่มของแพกเก็ต (Combination of packets) ที่ถูกนิยามโดยผู้เชี่ยวชาญด้วยอีสดีเอสแอล โดยกลุ่มของแพกเก็ตที่เป็นสมมติฐานของปัญหาจะถูกเข้ารหัสเป็นชุดโครโมโซมเพื่อใช้ในการค้นหาด้วยขั้นตอนวิธีเชิงพันธุกรรม โดยรหัสพันธุกรรมภายในโครโมโซมจะใช้เพื่อแทนค่าตำแหน่งอ้างอิงของแพกเก็ตที่อยู่ในขอบเขตของการค้นหา



ภาพที่ 29 การทำงานของระบบตรวจจับการบุกรุกใช้โครงสร้างของกฎ

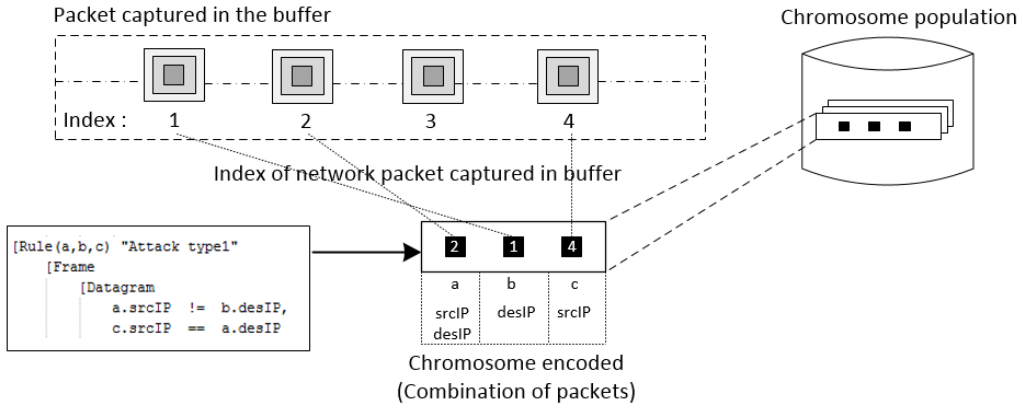
การทำงานของระบบตรวจจับการบุกรุกจะใช้โครงสร้างของกฎ (Rule structure) ในภาพที่ 29 ที่ได้จากการแจงส่วนอีสดีเอสแอลสคริปต์ ในการตรวจสอบค่าคุณลักษณะของแพกเก็ตเครือข่ายที่ตรงกับกฎที่ได้นิยามไว้หรือไม่ ซึ่งการทำงานของขั้นตอนวิธีเชิงพันธุกรรมจะอยู่บนพื้นฐานของวากยสัมพันธ์ของอีสดีเอสแอลในการควบคุมในแต่และขั้นตอนการดำเนินการของขั้นตอนวิธีเชิงพันธุกรรม (ภาพที่ 30) ดังนี้

1. จำนวนตัวแปรของอีสดีเอสแอลจะใช้ในการกำหนดจำนวนของแพกเก็ตในกลุ่มแพกเก็ตที่เป็นรูปแบบพฤติกรรมการบุกรุก
2. เงื่อนไขข้อกำหนดของอีสดีเอสแอลจะเป็นตัวกำหนดการทำงานของฟังก์ชันความเหมาะสมในการประเมินและให้คะแนนความคล้ายคลึงของโครโมโซมหรือกลุ่มของแพกเก็ตที่ตรงกับเงื่อนไขของภายในโครงสร้างของกฎ



ภาพที่ 30 ความสัมพันธ์ระหว่างขั้นตอนวิธีเชิงพันธุกรรมวากยสัมพันธ์ของอีสดีเอสแอล

โดยในการเข้ารหัสโครโมโซมของขั้นตอนวิธีเชิงพันธุกรรม จะใช้ตำแหน่งอ้างอิงของแพกเก็ตที่อยู่ในขอบเขตการค้นหาเป็นรหัสพันธุกรรมโดยจะเรียงตัวตามลำดับของตัวแปรในอีสดีเอสแอล ตัวอย่างในภาพที่ 31 แสดงตัวอย่างกฎการตรวจจับการบุกรุกชื่อ “Attack type1” มีตัวแปร 3 ตัวคือ a, b และ c การเข้ารหัสโครโมโซมหนึ่งๆ จะทำโดยสุ่มค่าอ้างอิงตำแหน่งของแพกเก็ต (Index refer to packet) ในบัฟเฟอร์ที่ดักจับมาจากกเครือข่ายมาเป็นรหัสพันธุกรรม โดยในหนึ่งโครโมโซมจะมีจำนวนรหัสพันธุกรรมเท่ากับจำนวนจำนวนตัวแปรของอีสดีเอสแอลคือ 3



ภาพที่ 31 การเข้ารหัสโครโมโซมของอีสดีเอสแอล

ซึ่งการตรวจสอบเงื่อนไขต่างๆในไอเอสดีเอสแอลในการค้นการการบุกรุกจะทำโดยเปรียบเทียบค่าของคุณลักษณะที่กำหนดอยู่ในเงื่อนไขกับค่าคุณลักษณะของแพกเก็ตที่ถูกอ้างอิงถึงตามตำแหน่งอ้างอิงที่อยู่ในโครโมโซม เช่นจากภาพแพกเก็ตตำแหน่งอ้างอิงที่ 2 ถูกอ้างอิงด้วยตัวแปร a โดยมีค่าคุณลักษณะที่ต้องตรวจสอบคือค่าหมายเลขไอพีต้นทาง และค่าหมายเลขไอพีปลายทางเป็นต้น

3.3 ความแตกต่างระหว่าง isDSL และ snort

Snort เป็นระบบตรวจจับการบุกรุกที่เปิดเผยต้นฉบับและเป็นที่นิยมใช้มากในปัจจุบัน เนื่องจากเป็นระบบตรวจจับการบุกรุกที่ใช้ทรัพยากรของระบบน้อยเนื่องจากอาศัยขั้นตอนวิธีการจับคู่แบบหลายแพทเทิร์น (multi-pattern matching algorithm) ทำให้สามารถตรวจจับรูปแบบพฤติกรรมการบุกรุกหลายรูปแบบได้ในครั้งเดียวโดยไม่กระทบต่อประสิทธิภาพการทำงานของระบบ โดยขั้นตอนวิธีที่ใช้ในปัจจุบันคือ Aho-Corasick algorithm

โดยกฎการตรวจจับการบุกรุกเครือข่ายของ snort จะมีลักษณะเป็นกฎอย่างง่ายที่ใช้ในการอธิบายรูปแบบพฤติกรรมการบุกรุกของแพกเก็ตหนึ่งๆ โดยการกำหนดเงื่อนไขของแพกเก็ตที่ทำการตรวจสอบและกำหนดการกระทำ (action) ของ snort ที่จะตอบสนองเมื่อตรวจจับพบรูปแบบการบุกรุกที่ตรงกับกฎนั้นๆ นอกจากนี้ snort ยังมีการพัฒนาส่วนที่เรียกว่า pre-processor ซึ่งเป็นส่วนที่ทำให้โปรแกรมเมอร์สามารถเขียนกฎที่มีลักษณะเฉพาะและมีความสามารถในการตรวจจับมากยิ่งขึ้น โดยกฎในส่วนนี้จะทำงานหลังจาก snort ถอดรหัสแพกเก็ตเครือข่ายแล้วจึงนำไปตรวจสอบด้วย pre-processor เช่น Stream4 pre-processor, Flow pre-processor, HTTP pre-processor, FTP/Telnet pre-processor และ SMTP pre-processor เป็นต้น โดย Stream4 pre-processor สามารถทำให้ snort ตรวจจับการบุกรุกด้วยการทวบสอบแพกเก็ตต่างๆในแต่ละเซสชั่น (session) ได้ สำหรับ Flow pre-processor ทำให้กฎของ snort สามารถใช้งาน flowbit ในการตรวจสอบรูปแบบการบุกรุกที่เกิดจากหลายกฎพร้อมกันได้ โดยในการเขียนกฎสามารถให้กฎกำหนดเงื่อนไขในการเปลี่ยนแปลงค่า flowbit ได้ เพื่อให้กฎอื่นที่ใช้งาน flowbit ใช้เป็นเงื่อนไขในการตรวจจับการบุกรุกเครือข่าย ตัวอย่างการใช้งาน flowbit ในการอธิบายความสัมพันธ์ระหว่างกฎแสดงดังภาพที่ 32

```
1   alert tcp any 143 -> any any (msg:"IMAP login";
2        content:"OK LOGIN"; flowbits:set,logged_in;
3        flowbits:noalert;)
4
5    alert tcp any any -> any 143 (msg:"IMAP LIST"; content:"LIST";
6        flowbits:isset,logged_in;)
```

ภาพที่ 32 ตัวอย่างกฎ snort ที่มีการเรียกใช้งาน flowbit

จากภาพที่ A ประกอบด้วย 2 กฎการตรวจจับที่จะแจ้งเตือนผู้ใช้งานกรณีที่ตรวจพบการบุกรุก สำหรับกฎแรกมีเงื่อนไขในการกำหนดค่า flowbit โดยใช้คำสั่ง “set” ตามด้วยชื่อ flowbit ในกรณีที่แพกเก็ตตรงกับเงื่อนไขที่กำหนดและมีการเรียกใช้ชื่อ flowbit ตามด้วยคำสั่ง “noalert” ทำให้กฎแรก (บรรทัดที่ 1) จะไม่แจ้งเตือนผู้ใช้ในกรณีที่แพกเก็ตตรงกับเงื่อนไข ในขณะที่กฎที่ 2 (บรรทัดที่ 5) เป็นกฎที่ใช้งาน flowbit มาเป็นเงื่อนไขในการตรวจจับโดยใช้คำสั่ง “isset” ตามด้วย

ชื่อ flowbit ที่เคยเรียกใช้งาน ซึ่งหาก flowbit ถูกกำหนดค่าไว้แล้วในกฎแรก จะทำให้กฎที่สองแจ้งเตือนว่ามีการบุกรุก ภายใต้เงื่อนไขว่ากฎแรกต้องถูกตรวจพบก่อน

การใช้งาน Stream4 pre-processor (flowbit) ทำให้ snort สามารถอธิบายรูปแบบการบุกรุกเครือข่ายที่เกิดขึ้นจากหลายแพกเก็ตได้ อย่างไรก็ตาม การอธิบายความสัมพันธ์จะเป็นในลักษณะความสัมพันธ์ระหว่างกฎมากกว่าความสัมพันธ์ระหว่างแพกเก็ต โดยหากเปรียบเทียบการทำงานของ flowbit กับการอธิบายความสัมพันธ์ระหว่างค่าคุณลักษณะที่อยู่ระหว่างแพกเก็ตของอีสดีเอสแอลแล้ว กฎของ snort ยังไม่สามารถอธิบายความสัมพันธ์ที่ซับซ้อนระหว่างแพกเก็ตได้ เช่น “a.srcIP != b.srcIP” (หมายเลขไอพีต้นทางของแพกเก็ต a ไม่เท่ากับหมายเลขไอพีต้นทางของแพกเก็ต b) หรือ “a.desPort = b.desPort” (หมายเลขพอร์ตปลายทางของแพกเก็ต a เท่ากับค่าหมายเลขพอร์ตปลายทางของแพกเก็ต b) ในสคริปต์อีสดีเอสแอล เป็นต้น นอกจากนี้ snort ยังไม่สามารถอธิบายความสัมพันธ์ที่มีเวลาหรือลำดับของแพกเก็ตมาเป็นเงื่อนไขในการตรวจจับการบุกรุกได้ ข้อจำกัดอีกประการหนึ่งของ snort คือ การเขียนกฎโดยเรียกใช้งาน pre-processor ดังภาพที่ B ผู้ใช้งานจำเป็นต้องมีความรู้เรื่องการโปรแกรม ซึ่งแตกต่างจากอีสดีเอสแอลที่ผู้เชี่ยวชาญสามารถเข้าใจได้ง่ายโดยไม่จำเป็นต้องมีความรู้เรื่องการโปรแกรม

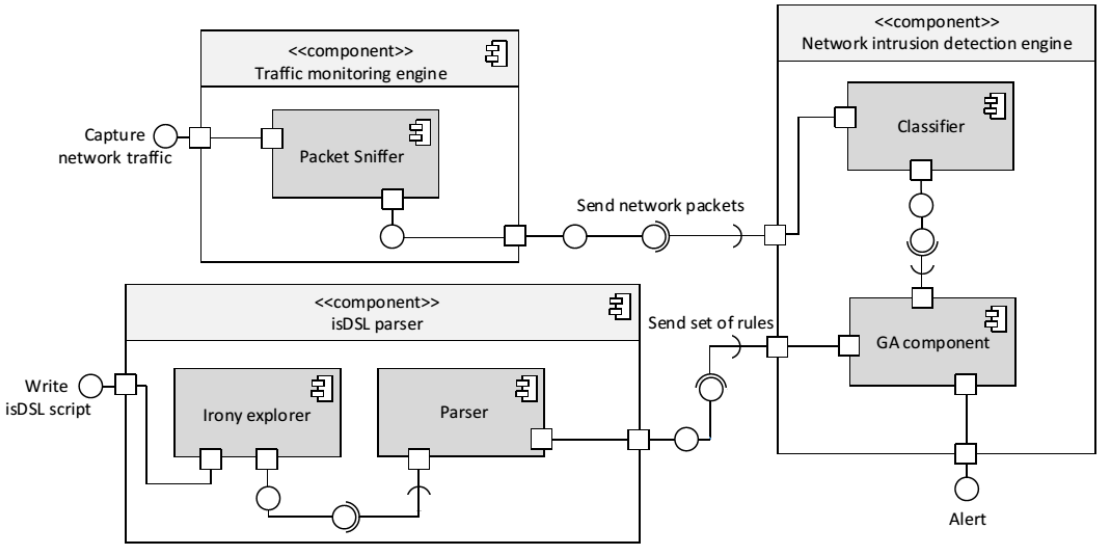
```
#define SRC_PORT_MATCH  1
#define SRC_PORT_MATCH_STR "example_preprocessor: src port match"
#define DST_PORT_MATCH  2
#define DST_PORT_MATCH_STR "example_preprocessor: dest port match"
void ExampleProcess(void *pkt, void *context)
{
    SFSnortPacket *p = (SFSnortPacket *)pkt;
    if (!p->ip4_header || p->ip4_header->proto != IPPROTO_TCP || !p->tcp_header)
    {
        /* Not for me, return */
        return;
    }

    if (p->src_port == portToCheck)
    {
        /* Source port matched, log alert */
        _dpd.alertAdd(GENERATOR_EXAMPLE, SRC_PORT_MATCH,
                      1, 0, 3, SRC_PORT_MATCH_STR, 0);
        return;
    }

    if (p->dst_port == portToCheck)
    {
        /* Destination port matched, log alert */
        _dpd.alertAdd(GENERATOR_EXAMPLE, DST_PORT_MATCH,
                      1, 0, 3, DST_PORT_MATCH_STR, 0);
        return;
    }
}
```

ภาพที่ 33 ตัวอย่างกฎการตรวจจับการบุกรุกแบบ pre-processor

บทที่ 4
การออกแบบและพัฒนาระบบ

4.1 สถาปัตยกรรมระบบ

สถาปัตยกรรมระบบที่พัฒนาขึ้นแบ่งออกเป็นสามส่วนหลัก คือ ตัวแจงส่วนอีสดีเอสแอล ตัวตรวจจับการบุกรุกเครือข่าย และตัวเฝ้าระวังการจราจรเครือข่าย โดยมีแผนภาพสถาปัตยกรรมระบบดังภาพที่ 34



ภาพที่ 34 สถาปัตยกรรมระบบที่พัฒนา

ซึ่งส่วนแจงส่วนจะเป็นส่วนที่ให้ผู้เชี่ยวชาญสามารถใช้ในการเขียนกฎ และตรวจสอบความผิดพลาดเพื่อให้เงื่อนไขภายในกฎไม่เกิดความซ้ำซ้อน และขัดแย้งกัน ตัวตรวจจับการบุกรุกเครือข่ายทำหน้าที่ในการดักจับแพกเก็ตเครือข่ายเพื่อนำมาวิเคราะห์ด้วยตัวตรวจจับการบุกรุกเครือข่ายที่สร้างขึ้น

4.2 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา

สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒาระบบประกอบด้วยรายการฮาร์ดแวร์และซอฟต์แวร์ดังต่อไปนี้

4.2.1 สภาพแวดล้อม

1. หน่วยประมวลผลอินเทล คอร์ ไอ7-2.80 กิกะเฮิร์ต (CPU Intel Core i72640M 2.80GHz)
2. หน่วยความจำ 8 กิกะไบต์ (8 GB RAM)
3. ฮาร์ดดิสก์ความจุ 250 กิกะไบต์ (250 GB HDD)
4. ระบบปฏิบัติการไมโครซอฟต์วินโดว์8.1 (Microsoft Windows 8.1) แบบ 64 บิต

4.2.2 เครื่องมือที่ใช้ในการพัฒนา

1. ไมโครซอฟต์ วิชวลสตูดิโอ 2010(Microsoft Visual Studio 2010)
2. ไมโครซอฟต์ ดอทเน็ตเฟรมเวิร์ค4.0 (Microsoft.Net Framework 4.0)
3. ไอโรนี 2013_03_10(Irony -.Net Language Workbench 2013_03_10)
4. พีแคบดอทเน็ต 0.10.0 (Pcap.Net 0.10.0)

4.3 การพัฒนาระบบ

4.3.1 การพัฒนาตัวแจงส่วน (Parser)

การพัฒนาตัวแจงส่วนจะใช้เครื่องมือพัฒนาไอโรนี [13] เพื่อสร้างไวยากรณ์ของระบบและตัวแจงส่วน การพัฒนาวายกสัมพันธ์โดยใช้เครื่องมือพัฒนาไอโรนีนั้นสามารถทำได้ง่ายและมีความต่อเนื่องไปยังตัวแจงส่วนรูปแบบวายกสัมพันธ์สำหรับไวยากรณ์ที่ออกแบบไว้ในบทที่ 3 เริ่มต้นด้วยการสร้างคลาสที่สืบทอดมาจากคลาสไวยากรณ์จากไอโรนี (Class Irony.Parsing.Grammar) ดังภาพที่ 35 โดยจะต้องกำหนดชื่อของภาษา หมายเลขเวอร์ชัน รายละเอียดในส่วนหัวของคลาส และค่าบูลีนที่กำหนดความไวต่ออักษรใหญ่เล็กของภาษาที่จะสร้าง

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Irony.Parsing;
using Irony.Interpreter.Ast;

namespace Irony.Samples.isDSL
{
    [Language("isDSL", "0.9", "Intrusion signature grammar")]
    class isDSLGrammar : Grammar
    {
        public isDSLGrammar()
            : base(false) //isDSL is case insensitive
        {

        }
    }
}
```

ภาพที่ 35 การสร้างคลาสที่สืบทอดมาจากคลาสไวยากรณ์จากไอโรนี

จากนั้นประกาศเทอร์มินอลของอีสดีเอสแอลตามเทอร์มินอลของในวากยสัมพันธ์ที่กำหนดไว้ในบทที่ 3 ลงในคลาสไวยากรณ์ที่สร้างขึ้น ดังภาพที่ 36

```
            // Terminal
            var Rule = ToTerm("Rule ");
            var Frame = ToTerm("Frame ");
            var Datagram = ToTerm("Datagram ");
            var TCP_Packet = ToTerm("TCP_Packet ");
            var UDP_Packet = ToTerm("UDP_Packet ");
            var DNS_Packet = ToTerm("DNS_Packet");

            var srcMac = ToTerm("srcMac");
            var desMac = ToTerm("desMac");
            var protocolType = ToTerm("protocolType");
            var arpOp = ToTerm("arpOp");

            var srcIP = ToTerm("srcIP");
            var desIP = ToTerm("desIP");
            var ttl = ToTerm("ttl");
            var fragment = ToTerm("fragment");
            var protocol = ToTerm("protocol");
            var lenght = ToTerm("lenght");
            var version = ToTerm("version");
            var id = ToTerm("id");

            var srcPort = ToTerm("srcPort");
            var desPort = ToTerm("desPort");
            var ackNumber = ToTerm("ackNumber");
            var seqNumber = ToTerm("seqNumber");
            var nexSeqNumber = ToTerm("nexSeqNumber");
            var winSize = ToTerm("winSize");
            var fin = ToTerm("fin");
            var syn = ToTerm("syn");
            var rst = ToTerm("rst");
            var psh = ToTerm("psh");
            var ack = ToTerm("ack");
            var ece = ToTerm("ece");
            var cwr = ToTerm("cwr");
            var urg = ToTerm("urg");
            var res = ToTerm("res");

            var dnsId = ToTerm("dnsId");
            var dnsOp = ToTerm("dnsOp");

            var Count = ToTerm("Count");
            var Sequence = ToTerm("Sequence");
```

ภาพที่ 36 การประกาศเทอร์มินอลของอีสดีเอสแอล

หลังจากนั้นประกาศเทอร์มินอลสำหรับตัวแปรและสัญลักษณ์ดังภาพที่ 37

