Spring 2019

Mirai Bot Scanner Summation Prototype

Charles V. Frank Jr.
Dakota State University

Follow this and additional works at: https://scholar.dsu.edu/theses

Part of the Databases and Information Systems Commons, Information Security Commons, Systems Architecture Commons, and the Theory and Algorithms Commons

Recommended Citation

This Dissertation is brought to you for free and open access by Beadle Scholar. It has been accepted for inclusion in Masters Theses & Doctoral Dissertations by an authorized administrator of Beadle Scholar. For more information, please contact repository@dsu.edu.
MIRAI BOT SCANNER SUMMATION PROTOTYPE

A dissertation submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Cyber Operations

March 2019

By

Charles V. Frank, Jr.

Dissertation Committee:

Dr. Wayne Pauli, Faculty Mentor and Chairman
Dr. Kyle Cronin, Committee Member
Dr. Stefani K. Hobratsch, Committee Member
Dr. Austin O’Brien, Committee Member
Dr. Yong Wang, Committee Member
This dissertation is approved as a credible and independent investigation by a candidate for the Doctor of Philosophy degree and is acceptable for meeting the dissertation requirements for this degree. Acceptance of this dissertation does not imply that the conclusions reached by the candidate are necessarily the conclusions of the major department or university.

Student Name: Charles V. Frank Jr.

Dissertation Title: Mirai Bot Scanner Summation Prototype

Dissertation Chair/Co-Chair: [Signature]
Date: 03/27/19

Committee member: [Signature]
Date: 03/27/19

Committee member: [Signature]
Date: 03/27/19

Committee member: [Signature]
Date: 03/27/19

Committee member: [Signature]
Date: 03/27/19

Original to Office of Graduate Studies and Research
Acid-free copies with written reports to library
ACKNOWLEDGMENT

A four-year journey with many technical twists and turns as well as many sleepless weekends. The Chinese philosopher Lao-Tzu suggests, “A journey starts with a single step.” I would like to thank my colleagues and classmates Corey Nance and Samuel Jarocki for working with me to co-author our first research paper concerning the Mirai botnet. Working with Corey and Samuel helped to peak my interest in Mirai. It has been indispensable for me to have a group of friends to discuss research ideas with. Dr. Wayne Pauli has been a tremendous chairman to help and guide me thru this dissertation journey. He has provided needed guidance and has suggested the members for the dissertation committee. My dissertation committee has been tremendous in providing timely feedback and suggestions. Dr. Kyle Cronin, Dr. Stefani Hobratsch, Dr. Austin O’Brien, and Dr. Yong Wang thank you