```
KeyTerm comma = ToTerm(",", "comma");
KeyTerm dot = ToTerm(".", "dot");

var identifier = TerminalFactory.CreateisDSLIdentifier("identifier");
var name = TerminalFactory.CreateisDSLRuleName("name");
var value = TerminalFactory.CreateisDSLString("value");
var number = TerminalFactory.CreateisDSLNumber("number");
```

ภาพที่ 37 การประกาศเทอร์มินอลสำหรับตัวแปรและสัญลักษณ์

โดยการสร้างเทอร์มินอลสำหรับตัวแปรจำเป็นต้องมีการสร้างฟังก์ชันในการระบุเทอร์มินอลที่ไอโรนีไม่ได้รองรับดังตัวอย่างฟังก์ชัน CreateisDSLNumber ในภาพที่ 38

```
public static NumberLiteral CreateisDSLNumber(string name)
{
    NumberLiteral term = new NumberLiteral(name, NumberOptions.IntOnly);
    term.DefaultIntTypes = new TypeCode[] { TypeCode.Int32 };
    return term;
}
```

ภาพที่ 38 ฟังก์ชันในการระบุเทอร์มินอลที่ไอโรนีไม่ได้รองรับ

หลังจากนั้นประกาศโหนดที่กำหนดในบทที่ 3 ตามตัวอย่างในภาพที่ 39

```
// Non terminal
var rule = new NonTerminal("rule");
var ruleName = new NonTerminal("ruleName");
var paramList = new NonTerminal("paramList");
var idList = new NonTerminal("idList");
var conList = new NonTerminal("conList");
var tcpStackCon = new NonTerminal("tcpStackCon");

// TCP_Packet
var tcpPacketConStmt = new NonTerminal("tcpPacketConStmt");
var t_Port = new NonTerminal("t_Port");
var t_Flag = new NonTerminal("t_Flag");
var t_PortCon = new NonTerminal("t_PortCon");
var t_AckNumberCon = new NonTerminal("t_AckNumberCon");
var t_SeqNumberCon = new NonTerminal("t_SeqNumberCon");
var t_NexSeqNumberCon = new NonTerminal("t_NexSeqNumberCon");
var t_WinSizeCon = new NonTerminal("t_WinSizeCon");
var t_FlagCon = new NonTerminal("t_FlagCon");
```

ภาพที่ 39 ตัวอย่างการประกาศโหนดของอีสดีเอสแอล

จากนั้นจึงสร้างไวยากรณ์สำหรับโหนดแต่โหนด เพื่อใช้ในการแจงส่วน โดยบางโหนดอาจเป็นไวยากรณ์ย่อยให้กับโหนดอื่นดังตัวอย่างในภาพที่ 40

```
// Rule
this.Root = rule;
rule.Rule = "[" + ruleName + paramList + conList + "]";
ruleName.Rule = Rule + name;
paramList.Rule = "(" + idList + ")";
idList.Rule = MakeStarRule(idList, comma, identifier);

conList.Rule = tcpStackCon + controlFunctionList;
tcpStackCon.Rule = "[" + Frame + frameCon + "]" | "[" + Datagram +
datagramCon + "]" | "[" + TCP_Packet + tcpPacketCon + "]" | "[" + UDP_Packet
+ udpPacketCon + "]" | "[" + DNS_Packet + dnsPacketCon + "]";
```

ภาพที่ 40 ตัวอย่างไวยากรณ์สำหรับโหนดของอีสดีเอสแอล

ในการประกาศกฎที่ลักษณะแบบสตาร์ โดยสามารถมีโหนดที่ซ้ำกันตั้งแต่ 0 โหนดขึ้นไป จะต้องเรียกใช้ใช้ฟังก์ชัน MakeStarRule โดยระบุโหนดหลัก สัญลักาณ์เชื่อมและโหนดย่อยที่จะวนสร้างซ้ำ โดยในภาพที่ 41 แสดงการประกาศไวยากรณ์ของเงื่อนไขที่ใช้ฟังก์ชัน MakeStarRule

```
frameConList.Rule = MakeStarRule(frameConList, comma, frameConStmt);
            datagramConList.Rule = MakeStarRule(datagramConList, comma,
datagramConStmt);
            tcpPacketConList.Rule = MakeStarRule(tcpPacketConList, comma,
tcpPacketConStmt);
            udpPacketConList.Rule = MakeStarRule(udpPacketConList, comma,
udpPacketConStmt);
            dnsPacketConList.Rule = MakeStarRule(dnsPacketConList, comma,
dnsPacketConStmt);
            controlFunctionList.Rule = MakeStarRule(controlFunctionList,
comma, controlFuncStmt);
```

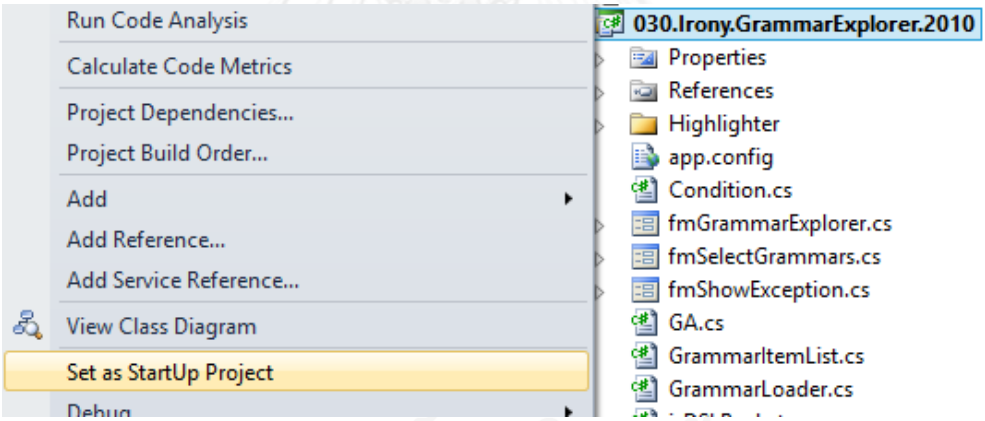
ภาพที่ 41 ตัวอย่างไวยากรณ์ของโหนดแบบสตาร์

สำหรับการกำหนดเทอร์มิทอลที่เป็นเครื่องหมายวรรคตอนหรือคู่สัญลักษณ์จะต้องกำหนดและลงทะเบียนด้วยฟังก์ชันของไอโรนีดังภาพที่ 42

```
            MarkPunctuation("[", "]", "(", ")");
            RegisterBracePair("(", ")");
            RegisterBracePair("[", "]");
```

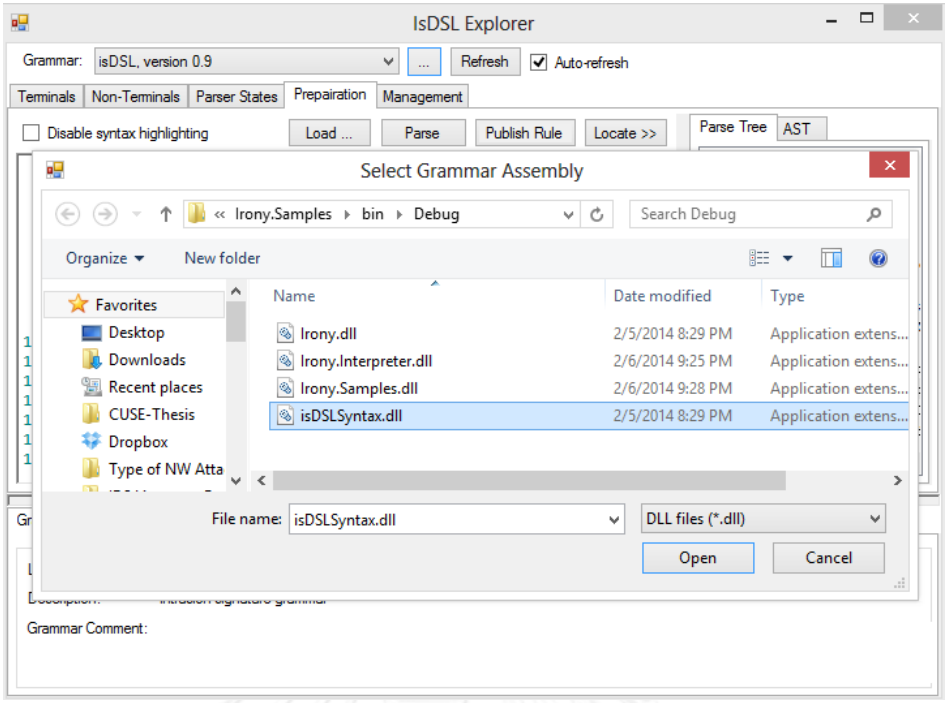
ภาพที่ 42 การกำหนดเทอร์มิทอลที่เป็นเครื่องหมายวรรคตอนหรือคู่สัญลักษณ์

เมื่อสร้างคลาสไวยากรณ์เสร็จแล้ว สามารถตรวจสอบความถูกต้องของไวยากรณ์ที่สร้างขึ้นได้ด้วยโปรแกรมแกรมมาเอ็กพลอเรอร์ของไอโรนีดังภาพที่ 43



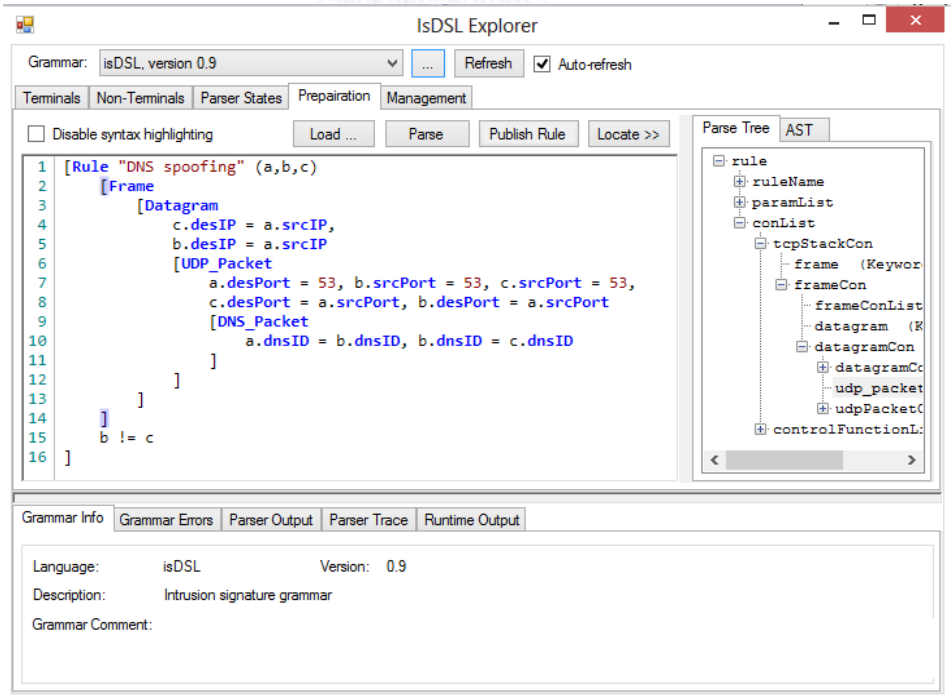
ภาพที่ 43 เลือกแกรมมาเอ็กพลอเรอร์ของไอโรนีเพื่อรันตรวจสอบ

การใช้งานแกรมมาเอ็กพลอเรอร์จะต้องคอมไพล์คลาสของไวยากรณ์ของอีสดีเอสแอลก่อน โดยจะออกมาอยู่ในรูปของคลาสไลบารีเพื่อเรียกใช้งานด้วยแกรมมาเอ็กพลอเรอร์โดยกดที่ปุ่ม ... ดังภาพที่ 44



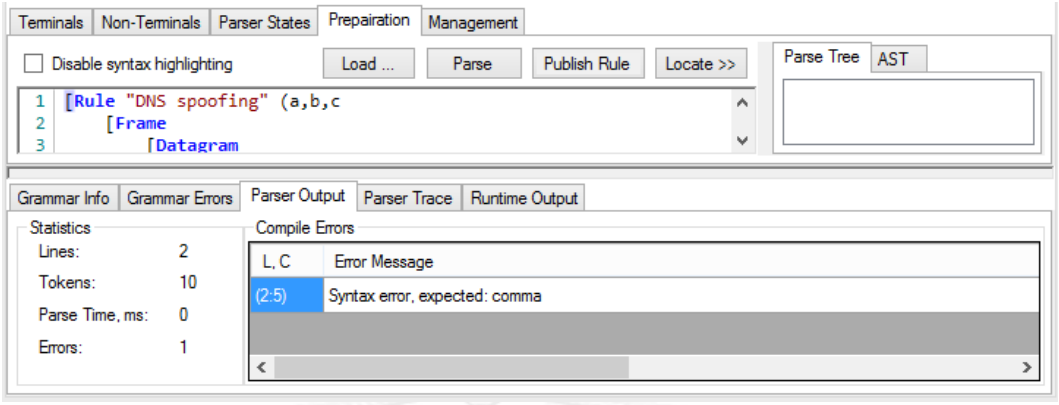
ภาพที่ 44 คลาสไลบารีเพื่อตรวจสอบด้วยแกรมมาเอ็กพลอเรอร์

การตรวจสอบไวยากรณ์สามารถทำได้โดยการเขียนสคริปต์ในแกรมมาเอ็กพลอเรอร์แล้วกดปุ่ม Parse เพื่อตรวจสอบไวยากรณ์ โดยแกรมมาเอ็กพลอเรอร์จะสร้างแผนภาพต้นไม้การแจงส่วนให้ผู้ใช้ทราบในหน้าต่างด้านขวาดังภาพที่ 45



ภาพที่ 45 กดปุ่ม Parse เพื่อตรวจสอบไวยากรณ์เพื่อตรวจสอบด้วยแกรมมาเอ็กพลอเรอร์

โดยหากพบความผิดพลาดของไวยากรณ์ในสคริปต์ที่เขียนจะมีการแจ้งเตือนให้แถบ Parser Output ดังภาพที่ 46



ภาพที่ 46 การแจ้งเตือนกรณีมีความผิดพลาดไวยากรณ์ในสคริปต์

งานวิจัยนี้ได้พัฒนาในส่วนของแกรมมาเอ็กพลอเรอร์ โดยเพิ่มเติมความสามารถเข้าไปเพื่อให้สามารถแปลงสคริปต์ของกฎการตรวจจับการบุกรุกที่เขียนโดยผู้เชี่ยวชาญให้อยู่ในรูปของโครงสร้างของกฎที่จะใช้ในการทำงานในส่วนระบบตรวจจับการบุกรุก และเพื่อการบริหารจัดการกฎที่เขียนขึ้นด้วยอีสดีเอสแอล โดยในกระบวนการแจงส่วนจะเพิ่มเติมในส่วนที่ใช้ในการตรวจสอบตัวแปรที่ไม่มีการประกาศไว้ แต่มีการใช้งานตัวแปรโดยพัฒนาเพิ่มเติมเข้าไปในฟังก์ชันแจงส่วนที่ได้จากไอโรนีดังภาพที่ 47, และภาพที่ 48

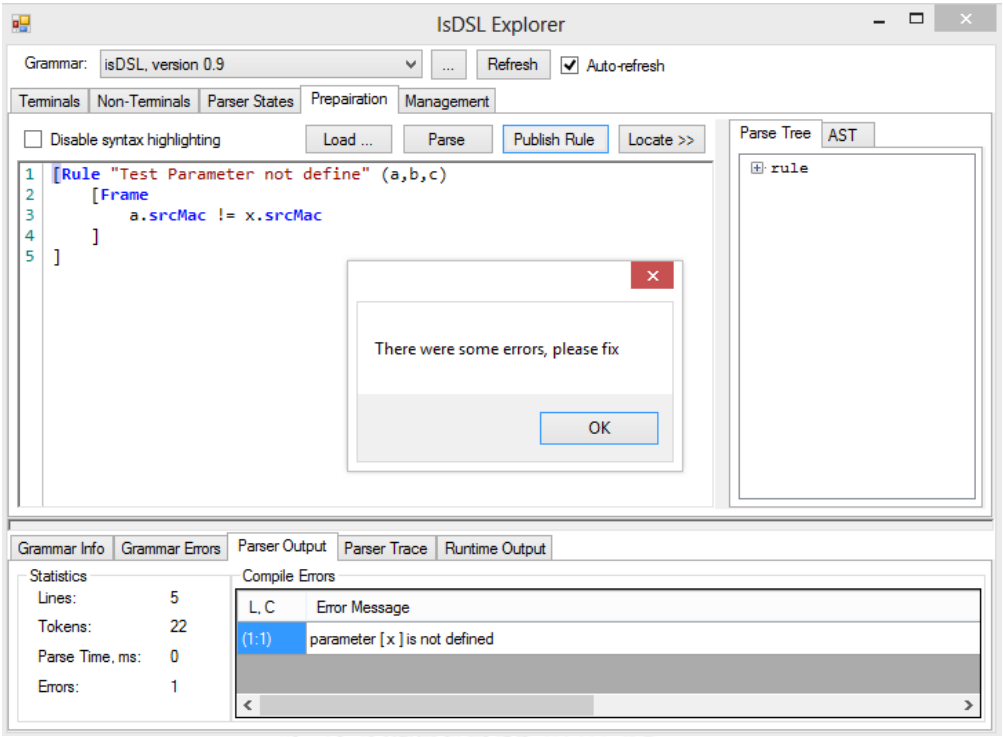
```
    private void ParseSample() {
      ClearParserOutput();
      if (_parser == null || !_parser.Language.CanParse()) return;
      _parseTree = null;
      GC.Collect(); //to avoid disruption of perf times with occasional
collections
      _parser.Context.TracingEnabled = chkParserTrace.Checked;
      try {
        _parser.Parse(txtSource.Text, "<source>");
      } catch (Exception ex) {
        gridCompileErrors.Rows.Add(null, ex.Message, null);
        tabBottom.SelectedTab = pageParserOutput;
        throw;
      } finally {
        _parseTree = _parser.Context.CurrentParseTree;
        isDSLCompile(); //  verify isDSL Grammar
        ShowCompilerErrors();

        if (chkParserTrace.Checked) {
          ShowParseTrace();
        }
        ShowCompileStats();
        ShowParseTree();
        ShowAstTree();
      }
    }
```

ภาพที่ 47 ฟังก์ชันการแจงส่วนที่พัฒนาอีสดีเอสแอลเพิ่มเติม

โดยเมื่อมีการใช้งานตัวแปรที่ไม่ได้มีการประกาศไว้ในส่วนหัวของอีสดีเอสแอลสคริปต์จะทำการแจ้งเตือนผู้ใช้ให้ทราบดังในภาพที่ 49 อีสดีเอสแอลสคริปต์มีการประกาศตัวแปร a, b และ c แต่มีการใช้งานตัวแปร x แกรมมาเอ็กพลอเรอร์จึงแจ้งเตือนความผิดพลาดว่า “parameter [x] is not defined”

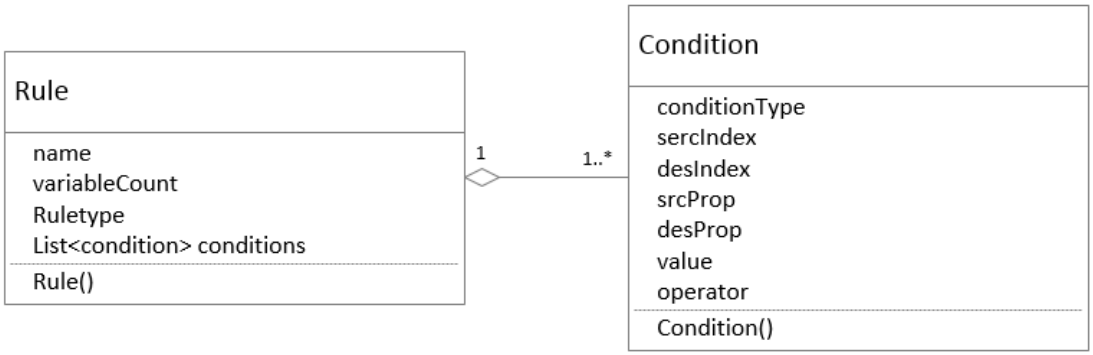
```
private bool IsHasNotDefID(ParseTreeNode idList, List<ParseTreeNode>
Identifier)
    {
        List<string> param = new List<string>();
        List<string> allId = new List<string>();
        foreach(ParseTreeNode id in idList.ChildNodes)
        {
            param.Add(id.FindTokenAndGetText());
        }
        foreach(ParseTreeNode id in Identifier)
        {
            if (!allId.Contains(id.FindTokenAndGetText()))
            {
                allId.Add(id.FindTokenAndGetText());
            }
        }
        foreach(string id in allId)
        {
            if (!param.Contains(id))
            {
                _parseTree.ParserMessages.Add(new
LogMessage(ErrorLevel.Error, SourceLocation.Empty,
                    "parameter [ " + id + " ] is not defined", new
ParserState("Compiled")));
                tabBottom.SelectedTab = pageParserOutput;

                return true;
            }

        }
        return false;
    }
```

ภาพที่ 48 ฟังก์ชันตรวจสอบความถูกต้องของการประกาศตัวแปร



ภาพที่ 49 การแจ้งเตือนกรณีมีการใช้งานตัวแปรที่ไม่ได้ประกาศ

นอกจากนี้ได้มีการพัฒนาในส่วนของฟังก์ชันที่ทำหน้าที่ในการแปลงจากอีสดีเอสแอลสคริปต์ให้กลายเป็นโครงสร้างของกฎที่จะใช้ในการทำงานในส่วนระบบตรวจจับการบุก โดยโครงสร้างของกฎมีแผนภาพคลาสดังภาพที่ 50



ภาพที่ 50 แผนภาพคลาสของโครงสร้างของกฎ

จากภาพในคลาส Rule จะมีคุณลักษณะต่างๆคือ ชื่อกฎ จำนวนตัวแปร ประเภทของกฎ และรายการของเงื่อนไข ซึ่งในกฎหนึ่งๆสามารถมีเงื่อนไขได้หลายเงื่อนไข โดยคลาสของเงื่อนไขมีชื่อเรียกว่าคลาส Condition โดยในการกระบวนการทำงานของฟังก์ชันที่ทำหน้าที่ในการแปลงจากอีสดีเอสแอลสคริปต์ให้กลายเป็นโครงสร้างของกฎจะมีการคอมไพล์อีสดีเอสแอลสคริปต์เพื่อสร้างตารางแฮชชื่อ ruleHash ที่ใช้ในการรวบรวมข้อมูลจากแผนภาพต้นไม้การแจงส่วนในฟังก์ชัน CreateTreeDictionary ดังภาพที่ 51, และภาพที่ 52

```
private void isDSLCompile()
{
    if (_parseTree.ParserMessages.Count == 0)
    {
        var rule = _parser.Context.CurrentParseTree.Root;

        // Create new Rule Hash dictionary
        ruleHash.Clear();
        CreateTreeDictionary(rule);

        // verify that parameter has no conflict
        var Identifier = ruleHash["Identifier"];
        var idList = ruleHash["idList"][0];
        IsHasNotDefID(idList, Identifier);
    }
}
```

ภาพที่ 51 การคอมไพล์อีสดีเอสแอลสคริปต์

```
private void CreateTreeDictionary(ParseTreeNode treeNode)
{
    if (treeNode != null)
    {
        if (ruleHash.ContainsKey(treeNode.Term.Name))
        {
            ruleHash[treeNode.Term.Name].Add(treeNode);
        }
        else
        {
            List<ParseTreeNode> nodeList = new List<ParseTreeNode>();
            nodeList.Add(treeNode);
            ruleHash.Add(treeNode.Term.Name, nodeList);
        }
        foreach (var node in treeNode.ChildNodes)
        {
            CreateTreeDictionary(node);
        }
    }
}
```

ภาพที่ 52 ฟังก์ชันที่ใช้สร้างตารางแฮชที่รวบรวมข้อมูลจากแผนภาพต้นไม้แจงส่วน

โดยแปลงจากอีสดีเอสแอลสคริปต์ให้กลายเป็นโครงสร้างของกฎจะใช้ตารางแฮชที่สร้างขึ้นในการรวบรวมข้อมูลชื่อกฎ จำนวนตัวแปรที่ใช้ และสร้างเงื่อนไขต่างๆ ตามที่ปรากฏในแผนภาพต้นไม้การแจงส่วนโดยในภาพที่ 53 แสดงตัวอย่างการรวบรวมเงื่อนไขทั้งหมดในโหนด frameConStmt (ประโยคเงื่อนไขทั้งหมดในชั้นอีเทอร์เน็ตเฟรม) เพื่อสร้างเป็นเงื่อนไขในโครงสร้างของกฎ

```
// Convert ParseTree to Rule structure
        Rule rule = new Rule();
        rule.variableCount = idList.Count;
        rule.name =
ruleHash["name"][0].FindTokenAndGetText().Replace("\"","");

        if (ruleHash.ContainsKey("frameConStmt"))
        {
            var frameConStmt = ruleHash["frameConStmt"];
            if (frameConStmt.Count > 0)
            {
                rule.layers.Add("frame");
                foreach (ParseTreeNode n in frameConStmt)
                {
                    Condition con = new Condition();
                    ParseTreeNode c = n.ChildNodes[0];
                    con = CreateCondition(c, idList);
                    rule.conditions.Add(con);
                }
            }
        }
```

ภาพที่ 53 ตัวอย่างการรวบรวมเงื่อนไขในโหนด frameConStmt

โดยในการสร้างเงื่อนไขจะต้องรวบรวมคุณลักษณะของคลาสเงื่อนไขซึ่งมีคุณลักษณะดังตารางที่ 11

ตารางที่ 11 คุณลักษณะของคลาส Condition

คุณลักษณะ	**ความหมาย**
ConditionType	ประเภทของเงื่อนไข
srcIndex	ค่าตำแหน่งของตัวแปรแรก
desIndex	ค่าตำแหน่งของตัวแปรที่สอง
srcProp	ค่าคุณลักษณะของแพกเก็ตที่ถูกตรวจสอบของตัวแปรแรก
desProp	ค่าคุณลักษณะของแพกเก็ตที่ถูกตรวจสอบของตัวแปรที่สอง
value	ค่าที่ใช้ในการตรวจสอบในกรณีที่มีการตรวจสอบค่า value กับค่าคุณลักษณะของแพกเก็ต
Op	เครื่องหมายการเปรียบเทียบ คือ “=”, “!=”, “in” และ “!in”

คุณลักษณะของคลาส Condition ถูกสร้างขึ้นเพื่อใช้งานโดยระบบตรวจจับการบุกเครือข่ายของอีสดีเอสแอล โดยประเภทของเงื่อนไข (Condition type) แบ่งออกเป็นประเภทต่างๆดังตารางที่ 12 และประเภทของเครื่องหมายดำเนินการในคุณลักษณะ Op (Operator) แบ่งออกเป็น 4 ประเภทดงตารางที่ 13

ตารางที่ 12 ประเภทของเงื่อนไข

ประเภท	ความหมาย	ตัวอย่าง
PIDoPID	เงื่อนไขระหว่างแพกเก็ตกับแพกเก็ต	a = b, a != b
PPoV	เงื่อนไขระหว่างค่าคุณลักษณะของแพกเก็ตกับค่า	a.port = 80
PPoPP	เงื่อนไขระหว่างค่าคุณลักษณะของแพกเก็ตกับค่าคุณลักษณะของแพกเก็ต	a.desPort = b.srcPort
FoV	เงื่อนไขในรูปแบบฟังก์ชัน	Sequence (b,a,c)

ตารางที่ 13 ประเภทของเครื่องหมายดำเนินการ

ประเภท	ความหมาย	ตัวอย่าง
Equal	เครื่องหมายเปรียบเทียบความเท่ากัน	a = b, a != b
NotEqual	เครื่องหมายเปรียบเทียบความไม่เท่ากัน	a.port = 80
In	เครื่องหมายเปรียบเทียบค่าไอพีแอดเดรสที่อยู่ในข่วงค่าหมายเลขเครือข่าย	a.srcIP in”192.168.1.0/24”
NotIn	เครื่องหมายเปรียบเทียบค่าไอพีแอดเดรสที่ไม่อยู่ในข่วงค่าหมายเลขเครือข่าย	b.desIP !in “192.168.0.1/24”

ซึ่งฟังก์ชันที่ทำหน้าที่ในการสร้างเงื่อนไขจากแผนภาพต้นไม้การแจงส่วนคือฟังก์ชัน CreateCondition โดยจะมีการแบ่งการเก็บข้อมูลคุณลักษณะของเงื่อนไข และแบ่งประเภทของเงื่อนไขที่ถูกสร้างดังแสดงในภาพที่ 54

```
private Condition CreateCondition(ParseTreeNode c, Dictionary<string, int>
idList)
    {
        Condition con = new Condition();

        con.op = c.ChildNodes[3].ChildNodes[0].FindTokenAndGetText() == "="
? Operator.Equal : c.ChildNodes[3].ChildNodes[0].FindTokenAndGetText() ==
"in" ? Operator.In : c.ChildNodes[3].ChildNodes[0].FindTokenAndGetText() ==
"!in" ? Operator.NotIn: Operator.NotEqual;
        con.srcIndex = idList[c.ChildNodes[0].FindTokenAndGetText()];
        con.srcProp = c.ChildNodes[2].FindTokenAndGetText();

        con.type = idList.ContainsKey(c.ChildNodes[4].FindTokenAndGetText())
? ConditionType.PPoPP : ConditionType.PPoV;
        if (con.type == ConditionType.PPoPP)
        {
            con.desIndex = idList[c.ChildNodes[4].FindTokenAndGetText()];
            con.desProp = c.ChildNodes[6].FindTokenAndGetText();
        }
        else
        {
            con.value = c.ChildNodes[4].FindTokenAndGetText();
        }

        return con;
    }
```

ภาพที่ 54 ฟังก์ชันที่ใช้ในการสร้างเงื่อนไขภายในกฎ

นอกจากนี้การเพิ่มเงื่อนไขเข้าไปในกฎนั้นจะมีการตรวจสอบความซ้ำซ้อนของเงื่อนไขต่างๆ เช่นเดียวกับความขัดแย้งของเงื่อนไขต่างๆภายในกฎ เพื่อป้องกันการเขียนเงื่อนไขที่มีความซ้ำซ้อนหรือขัดแย้งกันเอง โดยตัวอย่างของเงื่อนไขที่ซ้ำซ้อน และขัดแย้งกันเองเช่น

- ตัวอย่างเงื่อนไขที่ซ้ำซ้อนกัน : a.desIP = b. desIP กับ b.desIP = a.desIP
- ตัวอย่างเงื่อนไขที่ขัดแย้งกัน : a.desIP != b.desIP กับ a.desIP = b.desIP

โดยในการตรวจสอบเงื่อนไขความขัดแย้งและความซ้ำซ้อนของเงื่อนไข ทำโดยเก็บเงื่อนไขที่แปลงแล้วด้วย Dictionary (ตารางแฮชแบบหนึ่ง) ที่จะประกอบด้วยคู่ Key และ Value ของการแฮช โดยกระบวนการเพิ่มเงื่อนไขเข้าไปยัง Dictionary จะทำการตรวจสอบ Key ที่สร้างจากข้อมูลต่างๆ ของเงื่อนไข เช่นตัวแปร คุณสมบัติที่ตรวจสอบ ค่าของคุณลักษณะ และเครื่องหมายการตรวจสอบ เป็นต้น ซึ่งหามี Key ที่ซ้ำเดิมแสดงว่ามีการเพิ่มเงื่อนไขเข้าไปก่อนหน้าแล้วดังนั้นจุงสรุปได้ว่าเป็นเงื่อนไขที่มีความซ้ำซ้อน เช่นเดียวกับการตรวจสอบเงื่อนไขที่มีความขัดแย้งสามารถทำได้โดยเปลี่ยนเครื่องหมายกาตรวจสอบให้เป็นตรงข้ามและใช้การแฮชทำเป็น Key แล้วนำไปตรวจสอบกับ Key ใน Dictionary ก็จะสามารถตรวจสอบความขัดแย้งของกฎได้เช่นเดียวกันโดยในภาพที่ 55 แสดงฟังก์ชันที่ใช้ในการตรวจสอบเงื่อนไขที่มีความขัดแย้งหรือซ้ำซ้อนกับเงื่อนไขอื่น

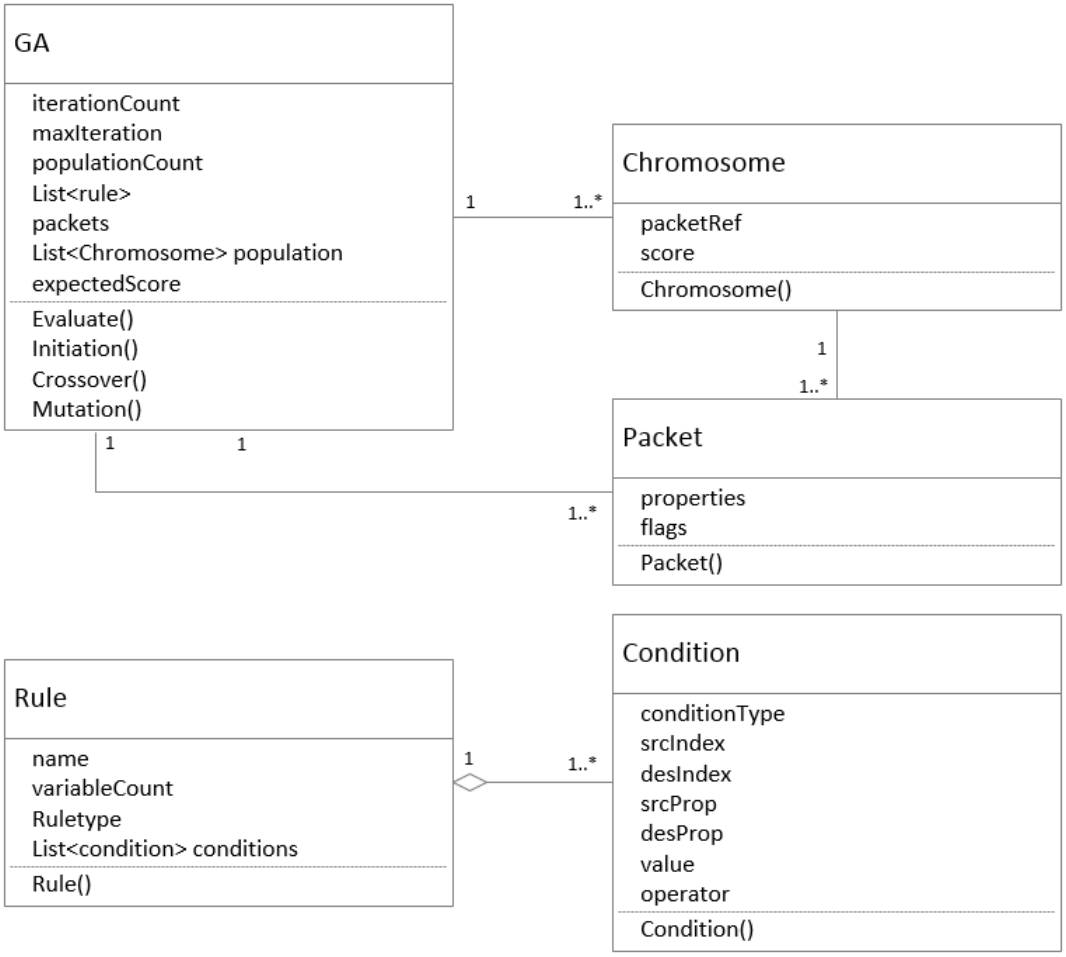
```
private bool ConditionVaidation(Condition c)
    {
        c.srcIndex = c.srcIndex==null?-1 : c.srcIndex;
        c.desIndex = c.desIndex == null ? -1 : c.desIndex;
        string srcIndex = c.srcIndex.ToString();
        string desIndex = c.desIndex.ToString();
        c.srcProp = c.srcProp == null ? string.Empty : c.srcProp;
        c.desProp = c.desProp == null ? string.Empty : c.desProp;
        c.value = c.value == null ? string.Empty : c.value;
        if (conditionHash.ContainsKey(GetMd5Hash(srcIndex + c.srcProp +
desIndex + c.desProp + c.value + c.op.ToString())) ||
            conditionHash.ContainsKey(GetMd5Hash(desIndex + c.desProp +
srcIndex + c.srcProp + c.value + c.op.ToString())) ||
            conditionHash.ContainsKey(GetMd5Hash(srcIndex + c.srcProp +
desIndex + c.desProp + c.value + ChangeOp(c.op).ToString())) ||
            conditionHash.ContainsKey(GetMd5Hash(desIndex + c.desProp +
srcIndex + c.srcProp + c.value + ChangeOp(c.op).ToString())))
        {
            return false;
        }
        else
        {
            conditionHash.Add(GetMd5Hash(srcIndex + c.srcProp + desIndex +
c.desProp + c.value + c.op.ToString()), c);
            conditionHash.Add(GetMd5Hash(desIndex + c.desProp + srcIndex +
c.srcProp + c.value + c.op.ToString()), c);
            conditionHash.Add(GetMd5Hash(srcIndex + c.srcProp + desIndex +
c.desProp + c.value + ChangeOp(c.op).ToString()), c);
            conditionHash.Add(GetMd5Hash(desIndex + c.desProp + srcIndex +
c.srcProp + c.value + ChangeOp(c.op).ToString()), c);
        }
        return true;
    }
```

ภาพที่ 55 ฟังก์ชันที่ใช้ในการตรวจสอบเงื่อนไขที่มีความขัดแย้งหรือซ้ำซ้อนกับเงื่อนไขอื่น

4.3.2 การพัฒนาตัวตรวจจับการบุกรุกเครือข่าย

การพัฒนาระบบตรวจจับการบุกรุกเครือข่ายของอีสดีเอสแอลเพื่อใช้ในการตรวจจับการบุกเครือข่าย โดยจากภาพที่ 34 ระบบตรวจจับการบุกรุกเครือข่ายจะมีส่วนต่อประสานกับตัวแจงส่วนอีสดีเอสแอลเพื่อให้สคริปต์ที่ผู้เชี่ยวชาญเขียนถูกแปลงให้อยู่ในโครงสร้างของกฎแล้วส่งมายังส่วนของระบบตรวจจับการบุกรุกเครือข่ายเพื่อใช้ในการประมวลผลข้อมูลเครือข่ายที่ได้จากตัวเฝ้าระวังการจราจรเครือข่ายที่มีส่วนต่อประสานเชื่อมถึงกัน

การทำงานของระบบตรวจจับการบุกรุกเครือข่ายของอีสดีเอสแอลได้ประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรมเพื่อใช้ในการหากลุ่มของแพกเก็ตที่มีพฤติกรรมตรงกับกฎที่ถูกนิยามขึ้นด้วยอีสดีเอสแอลโดยมีแผนภาพคลาสตามภาพที่ 56



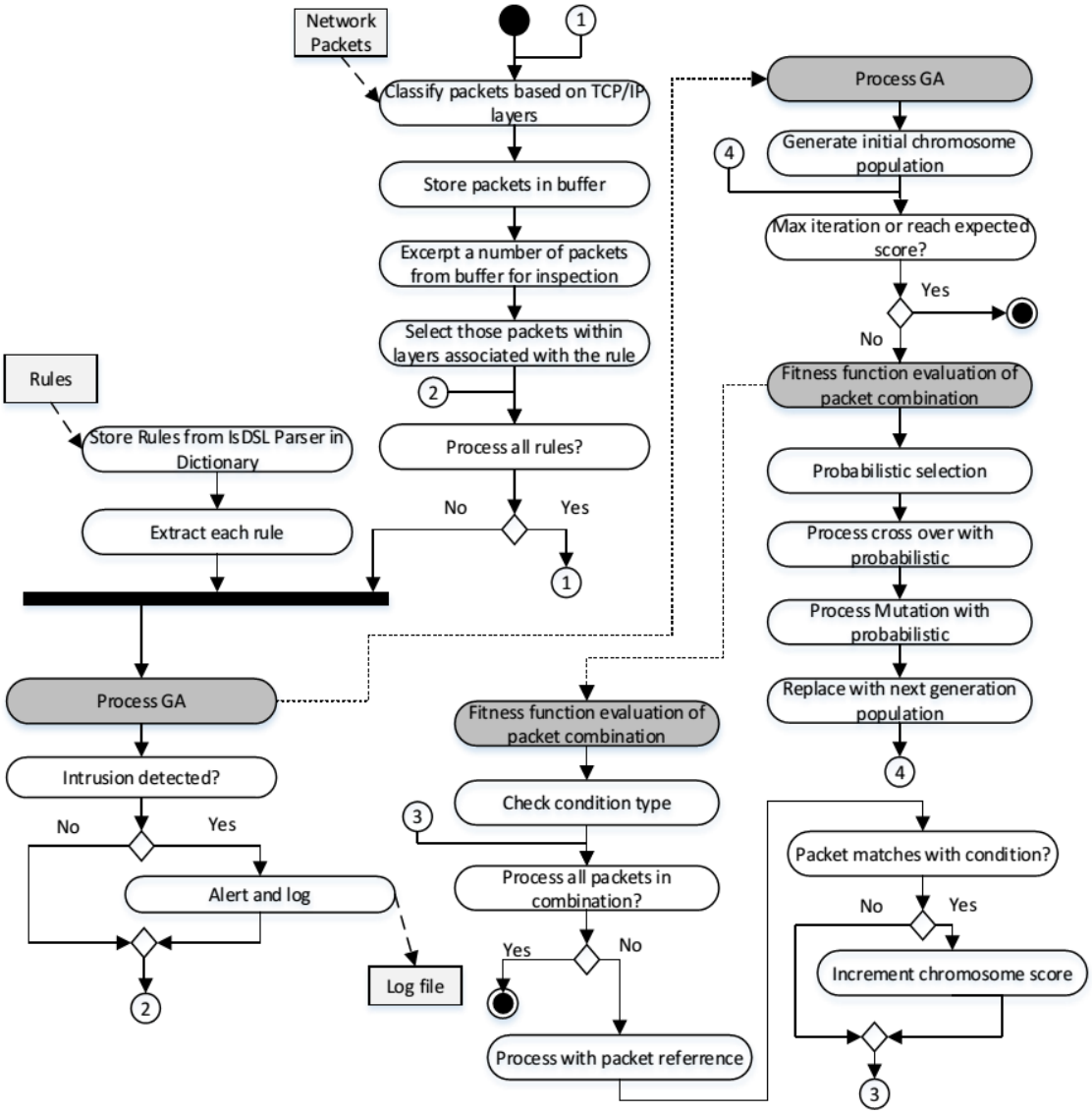
ภาพที่ 56 แผนภาพคลาสที่ใช้งานในระบบตรวจจับการบุกรุกเครือข่าย

โดยคลาส GA จะเป็นคลาสที่ดำเนินการขั้นตอนวิธีเชิงพันธุกรรมโดยมีฟังก์ชันการกำหนดค่าเริ่มต้นที่ทำหน้าที่ในการจัดเตรียมประชากรหรือ List<Chromosome> population ในคลาส GA เพื่อเข้าสู่กระบวนการวนซ้ำของขั้นตอนวิธีเชิงพันธุกรรม

คลาส Chromosome คือคลาสที่เป็นสายพันธุกรรมที่ใช้ในขั้นตอนวิธีเชิงพันธุกกรรม โดยสายพันธุกรรมหรือโครโมโซมจะถูกเข้ารหัสด้วยตำแหน่งอ้างอิงของกลุ่มแพกเก็ตที่เป็นพฤติกรรมการบุกรุกเครือข่าย (PacketRef ในคลาส Chromosome) ตามที่อีสดีเอสแอลสคริปต์กำหนด และแปลงมาเป็นโครงสร้างของกฎ (คลาส Rule) ซึ่งประกอบด้วยรายการเงื่อนไข (คลาส Condition)ที่จะต้องตรวจสอบคุณลักษณะ (properties ในคลาส Packet)ของแพกเก็ตตามตำแหน่งของกลุ่มของแพกเก็ตที่อ้างอิงซึ่งอยู่ในคลาส Packet

การเปรียบเทียบความคล้ายคลึงของค่าคุณลักษณะในแพกเก็ตเครือข่ายกับเงื่อนไขที่กำหนดในโครงสร้างของกฎในขั้นตอนการประเมินค่าความเหมาะสมในขั้นตอนวิธีเชิงพันธุกรรมจะมีการให้คะแนน (score ในคลาส Chromosome) กับกลุ่มของแพกเก็ตที่มีคุณลักษณะที่ตรงกับเงื่อนไข โดยหากตรงกับเงื่อนไข ค่าคะแนนจะถูกเพิ่มขึ้นหนึ่งคะแนน และหากค่าคุณลักษณะของกลุ่มแพกเก็ตตรงกับเงื่อนไขทั้งหมดจะทำให้การให้ค่าคะแนนกับโครโมโซมมีค่าเท่ากับจำนวนของเงื่อนไขทั้งหมดที่มีใน

โครงสร้างของกฎซึ่งมีค่าเท่ากับค่าคะแนนที่คาดหวัง (expected score ในคลาส GA) และหมายถึงกลุ่มของแพกเก็ตที่อ้างอิงโดยโครโมโซมที่มีค่าคะแนนมีค่าเท่ากับค่าคะแนนที่คาดหวังนั้นเป็นกลุ่มแพกเก็ตที่มีพฤติกรรมการบุกรุกที่ถูกกำหนดด้วยอีสดีเอสแอลสคริปต์โดยผู้เชี่ยวชาญ



ภาพที่ 57 แผนภาพกิจกรรมของระบบตรวจจับการบุกรุกเครือข่าย

จากภาพที่ 57 การทำงานของของระบบตรวจจับการบุกรุกเครือข่ายจะมีข้อมูลนำเข้า 2 ส่วน คือ1)แพกเก็ตเครือข่ายที่มาจากตัวเฝ้าระวังการจราจรเครือข่าย ซึ่งจะถูกนำมาจัดเก็บไว้ในบัฟเฟอร์เพื่อรอการประมวลผลของระบบตรวจจับการบุกรุก และ 2) โครงสร้างของกฎที่มาจากตัวแจงส่วนอีสดีเอสแอลของที่ผ่านการแปลงมาจากอีสดีเอสแอลสคริปต์ โดยการประมวลผลมีการดึงแพกเก็ตจากบัฟเฟอร์ไประบบตรวจจับการบุกรุกเพื่อประมวลผลครั้งละ 100 แพกเก็ตซึ่งเปรียบเสมือนการทำงานหน้าต่างเลื่อน (Sliding window) เพื่อตรวจสอบแพกเก็ตในบัฟเฟอร์ว่าเป็นการบุกรุกหรือไม่ และเมื่อประมวลผลเสร็จจะทำการลบแพกเก็ตที่ประมวลผลแล้วออกจากบัฟเฟอร์

การประมวลผลจะเริ่มจากการแยกโครงสร้างของกฎทั้งหมดที่จะทำการประมวลผล ซึ่งข้อมูลรายละเอียดในแต่ละโครงสร้างของกฎจะถูกใช้ในการชี้นำกระบวนการทำงานขั้นตอนวิธีเชิงพันธุกรรมเพื่อค้นหากลุ่มของแพกเก็ตที่ตรงกับเงื่อนไขภายในโครงสร้างของกฎ โดยการประมวลผลจะแตกสายการทำงาน (Thread) เพื่อสร้างการทำงานแบบคู่ขนาน (Parallel) แยกไปตามโครงสร้างของกฎ ซึ่งแต่ละสายการทำงานจะมีการสร้างคลาส GA เพื่อประมวลผลและสร้างรายการของแพกเก็ตที่จำลองมาจากแพกเก็ตที่ดึงมาจากบัฟเฟอร์ซึ่งเลือกเอาเฉพาะแพกเก็ตที่มีความเกี่ยวข้องกับโครงสร้างของกฎ โดยตรวจสอบจากข้อมูลลำดับชั้นของทีซีพี/ไอพีของกฎนั้นๆ หลังจากนั้นหากมีสายการทำงานใดตรวจจับพบกลุ่มของแพกเก็ตที่เป็นพฤติกรรมการบุกรุกเครือข่ายที่กำหนดในโครงสร้างของกฎ ระบบจะแจ้งเตือนไปยังผู้ใช้งานให้ทราบถึงการบุกรุกนั้นและบันทึกลงล็อกไฟล์

กระบวนการทำงานของขั้นตอนวิธีเชิงพันธุกรรมจะเริ่มจากการสร้างกลุ่มโครโมโซมหรือประชากรเริ่มต้นด้วยฟังก์ชันการทำงานในภาพที่ 58 แล้วนำประชากรที่สร้างขึ้นเข้าสู่กระบวนการวนซ้ำของขั้นตอนวิธีเชิงพันธุกรรม โดยเริ่มจากการประเมินค่าความเหมาะสมของโครโมโซมเพื่อให้คะแนน จากนั้นจะเข้าสู่ขั้นตอนการคัดเลือกโดยธรรมชาติซึ่งโครโมโซมที่มีคะแนนหรือค่าความเหมาะสมมาก ก็จะมีความน่าจะเป็นที่จะถูกเก็บไว้เป็นรุ่นถัดไปซึ่งมีอัตราส่วนของจำนวนโครโมโซมที่ถูกเก็บไว้เป็นรุ่นถัดไปจะมีค่าเท่ากับค่าอัตราส่วนการแทนที่ (Replacement ratio) และชุดโครโมโซมอีกส่วนจะถูกเข้าสู่กระบวนการไขว้เปลี่ยนและกระบวนการกลายพันธุ์ เพื่อสร้างโครโมโซมที่มีรูปแบบที่ต่างจากเดิม ซึ่งหมายถึงกลุ่มของแพกเก็ตแบบใหม่และนำไปรวมกับชุดโครโมโซมที่อยู่รอดจากการคัดเลือกโดยธรรมชาติและนำไปแทนที่ประชากรชุดก่อนหน้าและเข้าสู่กระบวนการวนซ้ำในรอบถัดไป ซึ่งหากระหว่างการประเมินค่าความเหมาะสมตรวจสอบพบว่าค่าคะแนนมีค่าเท่ากับค่าคะแนนที่คาดหวังก็จะหยุดเพื่อออกจากกระบวนการวนซ้ำและแจ้งเตือนให้ทราบถึงการบุกรุก แต่หากตรวจไม่พบจนครบรอบสูงสุดของการวนซ้ำ (ค่า max iteration ในคลาส GA) ก็จะหลุดออกจากการตรวจสอบโครงสร้างของกฎนั้นโดยไม่ตรวจพบการบุกรุกเครือข่าย

```
public void Initiation(){// From Class GA
     for (int i = 0; i < populationCount; i++){
         population.Add(new Chromosome(parameterCount, conditionCount,
maxSizeScope));
     }}
public Chromosome(int parameterCount, int conditionCount, int maxSizeScope){
// From Class Chromosome
       packetRef = new int[parameterCount];
       expectedScore = conditionCount;
       Random random = new Random();
       for (int i = 0; i < parameterCount; i++){
           var intArray = Enumerable.Range(0, maxSizeScope).OrderBy(t =>
RandNumber(0, maxSizeScope)).Take(parameterCount).ToArray();
           packetRef = intArray;
       }
           score = 0;}
```

ภาพที่ 58 ฟังก์ชันเริ่มต้นในคลาส GA และ ฟังก์ชันคอนสตรักเตอร์ของคลาส Chromosome

```
       public void Evaluate(){
      if (rule.type == RuleType.accross){
         Parallel.ForEach(rule.conditions, c =>{
            switch (c.type){
                case ConditionType.IDoID{
                     foreach (Chromosome p in population){
                         if (c.op == Operator.Equal &&
p.packetRef[c.srcIndex] == p.packetRef[c.desIndex]){
                            p.score++;
                          }
                          else if (c.op == Operator.NotEqual &&
p.packetRef[c.srcIndex] != p.packetRef[c.desIndex]){
                            p.score++;
                          }}}
                           break;
                case ConditionType.IDPoIDP:{
                     foreach (Chromosome p in population){
                          if (c.op == Operator.Equal &&
packets[p.packetRef[c.srcIndex]].properties[c.srcProp] ==
packets[p.packetRef[c.desIndex]].properties[c.desProp]){
                            p.score++;
                         }
                          else if (c.op == Operator.NotEqual &&
packets[p.packetRef[c.srcIndex]].properties[c.srcProp] !=
packets[p.packetRef[c.desIndex]].properties[c.desProp]){
                            p.score++;
                         }}}
                         break;
```

ภาพที่ 59 ส่วนของฟังก์ชันความเหมาะสม

โดยประเภทของกฎ (RuleType ในคลาส Rule) ถูกสร้างขึ้นเพื่อแบ่งประเภทของกฎออกเป็น 2 ประเภท คือ Singular และ Across โดยประเภทแรกคือกฎที่อธิบายรูปแบบพฤติกรรมการบุกรุกเครือข่ายที่มีเพียงแพกเก็ตเดียว ต่างจากประเภทที่สองที่ใช้ในการอธิบายความสัมพันธ์ของค่าคุณลักษณะระหว่างแพกเก็ตตั้งแต่สองแพกเก็ตขึ้นไป

จากภาพที่ 59 ฟังก์ชันความเหมาะสมที่ใช้ในการประเมินจะมีการแบ่งการทำงานในการประเมินตามประเภทของกฎและเงื่อนไขที่ทำการประเมินโดยจะทำงานแบบคู่ขนานเพื่อความเร็วในการประมวลผล โดยค่าคะแนนของโครโมโซมจะเพิ่มขึ้นตามค่าคุณลักษณะของแพกเก็ตที่ตรงกับเงื่อนไข

```
public void Crossover(){
       for (int i = 0; i < xOverPairNumber; i++){
           if (population.Count > 1) {
               List<int> pairIndex = Enumerable.Range(0,
population.Count).OrderBy(t => RandNumber(0,
population.Count)).Take(2).ToList();
              int value = population[pairIndex[0]].packetRef[0];
              population[pairIndex[0]].SetPacketRef(0,
population[pairIndex[1]].packetRef[0]);
              population[pairIndex[1]].SetPacketRef(0, value);
              psudoPopulation.Add(new Chromosome(population[pairIndex[0]]));
              psudoPopulation.Add(new Chromosome(population[pairIndex[1]]));

              Chromosome c1 = population[pairIndex[0]];
              Chromosome c2 = population[pairIndex[1]];

              population.Remove(c1);
              population.Remove(c2);
            }}}
```

ภาพที่ 60 ฟังก์ชันกระบวนการไขว้เปลี่ยน

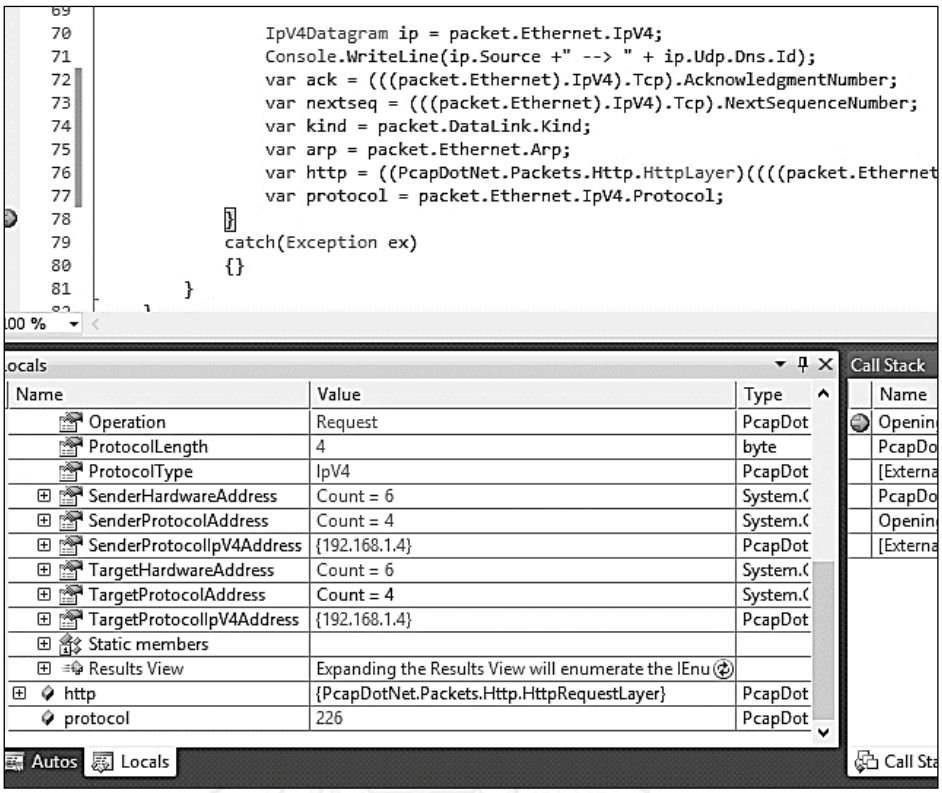
กระบวนการไขว้เปลี่ยนของระบบตรวจจับการบุกรุกเครือข่ายที่พัฒนาขึ้นเป็นประเภทการไข้วเปลี่ยนแบบจุดเดียว (Single point crossover) โดยจะแลกเปลี่ยนคู่กลุ่มรหัสพันธุกรรมจากส่วนหัวและส่วนหางระหว่างโครโมโซมเพียงจุดเดียวดังแสดงในภาพที่ 60

4.3.3 การพัฒนาตัวเฝ้าระวังการจราจรเครือข่าย

การพัฒนาตัวเฝ้าระวังการจราจรเครือข่ายได้เลือกใช้พีแคปดอทเนตไลบรารี [14] ในการดักจับแพกเก็ตจากเครือข่ายเพื่อมาประมวลผลด้วยส่วนตรวจจับการบุกรุกเครือข่ายที่พัฒนาขึ้น ซึ่งโดยปกติแพกเก็ตที่ส่งไปมาในเครือข่ายจะอยู่ในรูปของ Hex stream ดังภาพที่ 61 ซึ่งพีแคปดอทเนตไลบรารีจะถอดรหัส (Decode)ให้อยู่ในรูปที่มนุษย์สามารถอ่านได้ตามภาพที่ 62

```
0000  f0 de f1 d1 ce b6 00 1b  d4 dd 81 e0 08 00 45 00   ........ ......E.
0010  02 c9 83 03 40 00 4b 06  18 7c 45 ab f8 10 9d 3c   ....@.K. .|E....<
0020  b6 b7 01 bb 3c bd 6e 65  db 5f 56 58 a4 0c 50 18   ....<.ne ._VX..P.
0030  01 1b 77 0d 00 00 17 03  01 02 9c a1 df 46 29 4d   ..w..... .....F)M
0040  86 b2 df c2 83 ed 58 c6  0d 54 d4 f6 3a ec 4c c2   ......X. .T..:.L.
0050  f0 70 3f 05 e5 6e a4 7d  61 da 09 13 58 c4 05 9f   .p?..n.} a...X...
0060  e3 3c 9c d7 ef a1 8b 46  be a3 ac 27 18 86 b9 b9   .<.....F ...'....
0070  53 2d 9c 1e 18 19 1f 50  fa 30 e3 9f 1b a7 d7 63   S-.....P .0.....c
0080  18 7e ec 5b f5 67 1d 41  a3 c4 a1 04 99 02 ec b4   .~.[.g.A ........
0090  cd 80 d8 46 2c 2a 93 8e  a3 34 e3 5b eb da cc 18   ...F,*.. .4.[....
00a0  0e c3 56 56 69 26 7e c0  cf 21 65 bb 64 b6 68 34   ..VVi&~. .!e.d.h4
00b0  8a e1 54 4e 17 42 5c 18  12 a0 87 24 41 34 7d 9b   ..TN.B\. ...$A4}.
00c0  fe 78 db a0 b9 a9 25 14  65 d1 b9 25 09 8b f8 62   .x....%. e..%...b
```

ภาพที่ 61 แพกเก็ตเครือข่ายที่อยู่ในรูป Hex stream



ภาพที่ 62 ตัวอย่างค่าของแพกเก็ตที่ถูกแปลงให้อยู่ในรูปที่อ่านได้โดยพีแคปดอทเนตไลบรารี

โดยขั้นตอนของการพัฒนาเริ่มต้นที่การสร้างส่วนที่เชื่อมต่อกับส่วนต่อประสานเครือข่าย โดยเรียกใช้ออปเจ็ค (object) LivePacketDevice ของพีแคปดอทเนตไลบรารีเพื่อรวบรวมรายการของส่วนต่อประสานเครือข่ายดังภาพที่ 63

```
// Retrieve the device list from the local machine
IList<LivePacketDevice> allDevices= LivePacketDevice.AllLocalMachine;
```

ภาพที่ 63 การรวบรวมรายการของส่วนต่อประสานเครือข่าย

หลังจากเลือกส่วนต่อประสานเครือข่ายแล้วจึงเรียกสร้างส่วนที่ใช้ในการเชื่อมต่อกับส่วนต่อประสานเพื่อการดักจับแพกเก็ตเครือข่ายโดยมีพารามิเตอร์คือ หมายเลขพอร์ตที่จะเปิดรับแพกเก็ตเครือข่าย(ใช้พอร์ต 65536) และโหมดของการเปิดส่วนต่อประสาน (จะเลือกใช้โหมด Promiscuous เพื่อให้ส่วนต่อประสานสามารถรับแพกเก็ตทุกๆแพกเก็ต) และค่าเวลาสิ้นสุดการรอ (Timeout) ดังภาพที่ 64

```
 public static void NetworkMonitor(object Device)
    {
            PacketDevice selectedDevice = (PacketDevice)Device;
            using (PacketCommunicator communicator =
                    selectedDevice.Open(65536,
// portion of the packet to capture
                // 65536 guarantees that the whole packet will be captured
on all the link layers

PacketDeviceOpenAttributes.Promiscuous, // promiscuous mode
                                        1000))
// read timeout
            {
                //Console.WriteLine("Listening on " +
selectedDevice.Description + "...");

                // start the capture
                communicator.ReceivePackets(0, PacketHandler);
            }
    }
```

ภาพที่ 64 ฟังก์ชันที่ใช้ในการเชื่อมต่อกับส่วนต่อประสานเพื่อการดักจับแพกเก็ตเครือข่าย

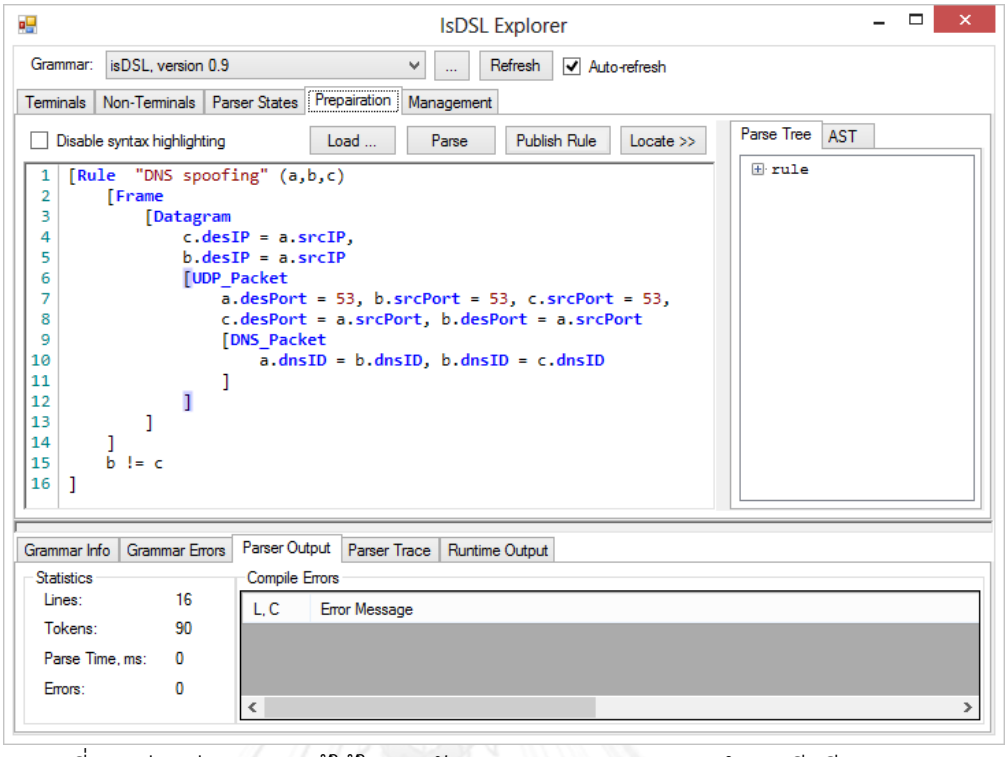
จากนั้นเมื่อมีแพกเก็ตเข้ามายังส่วนต่อประสานเครือข่ายจะถูกจัดการโดยฟังชันการจัดการแพกเก็ตเพื่อกรองเอาเฉพาะแพกเก็ตในโพรโทคอลทีซีพี/ไอพี ซึ่งอยู่ในขอบเขตของอีสดีเอสแอลดัง ภาพที่ 65

```
private  static void PacketHandler(Packet packet)
    {
        try
        {
            if (packet.IsValid && ((packet.Ethernet.IpV4.Protocol ==
IpV4Protocol.Tcp && packet.Ethernet.EtherType ==
PcapDotNet.Packets.Ethernet.EthernetType.IpV4 &&
packet.Ethernet.IpV4.Tcp.IsValid) ||
                (packet.Ethernet.IpV4.Protocol == IpV4Protocol.Udp &&
packet.Ethernet.EtherType == PcapDotNet.Packets.Ethernet.EthernetType.IpV4
&& packet.Ethernet.IpV4.Udp.IsValid) ||
                (packet.Ethernet.EtherType ==
PcapDotNet.Packets.Ethernet.EthernetType.Arp &&
packet.Ethernet.Arp.IsValid)))
            {
```

ภาพที่ 65 ส่วนของฟังชันการจัดการแพกเก็ตที่เข้ามายังส่วนต่อประสานเครือข่าย

4.4 การพัฒนาส่วนต่อประสานผู้ใช้งาน (User interface)

การพัฒนาส่วนติดต่อผู้ใช้งานพัฒนาด้วยวินโดวส์ฟอร์มแอพลิเคชัน (Windows form application) ที่พัฒนาโดยมีพื้นฐานบนไอโรนีแกรมมาเอ็กพลอเรอร์ โดยพัฒนาเพิ่มเติมในส่วนของส่วนต่อประสานผู้ใช้งานที่ใช้ในการเขียน แก้ไข และการประกาศ (Publish)กฎในอีสดีเอสแอลสคริปต์ โดยส่วนต่อประสานผู้ใช้งานเป็นไปตามภาพที่ 66



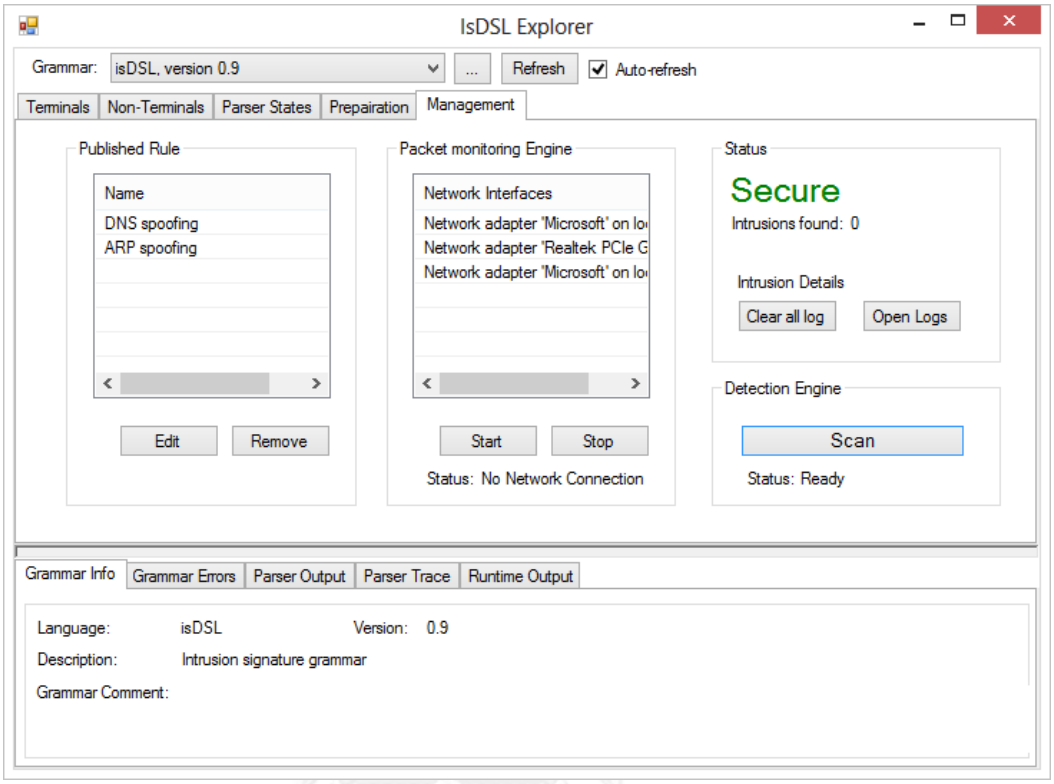
ภาพที่ 66 ส่วนต่อประสานผู้ใช้ในการจัดการกฎและควบคุมการทำงานอีสดีเอสแอล

ผู้ใช้งานสามารถกดปุ่ม Publish Rule เพื่อประกาศกฎที่ใช้ในการตรวจจับไปยังระบบตรวจจับการบุกรุกได้ ซึ่งโปรแกรมจะตรวจสอบวากยสัมพันธ์ของสคริปต์ที่เขียนในส่วนต่อประสานและแปลงอีสดีเอสแอลสคริปต์ให้อยู่ในรูปโครงสร้างของกฎ โดยการพัฒนาฟังก์ชันส่วนของการประกาศกฎเป็นไปตามภาพที่ 67

```
    private void Publish_btn_Click(object sender, EventArgs e)
    {
        ParseSample();
        if (lblParseErrorCount.Text != "0")
        {
            MessageBox.Show("There were some errors, please fix");
            return;
        }

       // Prameter of Rule
       Dictionary<string, int> idList = new Dictionary<string,int>();
        int idIndex = 0;
        foreach (ParseTreeNode n in ruleHash["idList"][0].ChildNodes)
        {
            idList.Add(n.FindTokenAndGetText(), idIndex);
            idIndex++;
        }

        // Convert ParseTree to Rule structure
        Rule rule = new Rule();
        rule.variableCount = idList.Count;
        rule.name =
ruleHash["name"][0].FindTokenAndGetText().Replace("\"","");
```

ภาพที่ 67 ส่วนหนึ่งของฟังก์ชันการ Publish กฎในไอโรนีแกรมมาเอ็กพลอเรอร์

นอกจากนี้ยังเพิ่มเติมในส่วนของการจัดการกฎอื่นๆที่ถูกประกาศไว้ก่อนหน้ารวมไปถึงการควบคุมการทำงานของระบบตรวจจับการบุกรุกของอีสดีเอสแอลตามภาพที่ 68



ภาพที่ 68 ส่วนต่อประสานผู้ใช้ในการจัดการกฎและควบคุมการทำงานอีสดีเอสแอล

โดยความสามารถของส่วนต่อประสานผู้ใช้ในภาพที่ X สามารถแบ่งออกเป็น 3 ส่วนหลักๆคือ

1. ส่วนที่ใช้ในการจัดการกฎของอีสดีเอสแอล(ส่วน Published Rule) โดยผู้ใช้งานสารมารถกลับไปแก้ไขสคริปต์ของกฎที่ถูกประกาศไว้ก่อนหน้าโดยการกดปุ่ม Edit และสามารถลบหรือลบกฎที่ไม่ต้องการออกจากระบบได้โดยการกดปุ่ม Delete
2. ส่วนที่ใช้ในการควบคุมการทำงานของตัวเฝ้าระวังการจราจรเครือข่าย (ส่วน Packet monitoring engine)เพื่อนำมาประมวลผล โดยผู้ใช้งานสามารถเลือกส่วนต่อประสานเครือข่ายจากรายการในตาราง Network interface และกดปุ่ม Start เพื่อเริ่มการดักจับข้อมูลจากเครือข่ายหรือกดปุ่ม Stop เพื่อหยุดการดักจับข้อมูลเครือข่าย
3. ส่วนที่ใช้ในการควบคุมการทำงานของระบบตรวจจับการบุกรุกเครือข่าย(ส่วน Detection engine) ผู้ใช้งานสามารถเริ่มการทำงานของระบบตรวจจับการบุกรุกเครือข่ายด้วยการกดปุ่ม Scan และสามารถเรียกดูล็อกไฟล์ย้อนหลังหรือลบล็อกไฟล์ด้วยการกดปุ่ม Open log และ ปุ่ม Clear log

โดยในภาพที่ 69 แสดงส่วนของฟังก์ชันที่ใช้ในการบันทึกโครงสร้างของกฎให้อยู่ในรูปของไฟล์ไบนารีเพื่อใช้ในการบันทึกและเรียกคืนกฎมายังตัวตรวจจับการบุกรุกเครือข่าย

```
// Save to bin file
        using (Stream stream = File.Open(filename, FileMode.Create))
        {
            var write = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
            write.Serialize(stream, ruleList);
        }

// Load List of previous isDSL rules
        string filename = Path.Combine(Environment.CurrentDirectory,
@"rule\Rules.isDSL");

        if (!File.Exists(filename))
        {
            ruleList = new Dictionary<string, Rule>();
        }
        else
        {
            using (Stream stream = File.Open(filename, FileMode.Open))
            {
                var read = new
System.Runtime.Serialization.Formatters.Binary.BinaryFormatter();
                ruleList = (Dictionary<string,
Rule>)read.Deserialize(stream);
            }
        }
```

ภาพที่ 69 ส่วนของฟังก์ชันที่ใช้ในการบันทึกและเรียกคืนโครงสร้างของกฎจากไฟล์ไบนารี

บทที่ 5
การประเมินและการวัดผล

5.1 การเปรียบเทียบความสามารถระหว่าง isDSL และกฎของ Snort

การเปรียบเทียบระหว่างกฎของ isDSL และ Snort ตามความสามารถดังตารางที่ 14

ตารางที่ 14 การเปรียบเทียบความสามารถของ isDSL และกฎของ Snort

กฎของ Snort	**isDSL**
1. Simple rule: สามารถตรวจสอบค่าคุณลักษณะในแพกเก็ตหนึ่งๆเท่านั้น	1. สามารถตรวจสอบค่าคุณลักษณะข้ามแพกเก็ตได้
2. Protocol preprocessor: สามารถระบุค่าการตรวจสอบของพารามิเตอร์ในแต่ละโพรโทคอลได้	2. ปัจจุบันขอบเขตความสามารถของ isDSL ยังไม่ครอบคลุมในส่วนอื่นๆ นอกเหนือจากส่วนหัวของแพกเก็ต
3. Flow & Steam preprocessor: สามารถตรวจสอบความสัมพันธ์ระหว่างกฎได้	3. N/A สำหรับภาษาแบบ declarative
4. ต้องการความรู้เรื่องการโปรแกรม	4. ไม่ต้องการความรู้เรื่องการโปรแกรม

5.2 ขีดความสามารถของอีสดีเอสแอลในการสร้างกฎที่ใช้ในการตรวจจับการบุกรุกเครือข่าย

จากประเภทการโจมตีบนโพรโทคอลทีซีพี/ไอพีทั้ง 6 ประเภทที่ถูกอธิบายไว้ในบทที่ 2 นั้น ได้ถูกนำมาวิเคราะห์เพื่อออกแบบและสร้างอีสดีเอสแอลในการสร้างกฎที่ใช้ในการตรวจจับการบุกรุกประเภทต่างๆ ประกอบการทบทวนวรรณกรรมในเรื่องของการแก้ไขและป้องกันการโจมตีประเภทต่างๆ พบว่าการโจมตีแบบปลอมแปลงนั้นตรวจจับได้ยาก เนื่องจากเป็นการอาศัยช่องโหว่ที่เป็นไปตามกฎของโพรโทคอลนั้นๆ เช่นเดียวกับการโจมตีเพื่อระงับการให้บริการซึ่งต้องอาศัยการตรวจสอบและการนับจำนวนแพกเก็ตในการตรวจจับการบุกรุกประเภทนี้

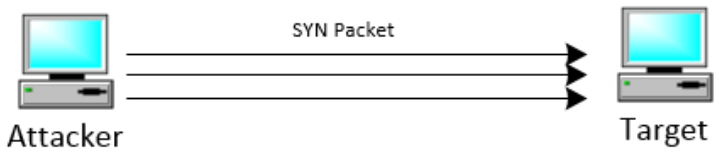
ตารางที่ 15 ขีดความสามารถของอีสดีเอสแอลในการสร้างกฎที่ใช้ในการตรวจจับการบุกรุกทั้ง 6 ประเภท

ประเภทการโจมตี	**สามารถเขียนเป็นกฎ isDSL**	**คำอธิบาย**
การโจมตีเพื่อระงับการให้บริการ	√	ตัวอย่างกฎ เช่น UDP loop flooding และ TCP SYN flooding เป็นต้น
การโจมตีโดยการดักจับแพกเก็ต	N/A	การป้องกันการโจมตีประเภทนี้กระทำได้โดยการเข้ารหัสลับข้อมูล เนื่องจากผู้บุกรุกไม่ได้ส่งแพกเก็ตไปยังเครือข่าย

		เพื่อโจมตี แต่เป็นการดักจับแพกเก็ตเพื่อนำไปวิเคราะห์เพื่อล้วงข้อมูลที่เป็นความลับ
การโจมตีแบบปลอมแปลง	√	ตัวอย่างกฎ เช่น DNS spoofing และ ARP spoofing เป็นต้น
การโจมตีตารางโปรเซส	√ แต่ต้องขยายความสามารถภาษาในการเข้าถึงข้อมูลนอกเหนือจากส่วนหัวของแพกเก็ต	เนื่องจากขอบเขตของอีสดีเอสแอลกำหนดอยู่บนส่วนหัวของแพกเก็ตเท่านั้น แต่ข้อมูลการสั่งงานโปรเซสนั้นอยู่ในส่วนของข้อมูลแพกเก็ต (ในชั้นแอปพลิเคชัน) และการโจมตีประเภทนี้สามารถแก้ไขได้โดยการจำกัดจำนวนโปรเซสที่เครื่องเจ้าบ้าน (Host) ได้
การโจมตีโดยกระบวนการสร้างหมายเลขลำดับของโพรโทคอลทีซีพี	N/A	ปัจจุบันการโจมตีประเภทนี้สามารถป้องกันได้โดยการเข้ารหัสลับข้อมูลในกระบวนการสร้างและส่งหมายเลขลำดับเพื่อไม่ให้ผู้บุกรุกสามารถทำนายได้โดยง่าย
การโจมตีแบบไอพีฮาร์ฟสแกน	√	ตัวอย่างเช่น IP half scan และ port scanning เป็นต้น

ตัวอย่างการโจมตีเพื่อระงับการให้บริการ

TCP SYN flooding: เป็นการโจมตีที่ผู้โจมตีส่งแพกเก็ต SYN จำนวนมากไปยังเครื่องเป้าหมาย ทำให้เครื่องเป้าหมายไม่สามารถรับการร้องขอบริการอื่นๆได้ ดังภาพที่ 70



ภาพที่ 70 การโจมตี TCP SYN flooding

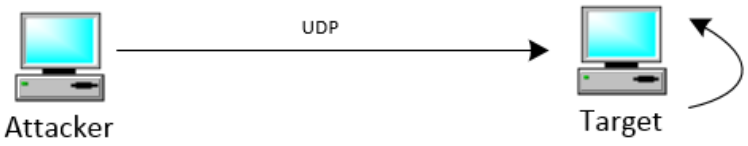
ตัวอย่างสคริปต์อีสดีเอสแอลของกฎที่ใช้ในการตรวจจับ TCP SYN flooding เป็นไปตามภาพที่ E

```
[Rule  "Syn flooding" (a)
      [TCP_Packet
          a.syn = 1
      ]
      Count(a,30)
]
```

ภาพที่ 71 กฎอีสดีเอสแอล TCP SYN flooding

จากภาพที่ 71 แพกเก็ต “a” มีเงื่อนไขการตรวจสอบ SYN flag ให้มีค่าเท่ากับ 1 และเงื่อนไขของฟังก์ชันควบคุม Count() นับจำนวนแพกเก็ต “a” ว่ามีจำนวนมากกว่าหรือเท่ากับ 30 แพกเก็ตหรือไม่ (โดยนับจำนวนแพกเก็ตที่มีค่าคุณลักษณะอื่นๆ เช่น หมายเลขไอพีต้นทางและปลายทางเดียวกัน) ในการตรวจจับการบุกรุกประเภทนี้ โดยหากตรงกับเงื่อนไข ระบบตรวจจับการบุกรุกจะทำการแจ้งเตือนให้ผู้ใช้งานทราบ

UDP loop flooding: ภาพที่ 72 แสดงการโจมตีประเภท UDP loop flooding ซึ่งเป็นการโจมตีที่แตกต่างจาก SYN flooding ที่ใช้การส่งแพกเก็ตจำนวนมากๆในการโจมตี แต่การโจมตีรูปแบบนี้จะอาศัยช่องโหว่ของพอร์ต 7 ที่เปิดให้สามารถเรียกใช้งานหรือสร้างโปรเซสได้ ในระบบปฏิบัติการลินุกซ์ซึ่งผู้โจมตีส่งแพกเก็ตที่ทำให้เครื่องเป้าหมายส่งแพกเก็ตซ้ำๆให้กับเครื่องตัวเอง เช่น การเรียกคำสั่ง HPING ผ่านยูดีพีแพกเก็ตในภาพที่ 73 โดยทั่วไปเรียกการโจมตีประเภทนี้ว่าหยดน้ำตา (Tear drop) เนื่องจากเป็นการโจมตีที่ใช้เพียงแพกเก็ตเดียวในการระงับการให้บริการ



ภาพที่ 72 การโจมตี UDP loop flooding

```
root@badhost:~/coding# hping -c 1 --udp -s 4343 -p 7 -d 1
1xx.168.xxx.xxx
  HPING 1xx.168.xxx.xxx (eth0 1xx.168.xxx.xxx): udp mode set,
28 headers + 1 data bytes
  len=46 ip=1xx.168.xxx.xxx ttl=36 id=63524 seq=0 rtt=284.6 ms

  --- 1xx.168.xxx.xxx hping statistic ---
  1 packets transmitted, 1 packets received, 0% packet loss
  round-trip min/avg/max = 284.6/284.6/284.6 ms
```

ภาพที่ 73 การเรียกคำสั่ง HPING เพื่อทำ UDP loop flooding

ตัวอย่างสคริปต์อีสดีเอสแอลของกฎที่ใช้ในการตรวจจับ UDP loop flooding ดังภาพที่ 74

```
[Rule  "UDP loop flooding" (a)
      [Datagram
          a.desIP = a.srcIP
          [UDP_Packet
              a.desPort = 7
          ]
      ]
      Count(a,20)
]
```

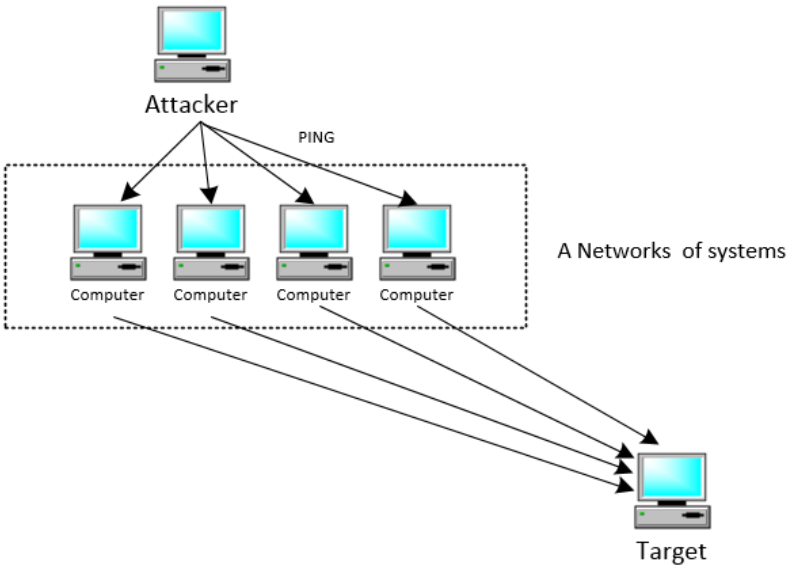
ภาพที่ 74 กฎที่ใช้ในการตรวจจับ UDP loop flooding

กฎที่สร้างขึ้นจะตรวจสอบแพกเก็ตที่มีหมายเลขพอร์ตปลายทางมีค่าเท่ากับ 7 และนับจำนวนแพกเก็ตว่ามีจำนวนมากกว่าหรือเท่ากับ 20 แพกเก็ตหรือไม่เพื่อการตรวจจับการบุกรุกประเภทนี้

งานวิจัยนี้มีขอบเขตของการสร้างภาษาจำเพาะโดเมนซึ่งถูกกำหนดอยู่เฉพาะในส่วนหัวของอีเทอร์เน็ตเฟรม, อาร์พแพกเก็ต, ไอพีดาตาแกรม, ทีซีพีแพกเก็ต, ยูดีพีแพกเก็ต และดีเอนเอสแพกเก็ต ซึ่งไม่ครอบคลุมถึงการโจมตีที่เกิดขึ้นกับโพรโทคอลอื่นๆในชั้นต่างๆ เช่น โพรโทคอลไอซีเอ็มพี (ICMP) ในชั้นโฮสต์ทูโฮสต์ที่ใช้ในการส่งข้อแจ้งเตือนในกรณีที่เครือข่ายเกิดปัญหา หรือโพรโทคอลต่างๆที่อยู่บนชั้นแอปพลิเคชัน เช่น เฮชทีทีพี (HTTP) และเอฟทีพี (FTP) เป็นต้น

ตัวอย่างการโจมตีเพื่อระงับการให้บริการบนโพรโทคอลเฮชทีทีพีในชั้นแอปพลิเคชันซึ่งเป็นโพรโทคอลที่อยู่นอกเหนือขอบเขตงานวิจัย เช่น ในการให้บริการเว็บเซิร์ฟเวอร์ Microsoft IIS เวอร์ชัน 3.0 สามารถระงับการให้บริการได้โดยการส่งเฮชทีทีพีแพกเก็ตที่มีคำสั่งร้องขอ คือ “GET /?foo=XX< *1180>XX HTTP/1.0” จะทำให้บริการไม่สามารถให้บริการได้เช่นเดียวกับในเว็ปเว็บเซิร์ฟเวอร์ Apache 1.1.1 ที่มีช่องโหว่ของการค้นหาไฟล์ “index.html” ซึ่งผู้โจมตีสามารถร้องขอยูอาร์แอล (URL) “http://www,server.com//////////[many]//////” ก็จะสามารถระงับการให้บริการของ Apache เวอร์ชันนี้ได้เช่นเดียวกัน

Ping flooding: เป็นการโจมตีที่ผู้โจมตีอาศัยการ Ping ที่ใช้ในการตรวจสอบการเชื่อมต่อระหว่างเครื่องในเครือข่ายโดยการส่ง Ping ปริมาณมากๆไปยังเครื่องเป้าหมาย เพื่อให้ไม่สามารถรับการร้องขอบริการอื่นๆได้ โดยทั่วไปจะใช้แพกเก็ต ICMP_ECHO หรือ ICMP_ECHOREPLY ในการโจมตี นอกจากนี้ ยังสามารถโจมตีโดยการ Ping ไปยังเครื่องอื่นๆก่อน แต่ตั้งค่าไอพีต้นทางไปยังเครื่องเป้าหมายทำให้เครื่องอื่นๆที่ได้รับแพกเก็ต Ping จะตอบกลับไปยังเครื่องเป้าหมายจำนวนมาก โดยทั่วไปเรียกการโจมตีประเภทนี้ว่า Smurf Attack ซึ่งเป็นการโจมตีประเภท DDOS (Distributed Denial of Service) คือใช้เครื่องปริมาณมากในการโจมตีเพื่อระงับการให้บริการ ดังภาพที่ 75



ภาพที่ 75 Smurf Attack

Large packet (ping of death): เหมือนการโจมตีประเภท Ping flooding แต่ผู้โจมตีจะส่งแพกเก็ตที่มีขนาดใหญ่หรือมีข้อมูลปริมาณมากๆไปยังเครื่องเป้าหมายผ่านการ Ping โดยส่วนใหญ่จะส่ง ICMP_ECHO และแนบผ่านไปกับโพรโทคอลไอพี และอาศัยการ Fragmentation ให้เป็นแพกเก็ตย่อยๆและส่งไปในเครือข่ายเพื่อหลีกเลี่ยงการตรวจจับหรือกรองแพกเก็ตโดยการจำกัดขนาดของแพกเก็ต

การโจมตีแบบปลอมแปลงเป็นการโจมตีที่สามารถเกิดขึ้นกับฮาร์ดแวร์ และซอฟต์แวร์ที่ใช้ในการเชื่อมต่อ ทำให้การส่งข้อมูลจากต้นทางไปยังปลายทางผิดพลาด โดยการโจมตีประเภทนี้ตรวจจับได้ยากมากสำหรับการใช้งานเครือข่ายในปัจจุบัน โดยผู้บุกรุกจะทำการปลอมแปลงข้อมูลแพกเก็ตที่ส่งในเครือข่ายเพื่อช่วงชิงข้อมูล หรือทำให้ผู้โจมตีได้สิทธิ์เพื่อใช้ในการเข้าถึงข้อมูลที่สำคัญ เช่น การเชื่อมต่อแบบเซิร์ฟเวอร์ไคลแอนต์ (Server-Client) ในโพรโทคอลทีซีพี ที่มีหมายเลขลำดับ (Sequence number) ในการเชื่อมต่อของแพกเก็ตที่ส่งเพื่อป้องการความซ้ำซ้อนและความผิดพลาดในการส่งข้อมูล ซึ่งระหว่างการเริ่มต้นการเชื่อมต่อของโพรโทคอลทีซีพี จะมีกระบวนการทรีเวย์แฮนด์เชคเพื่อสร้างเซสชั่นการเชื่อมต่อระหว่างเครื่องที่ให้บริการและเครื่องปลายทาง ซึ่งผู้บุกรุกที่สามารถล่วงรู้หมายเลขลำดับที่ใช้ในการเชื่อมต่อสามารถสร้างแพกเก็ตปลอมโดยใช้หมายเลขการเชื่อมต่อนั้นในการส่งข้อมูลเพื่อหลอกผู้บุกรุกได้รับเซสชันการเชื่อมต่อไป โดยทั่วไปเรียกการโจมตีแบบนี้ว่า การช่วงชิงเซสชัน (Session hijacking) หรือการโจมตีแบบปลอมแปลงอาร์พ ที่ผู้โจมตีส่งเพ็คเกจอาร์พที่ปลอมขึ้นไปยังเครือข่าย โดยใส่ข้อมูลแม็คแอดเดรส (Mac address) ของเครื่องผู้โจมตีกับไอพีแอดเดรสของเครื่องเป้าหมาย ทำให้การส่งข้อมูลไปหาเครื่องเป้าหมายถูกเปลี่ยนไปให้ส่งไปยังเครื่องของผู้โจมตีแทน โดยตัวอย่างการเขียนสคริปต์อีสดีเอสแอลในการตรวจจับการโจมตีแบบปลอมแปลงดังภาพที่ 79 “DNS spoofing attack” และ ภาพที่ 80 “ARP spoofing attack” ในหัวข้อ 5.2

การโจมตีแบบไอพีฮาร์ฟสแกนเป็นการโจมตีที่ผู้บุกรุกต้องการทราบข้อมูลของเครื่องเป้าหมายซึ่งอาจเรียกการโจมตีประเภทนี้ว่า “Port Scanning” โดยการส่งแพกเก็ตร้องขอไปยัง

พอร์ตต่างๆของเครื่องเป้าหมายและวิเคราะห์การตอบกลับจากเครื่องเป้าหมายตามโพรโทคอลของบริการนั้นๆ เนื่องจากการให้บริการต่างๆจะมีหมายเลขพอร์ตเฉพาะที่ใช้ในการเชื่อมต่อกับบริการหนึ่งๆ เช่น พอร์ต 80 คือบริการเว็บเซิร์ฟเวอร์, พอร์ต 53 คือบริการระบบชื่อโดเมน, พอร์ต 1443 คือ บริการเซิร์ฟเวอร์ฐานข้อมูล SQL และพอร์ต 20-21 คือบริการไฟล์เซิร์ฟเวอร์ นอกจากนี้ การโจมตีประเภทนี้ยังกระทำเพื่อหวังผลที่จะทราบชนิดของระบบปฏิบัติการของเครื่องเป้าหมายเพื่อทำการโจมตีไปยังช่องโหว่ของระบบปฏิบัติการนั้นๆ โดยแต่ละระบบปฏิบัติการจะมีการเปิดพอร์ตเฉพาะในการเชื่อมต่อ ตัวอย่างเช่น ระบบปฏิบัติการวินโดวส์จะเปิดให้บริการบนหมายเลขพอร์ต 135-139 และ 443 เป็นต้น โดยการโจมตีประเภทนี้เครื่องเป้าหมายจะไม่สามารถบันทึกล็อกไฟล์ของการเชื่อมต่อไว้ได้ (เพราะไม่เกิดการเชื่อมต่อจริง) เนื่องจากผู้บุกรุกอาศัยช่องโหว่ของกระบวนการทรีเวย์แฮนด์เชคในโพรโทคอลทีซีพี/ไอพี โดยการส่งเพียงแพกเก็ต SYN ไปยังพอร์ตต่างๆของเครื่องเป้าหมายและนับจำนวน SYN/ACK แพกเก็ตที่ตอบกลับจากเครื่องเป้าหมาย จากนั้นจึงส่งแพกเก็ต RST เพื่อรีเซตหรือยกเลิกการเชื่อมต่อนั้น โดยหากบางพอร์ตของเครื่องเป้าหมายมีการเปิดให้บริการก็จะมีการตอบกลับไปยังผู้โจมตีก่อนที่จะมีการยกเลิกการเชื่อมต่อนั้น

ตัวอย่างการเขียนสคริปต์อีสดีเอสแอลในการตรวจจับการโจมตี Port Scanning ดังภาพที่ 76

```
[Rule "Port Scanning" (a,b,c,d)
     [Datagram
          a.desIP = b.desIP,
          b.desIP = c.desIP,
          c.desIP = d.desIP
          [TCP_Packet
               a.desPort = 21, a.syn = 1,
               b.desPort = 80, b.syn = 1,
               c.desPort = 21, c.rst = 1,
               d.desPort = 80, d.rst = 1,
]]]
```

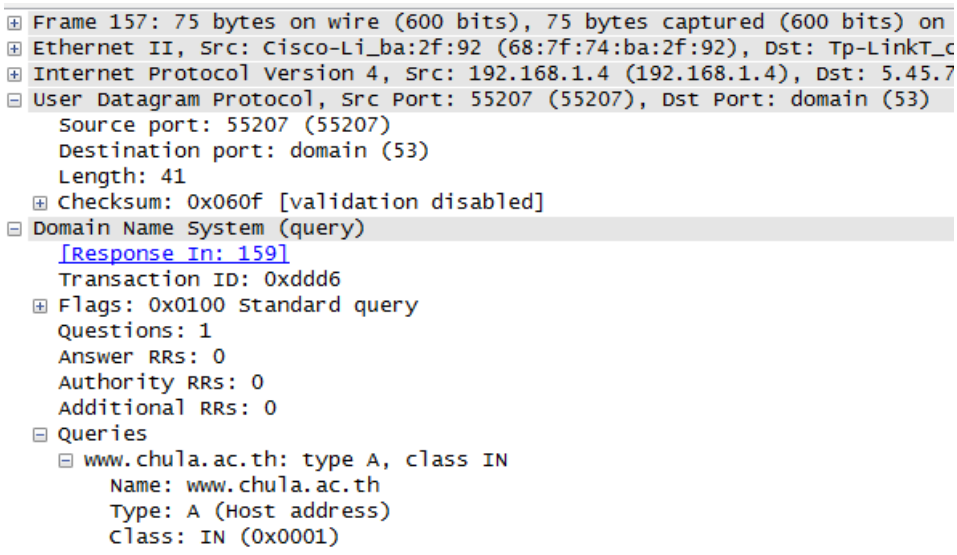
ภาพที่ 76 กฎที่ใช้ในการตรวจจับ Port Scanning

จากภาพที่ 76 เป็นตัวอย่างการเขียนกฎที่ใช้ในการตรวจจับการบุกรุกประเภทไอพีฮาร์ฟสแกน หรือ Port Scanning โดยอาศัยความสัมพันธ์ของค่าคุณลักษณะระหว่างแพกเก็ต 4 แพกเก็ตระหว่าง SYN แพกเก็ตกับ RST แพกเก็ต ซึ่งมีการตรวจสอบหมายเลขพอร์ตคือ 80 และ 21 ที่มีไอพีต้นทางเดียวกันและไอพีปลายทางเดียวกัน อย่างไรก็ตาม การโจมตีประเภทนี้สามารถหลีกเลี่ยงการตรวจจับได้โดยการไม่ส่งแพกเก็ต SYN ในเวลาที่ใกล้เคียงกัน หรือผู้โจมตีอาจใช้เครื่องหลายเครื่องเพื่อทำการโจมตีประเภทนี้ซึ่งทำให้ตรวจจับได้ยากหากใช้เพียงการอธิบายค่าคุณลักษณะระหว่างแพกเก็ตเพียงอย่างเดียว ซึ่งการตรวจจับการโจมตีประเภทนี้จะเป็นแนวทางการวิจัยต่อไปในอนาคต เช่นการเก็บสถานะของกฎ หรือความสัมพันธ์ระหว่างกฎเพื่อให้อีสดีเอสแอลมีความสามารถครอบคลุมการโจมตีประเภทนี้ได้

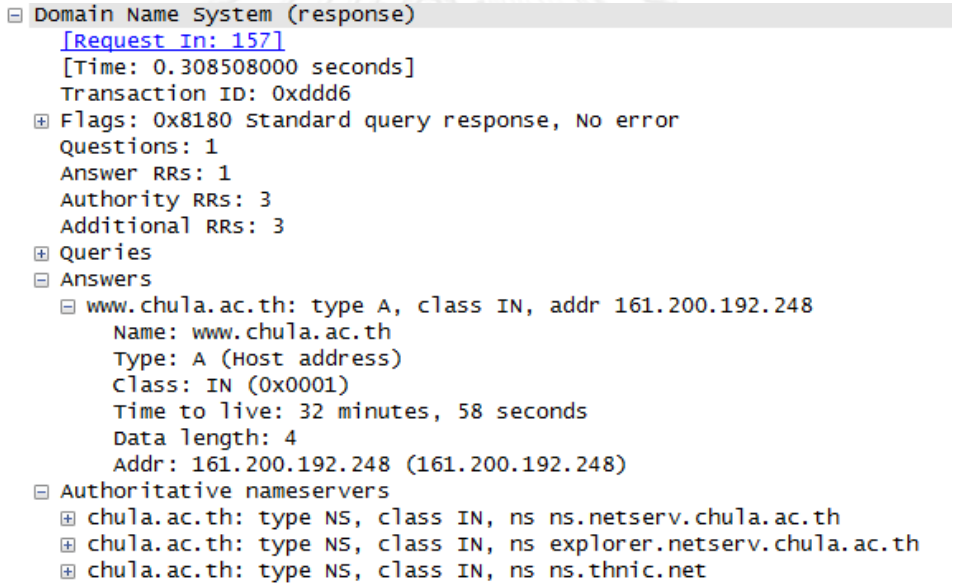
5.3 การประเมินวัดผลความสามารถในการตรวจจับการบุกรุกเครือข่าย

แนวทางการประเมินวัดผลจะทำการทดลองเพื่อทดสอบประสิทธิภาพในการตรวจจับของระบบตรวจจับการบุกรุกเครือข่ายที่ประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรม โดยใช้สคริปต์อีสดีเอสแอลที่เขียนขึ้นโดยผู้เชี่ยวชาญมาใช้ในการตรวจจับการบุกรุกเครือข่ายจากข้อมูลล็อกการจราจรทางเครือข่าย (Network traffic log) ที่สร้างขึ้นเพื่อจำลองการบุกรุกเครือข่ายและใช้ในการทดลองวัดประสิทธิภาพในการตรวจจับของระบบตรวจจับการบุกรุกเครือข่ายโดยรูปแบบการบุกรุกเครือข่ายที่ถูกเลือกเพื่อมาใช้ในการทดลองจะเป็นรูปแบบการโจมตีแบบปลอมแปลง (Spoofing attack) ซึ่งเป็นรูปแบบการโจมตีที่ตรวจจับได้ข้อนข้างยากเนื่องจากการเป็นรูปแบบการโจมตีที่ประกอบด้วยหลายแพกเก็ตโดยได้เลือกการโจมตีแบบปลอมแปลงมาทดลอง 2 รูปแบบคือ

1. การโจมตีแบบปลอมแปลงดีเอ็นเอส (DNS spoofing attack) : เป็นการโจมตีที่มีการปลอมแปลงข้อมูลดีเอ็นเอสเพื่อให้เครื่องเป้าหมายมีการเชื่อมต่อกับเครื่องปลายทางที่ไม่ถูกต้องโดยการเปลี่ยนค่าไอพีแอสเดรสของเครื่องปลายทาง ซึ่งการโจมตีด้วยการปลอมแปลงข้อมูลดีเอ็นเอสอาจเป็นการทำเพื่อขโมยข้อมูล หรือเผยแพร่ข้อมูลที่เป็นภัยคุกคามไปยังเครื่องเป้าหมาย โดยอาศัยกระบวนการทำงานของระบบชื่อโดเมน(Domain name system) ซึ่งเป็นระบบที่รับผิดชอบในการแปลงจากชื่อโดเมนของเครื่องที่ให้บริการให้เป็นไอพีแอดเดรสของเครื่องที่ให้บริการ เช่น ผู้ใช้งานอินเตอร์เนตต้องการเข้าสู่เว็บไซต์ของมหาวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย “www.chula.ac.th” เครื่องของผู้ใช้งานจะทำการร้องขอปยังดีเอ็นเอสเซิร์ฟเวอร์เพื่อถามว่า “www.chula.ac.th” ไอพีแอสเดรสเป็นอะไร จากนั้นดีเอ็นเอสเซิร์ฟเวอร์จะประมวลผลเพื่อค้นหาไอพีแอสเดรสและตอบกลับไปยังเครื่องผู้ใช้งาน หลังจากนั้นเครื่องผู้ใช้งานจะทำการเชื่อมต่อไปยังเครื่องที่ให้บริการ “www.chula.ac.th” ตามไอพีแอดเดรสที่ได้รับจากดีเอ็นเอสเซิร์ฟเวอร์โดยในภาพที่ 77 แสดงกระบวนการร้องขอของดีเอ็นเอสแพกเก็ตที่มีคำสั่งในการร้องขอ (Query) และชื่อโดเมน “www.chula.ac.th” ที่ดักจับด้วยโปรแกรม Wireshake และและภาพที่ 78 แสดงแพกเก็ตตอบกลับจากดีเอ็นเอสเซิร์ฟเวอร์ ซึ่งแสดงหมายเลขไอพีแอสเดรสของเครื่องที่ให้บริการ “www.chula.ac.th”



ภาพที่ 77 รายละเอียดแพกเก็ตดีเอ็นเอสที่เป็นการร้องขอ



ภาพที่ 78 รายละเอียดแพกเก็ตดีเอ็นเอสที่ตอบกลับ

โดยผู้โจมตีอาจทำการโจมตีโดยการปลอมแปลงแพกเก็ตดีเอนเอสที่ตอบกลับ (DNS response packet) และส่งไปยังผู้ที่ทำการร้องขอไปยังดีเอนเอสเซิร์ฟเวอร์ หรือโจมตีการปลอมแปลงแพกเก็ตเพื่อให้เครื่องเป้าหมายหันมาใช้ดีเอ็นเอสเซิร์ฟเวอร์ปลอมที่ผู้โจมตีสร้างขึ้นเป็นต้น โดยในภาพที่ 79 แสดงอีสดีเอสแอลสคริปต์ที่เขียนกฎที่ใช้ในการตรวจจับพฤติกรรมการบุกรุกแบบปลอมแปลงดีเอนเอส หรือ “DNS spoofing”

```
[Rule "DNS spoofing" (a,b,c)
   [Frame
       [Datagram
          c.desIP = a.srcIP,
          b.desIP = a.srcIP
          [UDP_Packet
             a.desPort = 53, b.srcPort = 53, c.srcPort = 53,
             c.desPort = a.srcPort, b.desPort = a.srcPort
             [DNS_Packet
                a.dnsID = b.dnsID, b.dnsID = c.dnsID
    ]]]]
    b != c
]
```

ภาพที่ 79 การโจมตีแบบ DNS spoofing ในรูปของอีสดีเอสแอลสคริปต์

โดยกฎได้อธิยายถึงความสัมพันธ์ของแพกเก็ตที่สนใจ 3 แพกเก็ตตามตัวแปรที่กำหนดในการสร้างกฎ คือ a, b ,และ c โดยแพกเก็ต a เป็นแพกเก็ตที่ถูกส่งมาจากผู้ใช้งานที่ทำการร้องขอไปยังดีเอ็นเอสเซิร์ฟเวอร์ โดยมีแพกเก็ต b และแพกเก็ต c ที่เป็นการตอบกลับจากไอพีแอดเดรส 2 แหล่งซึ่งโดยมีดีเอนเอสไอดี หรือของการร้องขอเดียวกันจึงถือเป็นการโจมตีแบบปลอมแปลงดีเอนเอส

2. การโจมตีแบบปลอมแปลงอาร์พ (Arp spoofing attack) : เป็นการโจมตีที่ผู้โจมตีส่งเพ็คเกจอาร์พที่ปลอมขึ้นไปยังเครือข่าย โดยใส่ข้อมูลแม็คแอดเดรส (Mac address) ของเครื่องผู้โจมตีกับไอพีแอดเดรสของเครื่องเป้าหมาย ทำให้การส่งข้อมูลไปหาเครื่องเป้าหมายถูกเปลี่ยนไปให้ส่งไปยังเครื่องของผู้โจมตีแทน ซึ่งการทำงานโดยปกติของโพรโทคอลอาร์พนั้นจะควบคุมการส่งข้อมูลภายในเครือข่ายท้องถิ่นโดยใช้หมายเลขแม็คแอดเดรสเป็นตัวกำหนดโหนดในการส่งข้อมูลซึ่งจะมีกระบวนการประกาศหรือกระจายแพกเก็ต (ใช้แม็คแอดเดรสปลายทางเท่ากับ FF:FF:FF:FF:FF:FF) ไปในเครือข่ายท้องถิ่นเพื่อถามหาหมายเลขแม็คแอดเดรสของผู้รับที่ตรงกับหมายเลขไอพีแอดเดรสปลายทาง ซึ่งหากเครื่องที่มีหมายเลขไอพีแอดเดรสตรงกับแพกเก็ตอาร์พที่ถามนั้น ก็จะตอบกลับหมายเลขแม็คแอดเดรสไปยังผู้ส่งทำให้การเชื่อมต่อเป็นไปอย่างถูกต้อง การโจมตีประเภทนี้อาจส่งผลให้เกิดการโจมตีแบบคนกลางได้โดยการเปลี่ยนข้อมูลที่ได้รับแล้วส่งต่อไปยังเครื่องเป้าหมาย โดยตัวอย่างของกฎอีสดีเอสแอลที่ใช้ในการตรวจจับการโจมตีแบบปลอมแปลงอาร์พเป็นไปตามภาพที่ 80

```
[Rule "ARP spoofing" (a,b,c)
    [Frame
        a.desMac = "FF:FF:FF:FF:FF:FF",
        b.srcMac != c.SrcMac
        [Datagram
            a.desIP = b.srcIP, a.desIP = c.srcIP,
        ]]]
```

ภาพที่ 80 การโจมตีแบบ ARP spoofing ในรูปของอีสดีเอสแอลสคริปต์

โดยการทดลองมีจุดประสงค์เพื่อต้องการหาประสิทธิภาพของระบบตรวจจับการบุกรุกโดยวัดจากค่าเวลาที่ใช้ในการค้นหากลุ่มของแพกเก็ตที่เป็นพฤติกรรมการบุกรุกของทั้ง 2 รูปแบบการโจมตีที่ถูกเลือกและหาความสัมพันธ์ระหว่างอัตราการกลายพันธุ์และจำนวนตำแหน่งที่หลายพันธุ์ในขั้นตอนวิธีเชิงพันธุกรรมที่ส่งผลต่อเวลาในการตรวจจับของระบบตรวจจับการบุกรุก โดยในการทดลองมีการกำหนดตัวแปรของขั้นตอนวิธีเชิงพันธุกรรมเริ่มต้นดังนี้

1. จำนวนประชากรที่ใช้มีค่าเท่ากับ 19 โครโมโซมในแต่ละรุ่น
2. อัตตราการแทนที่สำหรับการไขว้เปลี่ยนมีค่าเท่ากับ 0.3 ซึ่งทำให้มีโครโมโซม 3 คู่จะถูกเลือกโดยความน่าจะเป็นเพื่อทำการไขว้เปลี่ยน
3. การไขว้เปลี่ยนที่ใช้เป็นประเภทการไขว้เปลี่ยนแบบจุดเดียว โดยทำการไขว้เปลี่ยนที่ตำแหน่งแรกของโครโมโซม

ในตารางที่ 16 และตารางที่ 17 แสดงผลของเวลาเฉลี่ยที่ใช้ในการตรวจจับการบุกรุก (Detection time)ของรูปแบบการบุกรุกทั้ง 2 รูปแบบ โดยเฉลี่ยจากการเวลาที่ใช้ในการตรวจจับการบุกรุกจำนวน 100 ค่า (ทำการทดลองตรวจจับการบุกรุกซ้ำ 100 ครั้งในแต่ละรูปแบบการโจมตี)

ตารางที่ 16 ค่าเวลาเฉลี่ยที่ใช้ในการตรวจจับการโจมตีแบบปลอมแปลงอาร์พ (ในหน่วยวินาที)

อัตราการกลายพันธุ์	**จำนวนตำแหน่งของการกลายพันธุ์**		
	1	2	3
0.1	0.1181	0.2166	**0.2724**
0.2	**0.0959**	0.1667	0.3496
0.3	0.1115	**0.1606**	0.3698
0.4	0.1220	0.2007	0.3057
0.5	0.1234	0.2216	0.3405
0.6	0.1295	0.1952	0.3399

ตารางที่ 17 ค่าเวลาเฉลี่ยที่ใช้ในการตรวจจับการโจมตีแบบปลอมแปลงดีเอ็นเอส (ในหน่วยวินาที)

อัตราการกลายพันธุ์	**จำนวนตำแหน่งของการกลายพันธุ์**		
	1	2	3
0.1	0.4656	0.4341	0.4985
0.2	**0.4505**	**0.3557**	0.5367
0.3	0.4607	0.4197	**0.4758**
0.4	0.4692	0.4251	0.5230
0.5	0.4995	0.4206	0.5063
0.6	0.5388	0.4225	0.5399

จากการทดลองพบว่าอัตราการกลายพันธุ์ที่เหมาะสมที่สุดมีค่าเท่ากับ 0.2 ซึ่งส่งผลให้ค่าของเวลาที่ใช้ในการตรวจจับการบุกรุกน้อยที่สุด โดยจากการทดลองค่าเวลาเฉลี่ยที่ใช้ในการตรวจจับการบุกรุกของการโจมตีแบบปลอมแปลงอาร์พที่น้อยที่สุดมีค่าเท่ากับ 0.0959 วินาที และค่าเวลาเฉลี่ยที่ใช้ในการตรวจจับการบุกรุกของการโจมตีแบบปลอมแปลงดีเอ็นเอสที่น้อยที่สุดมีค่าเท่ากับ 0.3557 วินาที โดยจากตารางผลการทดลองพบว่าในค่าอัตราการกลายพันธุ์หนึ่งๆ การเลือก 3 ตำแหน่งในการกลายพันธุ์จะทำให้เวลาที่ใช้ในการตรวจจับแย่ที่สุดเมื่อเปรียบเทียบกับการใช้ 1 หรือ 2 ตำแหน่งในการกลายพันธุ์ แต่เนื่องจากกฎที่ใช้ในการตรวจจับการโจมตีแบบปลอมแปลงดีเอ็นเอสมีจำนวนเงื่อนไขที่ใช้ในการตรวจสอบแพกเก็ตที่สนใจมีจำนวนมากกว่าจำนวนเงื่อนไขของกฎที่ใช้ในการการโจมตีแบบปลอมแปลงอาร์พทำให้กระบวนการประเมินค่าความเหมาะสมของการโจมตีแบบปลอมแปลงดีเอ็นเอสใช้เวลานานกว่า ซึ่งการเพิ่มจำนวนตำแหน่งการกลายพันธุ์จาก 1 เป็น 2 ทำให้การเปลี่ยนแปลงรูปแบบของโครโมโซมให้เป็นไปอย่างรวดเร็วขึ้น ส่งผลให้ค่าเวลาเฉลี่ยที่ใช้ในการตรวจจับการโจมตีแบบปลอมแปลงดีเอ็นเอสดีขึ้น

บทที่ 6
สรุปผลการวิจัย และข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอการสร้างภาษาจำเพาะโดเมนเพื่อการตรวจจับการบุกรุกเครือข่ายเพื่อให้ให้ผู้เชี่ยวชาญสามารถสร้างกฎที่ที่ใช้ในการตรวจจับพฤติกรรมการบุกรุกเครือข่ายได้โดยง่าย และสามารถอธิบายรูปแบบการบุกรุกที่เกิดจากความสัมพันธ์ของค่าคุณลักษณะระหว่างแพกเก็ต พร้อมกับการสร้างเครื่องมือที่ช่วยในการแก้ไขภาษาที่สามารถแจ้งเตือนเมื่อมีข้อผิดพลาดเบื้องต้นได้ นอกจากนี้การประยุกต์ใช้ขั้นตอนวิธีเชิงพันธุกรรมในการค้นหากลุ่มของแพกเก็ตที่ตรงกับกฎของอีสดีเอสแอลทำให้สามารถที่จะตรวจสอบรูปแบบกลุ่มของแพกเก็ตเพื่อหาพฤติกรรมการบุกรุกได้หลายรูปแบบพร้อมๆกัน แทนที่จะตรวจสอบครั้งละรูปแบบซึ่งแตกต่างจากขั้นตอนวิธีอื่นๆ โดยระบบตรวจจับการบุกรุกเครือข่ายที่สร้างขึ้นสามารถแสดงให้เห็นถึงประสิทธิภาพในการตรวจจับของแนวความคิดของงานวิจัยนี้ดังที่แสดงในบทที่ 5

6.2 ข้อจำกัด

1. อีสดีเอสแอลยังไม่สามารถอธิบายรูปแบบพฤติกรรมการบุกรุกเครือข่ายที่มีลักษณะหรือค่าคุณลักษณะของแพกเก็ตที่นอกเหนือขอบเขตของอีสดีเอสแอลได้
2. ระบบตรวจจับการบุกรุกที่พัฒนาขึ้นจำเป็นต้องใช้งานกับการ์ดเครือข่ายที่สนับสนุนการทำงานแบบ promiscuous โหมดเพื่อให้สามารถดักจับแพกเก็ตได้
3. ส่วนที่ใช้ในการแก้ไขอีสดีเอสแอลที่พัฒนาขึ้นยังไม่สามารถตรวจสอบข้อผิดพลาดที่มากไปกว่าการพิมพ์ผิดวากยสัมพันธ์ของภาษาอีสดีเอสแอลได้
4. ระบบตรวจจับการบุกรุกเครือข่ายที่พัฒนาขึ้นเป็นระบบต้นแบบที่ยังไม่สามารถใช้กับสถานการณ์เครือข่ายที่มีปริมาณการใช้งานที่สูงได้

6.3 แนวทางการวิจัยต่อ

1. เพิ่มขอบเขตของลำดับชั้นของโพรโทคอลทีซีพี/ไอพี ให้ครอบคลุมไปยังชั้นอื่นๆ เช่น โพรโทคอลเฮชทีทีพี (HTTP)
2. พัฒนาฟังก์ชันควบคุม และเครื่องหมายดำเนินการเพิ่มเติม เพื่อให้สามารถอธิบายกฎการตรวจจับให้มีความครอบคลุมมากยิ่งขึ้น

รายการอ้างอิง

1. Ghosh, D., *DSLs in Action*. 2011, Stamford, CT 06901, USA: Meaning Publications Co.
2. Roesch, M. *Snort-Lightweight Intrusion Detection for Networks*. 1999. Seattle, Washington, USA.
3. Paxson, V. *Bro: A System for Detecting Network Intruders in Real-Time*. 1998. San Antonio.
4. Patel, P., et al., *Network Intrusion Detection Types and Computation.* International Journal of Computer Science and Information Security, 2012. **10**(1).
5. Jaiganesh, V., S. Mangayarkarasi, and P. Sumathi, *Intrusion Detection Systems: A Survey and Analysis of Classification Techniques.* International Journal of Advanced Research in Computer and Communication Engineering, 2013. **2**.
6. Raju, P.N., *State-of-the-art intrusion detection: Technology, challenges, and evaluation*. 2005: Linkoping.
7. Comer, D.E., *Internetworking with TCP/IP Principles, Protocols, and Architectures*. 4 ed. 2002.
8. Almanei, S., et al., *TCP/IP protocol Possible Attack*, in *ECE 478/578 Computer and Network Security, Spring Term*. 2000: Oregon State University.
9. Alander, J.T. *An indexed bibliography of genetic algorithms: Years 1957 -1993*. 1994.
10. Salgueiro, P., et al., *Using Constraints for Intrusion Detection: The Ne-MODe System*, in *PADL*. 2011. p. 115-129.
11. Malik, H., H. Hemmati, and A.E. Hassan. *2013. Automatic detection of performance deviations in the load testing of large scale systems*. in *the 2013 International Conference on Software Engineering(ICSE '13)*. 2013. Piscataway, NJ, USA: IEEE Press.
12. Ryan, J., M.-J. Lin, and R. Miikkulainen *Intrusion Detection with Neural Network*. 1997.
13. *Irony .NET Language Implementation Kit*. 2013 September 2013; Available from: http://irony.codeplex.com/.
14. *Pcap .Net*. 2012 April 2012; Available from: http://pcapdotnet.codeplex.com/.

ภาคผนวก

ภาคผนวก ก.

ไวยากรณ์ของอีสดีเอสแอล

```
[Language("isDSL", "0.9", "Intrusion signature grammar")]
    class isDSLGrammar : Grammar
    {
        public isDSLGrammar()
            : base(true) //isDSL is case sensitive
        {
            // Terminal
            KeyTerm comma = ToTerm(",", "comma");
            KeyTerm dot = ToTerm(".", "dot");

            var identifier =
TerminalFactory.CreateisDSLIdentifier("identifier");
            var name = TerminalFactory.CreateisDSLRuleName("name");
            var value = TerminalFactory.CreateisDSLString("value");
            var number = TerminalFactory.CreateisDSLNumber("number");

            var Rule = ToTerm("Rule ");
            var Frame = ToTerm("Frame ");
            var Datagram = ToTerm("Datagram ");
            var TCP_Packet = ToTerm("TCP_Packet ");
            var UDP_Packet = ToTerm("UDP_Packet ");
            var DNS_Packet = ToTerm("DNS_Packet");
            var srcMac = ToTerm("srcMac");
            var desMac = ToTerm("desMac");
            var protocolType = ToTerm("protocolType");
            var arpOp = ToTerm("arpOp");

            var srcIP = ToTerm("srcIP");
            var desIP = ToTerm("desIP");
            var ttl = ToTerm("ttl");
            var fragment = ToTerm("fragment");
            var protocol = ToTerm("protocol");
            var lenght = ToTerm("lenght");
            var version = ToTerm("version");
            var id = ToTerm("id");

            var srcPort = ToTerm("srcPort");
            var desPort = ToTerm("desPort");
            var ackNumber = ToTerm("ackNumber");
            var seqNumber = ToTerm("seqNumber");
            var nexSeqNumber = ToTerm("nexSeqNumber");
            var winSize = ToTerm("winSize");
            var fin = ToTerm("fin");
            var syn = ToTerm("syn");
            var rst = ToTerm("rst");
            var psh = ToTerm("psh");
            var ack = ToTerm("ack");
            var ece = ToTerm("ece");
            var cwr = ToTerm("cwr");
            var urg = ToTerm("urg");
            var res = ToTerm("res");
            var ns = ToTerm("ns");
```

```
var dnsId = ToTerm("dnsId");
var dnsOp = ToTerm("dnsOp");

var Count = ToTerm("Count");
var Sequence = ToTerm("Sequence");

// Non terminal
var rule = new NonTerminal("rule");
var ruleName = new NonTerminal("ruleName");
var paramList = new NonTerminal("paramList");
var idList = new NonTerminal("idList");
var conList = new NonTerminal("conList");
var tcpStackCon = new NonTerminal("tcpStackCon");

var frameCon = new NonTerminal("frameCon");
var datagramCon = new NonTerminal("datagramCon");
var tcpPacketCon = new NonTerminal("tcpPacketCon");
var udpPacketCon = new NonTerminal("udpPacketCon");
var dnsPacketCon = new NonTerminal("dnsPacketCon");

var frameConList = new NonTerminal("frameConList");
var datagramConList = new NonTerminal("datagramConList");
var tcpPacketConList = new NonTerminal("tcpPacketConList");
var udpPacketConList = new NonTerminal("udpPacketConList");
var dnsPacketConList = new NonTerminal("dnsPacketConList");
var controlFunctionList = new NonTerminal("controlFunctionList");

// Frame
var frameConStmt = new NonTerminal("frameConStmt");
var f_Mac = new NonTerminal("f_Mac");
var f_MacCon = new NonTerminal("f_MacCon");
var f_ProtocolTypeCon = new NonTerminal("f_ProtocolTypeCon");
var f_ArpOpCon = new NonTerminal("f_ArpsOpCon");

// Datagram
var datagramConStmt = new NonTerminal("datagramConStmt");
var d_Ip = new NonTerminal("d_Ip");
var d_IpCon = new NonTerminal("d_IpCon");
var d_TtlCon = new NonTerminal("d_TtlCon");
var d_FragmentCon = new NonTerminal("d_fragmentCon");
var d_ProtocolCon = new NonTerminal("d_ProtocolCon");
var d_LenghtCon = new NonTerminal("d_LenghtCon");
var d_VersionCon = new NonTerminal("d_VersionCon");
var d_IdCon = new NonTerminal("d_IdCon");
var ipValue = new NonTerminal("ipValue");

// TCP_Packet
var tcpPacketConStmt = new NonTerminal("tcpPacketConStmt");
var t_Port = new NonTerminal("t_Port");
var t_Flag = new NonTerminal("t_Flag");
var t_PortCon = new NonTerminal("t_PortCon");
var t_AckNumberCon = new NonTerminal("t_AckNumberCon");
var t_SeqNumberCon = new NonTerminal("t_SeqNumberCon");
var t_NexSeqNumberCon = new NonTerminal("t_NexSeqNumberCon");
var t_WinSizeCon = new NonTerminal("t_WinSizeCon");
var t_FlagCon = new NonTerminal("t_FlagCon");

// UDP_Packet
var udpPacketConStmt = new NonTerminal("udpPacketConStmt");
```

```
            var u_Port = new NonTerminal("u_Port");
            var u_LenghtCon = new NonTerminal("u_LenghtCon");
            var u_PortCon = new NonTerminal("u_PortCon");

            // DNS_Packet
            var dnsPacketConStmt = new NonTerminal("dnsPacketConStmt");
            var d_DnsIdCon = new NonTerminal("d_DnsIdCon");
            var d_DnsOpCon = new NonTerminal("d_DnsOpCon");

            // Control Function
            var controlFuncStmt = new NonTerminal("controlFuncStmt");
            var countFunction = new NonTerminal("countFunction");
            var seqFunction = new NonTerminal("seqFunction");
            var seqParamList = new NonTerminal("seqParamList");
            var packetComparer = new NonTerminal("packetComparer");

            var biOp = new NonTerminal("biOp");
            var inOp = new NonTerminal("inOp");


            // Rule
            this.Root = rule;
            rule.Rule = "[" + ruleName + paramList + conList + "]";
            ruleName.Rule = Rule + name;
            paramList.Rule = "(" + idList + ")";
            idList.Rule = MakeStarRule(idList, comma, identifier);

            conList.Rule = tcpStackCon + controlFunctionList;
            tcpStackCon.Rule = "[" + Frame + frameCon + "]" | "[" + Datagram +
datagramCon + "]" | "[" + TCP_Packet + tcpPacketCon + "]" | "[" + UDP_Packet +
udpPacketCon + "]" | "[" + DNS_Packet + dnsPacketCon + "]";

            frameCon.Rule = frameConList | frameConList + "[" + Datagram +
datagramCon + "]";
            datagramCon.Rule = datagramConList | datagramConList + "[" +
TCP_Packet + tcpPacketCon + "]" | datagramConList + "[" + TCP_Packet +
tcpPacketCon + "]" + "[" + UDP_Packet + udpPacketCon + "]" | datagramConList +
"[" + UDP_Packet + udpPacketCon + "]";
            tcpPacketCon.Rule = tcpPacketConList;
            udpPacketCon.Rule = udpPacketConList | udpPacketConList + "[" +
DNS_Packet + dnsPacketCon + "]";
            dnsPacketCon.Rule = dnsPacketConList;

            frameConList.Rule = MakeStarRule(frameConList, comma,
frameConStmt);
            datagramConList.Rule = MakeStarRule(datagramConList, comma,
datagramConStmt);
            tcpPacketConList.Rule = MakeStarRule(tcpPacketConList, comma,
tcpPacketConStmt);
            udpPacketConList.Rule = MakeStarRule(udpPacketConList, comma,
udpPacketConStmt);
            dnsPacketConList.Rule = MakeStarRule(dnsPacketConList, comma,
dnsPacketConStmt);
            controlFunctionList.Rule = MakeStarRule(controlFunctionList, comma,
controlFuncStmt);

            // Frame Rule
            frameConStmt.Rule = f_MacCon | f_ProtocolTypeCon | f_ArpOpCon;
            f_Mac.Rule = srcMac | desMac;
```

```
            f_MacCon.Rule = identifier + dot + f_Mac + biOp + identifier + dot
+ f_Mac |
                            identifier + dot + f_Mac + biOp + value;
            f_ProtocolTypeCon.Rule = identifier + dot + protocolType + biOp +
identifier + dot + protocolType |
                                     identifier + dot + protocolType + biOp +
value;
            f_ArpOpCon.Rule = identifier + dot + arpOp + biOp + identifier +
dot + arpOp |
                              identifier + dot + arpOp + biOp + value;

            // Datagram Rule
            datagramConStmt.Rule = d_IpCon | d_TtlCon | d_FragmentCon |
d_ProtocolCon | d_LenghtCon | d_VersionCon | d_IdCon;
            d_Ip.Rule = srcIP | desIP;
            ipValue.Rule = value;
            d_IpCon.Rule = identifier + dot + d_Ip + biOp + identifier + dot +
d_Ip |
                           identifier + dot + d_Ip + biOp + ipValue |
                           identifier + dot + d_Ip + inOp + ipValue;
            d_TtlCon.Rule = identifier + dot + ttl + biOp + identifier + dot +
ttl |
                            identifier + dot + ttl + biOp + value;
            d_FragmentCon.Rule = identifier + dot + fragment + biOp +
identifier + dot + fragment |
                                 identifier + dot + fragment + biOp + value;
            d_ProtocolCon.Rule = identifier + dot + protocol + biOp +
identifier + dot + protocol |
                                 identifier + dot + protocol + biOp + value;
            d_LenghtCon.Rule = identifier + dot + lenght + biOp + identifier +
dot + lenght |
                               identifier + dot + lenght + biOp + value;
            d_VersionCon.Rule = identifier + dot + version + biOp + identifier
+ dot + version |
                                identifier + dot + version + biOp + value;
            d_IdCon.Rule = identifier + dot + id + biOp + identifier + dot + id
|
                           identifier + dot + id + biOp + value;

            // TCP Rule
            tcpPacketConStmt.Rule = t_PortCon | t_FlagCon | t_AckNumberCon |
t_SeqNumberCon | t_NexSeqNumberCon | t_WinSizeCon;
            t_Port.Rule = srcPort | desPort;
            t_Flag.Rule = fin | syn | rst | psh | ack | ece | cwr | urg | res |
ns;
            t_PortCon.Rule = identifier + dot + t_Port + biOp + identifier +
dot + t_Port |
                             identifier + dot + t_Port + biOp + number;
            t_FlagCon.Rule = identifier + dot + t_Flag + biOp + identifier +
dot + t_Flag |
                             identifier + dot + t_Flag + biOp + number;
            t_AckNumberCon.Rule = identifier + dot + ackNumber + biOp +
identifier + dot + ackNumber |
                             identifier + dot + ackNumber + biOp + number;
            t_SeqNumberCon.Rule = identifier + dot + seqNumber + biOp +
identifier + dot + seqNumber |
                             identifier + dot + seqNumber + biOp + number;
            t_NexSeqNumberCon.Rule = identifier + dot + nexSeqNumber + biOp +
identifier + dot + nexSeqNumber |
```

```
                                identifier + dot + nexSeqNumber + biOp + number;
            t_WinSizeCon.Rule = identifier + dot + winSize + biOp + identifier
+ dot + winSize |
                                identifier + dot + winSize + biOp + number;


            // UDP Rule
            udpPacketConStmt.Rule = u_PortCon | u_LenghtCon;
            u_Port.Rule = srcPort | desPort;
            u_PortCon.Rule = identifier + dot + u_Port + biOp + identifier +
dot + u_Port |
                                identifier + dot + u_Port + biOp + number;
            u_LenghtCon.Rule = identifier + dot + lenght + biOp + identifier +
dot + lenght |
                                identifier + dot + lenght + biOp + value;


            // DNS Rule
            dnsPacketConStmt.Rule = d_DnsIdCon | d_DnsOpCon;
            d_DnsIdCon.Rule = identifier + dot + dnsId + biOp + identifier +
dot + dnsId |
                                identifier + dot + dnsId + biOp + number;
            d_DnsOpCon.Rule = identifier + dot + dnsOp + biOp + identifier +
dot + dnsOp |
                                identifier + dot + dnsOp + biOp + value;


            // Control Function Rule
            controlFuncStmt.Rule = countFunction | seqFunction |
packetComparer;
            countFunction.Rule = Count + "(" + identifier + comma + number +
")";
            seqParamList.Rule = MakePlusRule(seqParamList, comma, identifier);
            seqFunction.Rule = Sequence + "(" + seqParamList + ")";
            packetComparer.Rule = identifier + biOp + identifier;

            biOp.Rule = ToTerm("=") | ToTerm("!=");
            inOp.Rule = ToTerm("in") | ToTerm("!in");
            RegisterOperators(0, "=", "!=");

            MarkPunctuation("[", "]", "(", ")");
            RegisterBracePair("(", ")");
            RegisterBracePair("[", "]");

        }
    }
```

ภาพที่ 81 ไวยากรณ์ของอีสดีเอสแอล

ภาคผนวก ข.

ไอโรนี (Irony - .NET Language Implementation Kit.)

หลักการทำงาน

ไอโรนีเป็นชุดพัฒนาสำหรับสร้างภาษาโดเมนจำเพาะบนแพลตฟอร์มดอทเน็ตเฟรมเวิร์ค (.NET Framework) [13] ซึ่งมีความแตกต่างจากชุดพัฒนาอื่นๆ โดยไอโรนีจะสามารถนิยามไวยกรณ์หรือวากยสัมพันธ์ได้โดยใช้ภาษาซีชาร์ปโดยตรง รวมถึงโอเปอเรเตอร์ต่างๆที่อยู่ในภาษาซีชาร์ปก็สามารถนำมาทำให้เกิดผลใหม่ได้ โดยในการสร้างตัวแจงส่วนมีขั้นตอนการสร้างตัวแจงส่วนดังตัวอย่างด้านล่างตามภาพที่ 82, ภาพที่ 83, ภาพที่ 84, ภาพที่ 85,และภาพที่ 86 ตามลำดับ

```
using System;
using System.Collections.Generic;
using System.Text;
using Irony.Parsing;
using Irony.Ast;

namespace Irony.Samples {
  // This grammar describes programs that consist of simple expressions and assignments
  // for ex:
  // x = 3
  // y = -x + 5
  //  the result of calculation is the result of last expression or assignment.
  //  Irony's default  runtime provides expression evaluation.
  //  supports inc/dec operators (++,--), both prefix and postfix,
  //  and combined assignment operators like +=, -=, etc.

  [Language("ExpressionEvaluator", "1.0", "Multi-line expression evaluator")]
  public class ExpressionEvaluatorGrammar : Irony.Parsing.Grammar {
    public ExpressionEvaluatorGrammar() {

      // 1. Terminals
      var number = new NumberLiteral("number");
      //Let's allow big integers (with unlimited number of digits):
      number.DefaultIntTypes = new TypeCode[] { TypeCode.Int32, TypeCode.Int64,
NumberLiteral.TypeCodeBigInt };
      var identifier = new IdentifierTerminal("identifier");
      var comment = new CommentTerminal("comment", "#", "\n", "\r");
```

ภาพที่ 82 การประกาศคลาสภาษาจำเพาะโดเมนของไอโรนี

```
      //comment must to be added to NonGrammarTerminals list; it is not used directly in
grammar rules,
      // so we add it to this list to let Scanner know that it is also a valid terminal.
      base.NonGrammarTerminals.Add(comment);

      // 2. Non-terminals
      var Expr = new NonTerminal("Expr");
      var Term = new NonTerminal("Term");
      var BinExpr = new NonTerminal("BinExpr", typeof(BinExprNode));
      var ParExpr = new NonTerminal("ParExpr");
      var UnExpr = new NonTerminal("UnExpr", typeof(UnExprNode));
      var UnOp = new NonTerminal("UnOp");
      var BinOp = new NonTerminal("BinOp", "operator");
      var PostFixExpr = new NonTerminal("PostFixExpr", typeof(UnExprNode));
      var PostFixOp = new NonTerminal("PostFixOp");
      var AssignmentStmt = new NonTerminal("AssignmentStmt", typeof(AssigmentNode));
      var AssignmentOp = new NonTerminal("AssignmentOp", "assignment operator");
      var Statement = new NonTerminal("Statement");
      var ProgramLine = new NonTerminal("ProgramLine");
      var Program = new NonTerminal("Program", typeof(StatementListNode));
```

ภาพที่ 83 การประกาศตัวแปรเทอมของไอโรนี

```
      // 3. BNF rules
      Expr.Rule = Term | UnExpr | BinExpr | PostFixExpr;
      Term.Rule = number | ParExpr | identifier;
      ParExpr.Rule = "(" + Expr + ")";
      UnExpr.Rule = UnOp + Term;
      UnOp.Rule = ToTerm("+") | "-" | "++" | "--";
      BinExpr.Rule = Expr + BinOp + Expr;
      BinOp.Rule = ToTerm("+") | "-" | "*" | "/" | "**";
      PostFixExpr.Rule = Term + PostFixOp;
      PostFixOp.Rule = ToTerm("++") | "--";
      AssignmentStmt.Rule = identifier + AssignmentOp + Expr;
      AssignmentOp.Rule = ToTerm("=") | "+=" | "-=" | "*=" | "/=";
      Statement.Rule = AssignmentStmt | Expr | Empty;
      ProgramLine.Rule = Statement + NewLine;
      Program.Rule = MakeStarRule(Program, ProgramLine);
      this.Root = Program;       // Set grammar root
```

ภาพที่ 84 การนิยามกฎให้กับวากยสัมพันธ์ในไอโรนี

```
      // 4. Operators precedence
      RegisterOperators(1, "+", "-");
      RegisterOperators(2, "*", "/");
      RegisterOperators(3, Associativity.Right, "**");
```

ภาพที่ 85 การลงทะเบียนโอเปอร์เรเตอร์ของไอโรนี

```
      // 5. Punctuation and transient terms
      RegisterPunctuation("(", ")");
      RegisterBracePair("(", ")");
      MarkTransient(Term, Expr, Statement, BinOp, UnOp, PostFixOp, AssignmentOp,
ProgramLine, ParExpr);
```

ภาพที่ 86 ตัวอย่างการประกาศคู่สัญลักษณ์ในไอโรนี

ภาคผนวก ค.

ตัวอย่างและค่าที่เป็นไปได้ของค่าคุณลักษณะในอีสดีเอสแอล

รายละเอียดของค่าคุณลักษณะที่ใช้ในการเขียนเงื่อนไขของอีสดีเอสแอล เพื่ออธิบายความสัมพันธ์ของค่าคุณลักษณะของแพกเก็ตแสดงดังตารางที่ 18

ตารางที่ 18 รายละเอียดตัวอย่างและค่าที่เป็นไปได้ของค่าคุณลักษณะในเอสดีเอสแอล

เทอร์มินอล	แทนคุณลักษณะ	ตัวอย่างค่าคุณลักษณะหรือค่าที่เป็นไปได้
srcMac	Source Mac address	ตัวอย่าง a.srcMac = “AA:AA:AA:AA:AA:AA”
desMac	Destination Mac Address	ตัวอย่าง a.desMac = “FF:FF:FF:FF:FF:FF”
protocolType	Protocol type	ค่าที่เป็นไปได้ “Arp”, “Tcp”, “Udp”, “Dns”, “Ipv4”
arpOp	Arp Operation	ค่าที่เป็นไปได้ “Request”, “Reply”
srcIP	Source IP address	ตัวอย่าง a.srcIP = “127.0.0.1”, b.src in “192.168.0.1/24”
desIP	Destination IP address	ตัวอย่าง a.desIP = “127.0.0.1”, ”, b.desIP in “192.168.0.1/24”
ttl	Time to Live	ตัวอย่าง a.ttl = 10, b.ttl = 123
fragment	Fragmentation Flag	ค่าที่เป็นไปได้ “None”, “MoreFragments”, “DoNotFragment”
protocol	Protocol field	ค่าที่เป็นไปได้ “Tcp”, “Udp”, “Dns”
length	Total length	ตัวอย่าง a.length = 250
version	Version	ค่าที่เป็นไปได้ 4, 6
id	Identification	ตัวอย่าง a.id = 112357856
srcPort	Source Port	ตัวอย่าง a.srcPort = 6113
desPort	Destination port	ตัวอย่าง a.desPort = 8080
ackNumber	Acknowledgement number	ตัวอย่าง a.ackNumber = “123456789”
seqNumber	Sequence Number	ตัวอย่าง a.seqNumber = “123456789”
nextSeqNunber	Next sequence number	ตัวอย่าง a.nextSeqNunber = “123456789”
winSize	Window size	ตัวอย่าง a.winSize = 25
fin	FIN flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
syn	SYN flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
rst	RST flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)

psh	PSH flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
ack	ACK flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
ece	ECE flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
cwr	CWR flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
urg	URG flag	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
res	Reserved field	ค่าที่เป็นไปได้ ค่าบิต (0 หรือ 1)
dnsID	DNS identification	ตัวอย่าง a.dnsID = 123456789
dnsOp	DNS Operation	ค่าที่เป็นไปได้ “Query”, “IQuery”, “Status”, “Notify”, “Update”

ประวัติผู้เขียนวิทยานิพนธ์

นายคณิน โชติวรรักษ์ เกิดเมื่อวันที่ 15 ธันวาคม พ.ศ. 2532 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต (วศ.บ.) สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2554 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2555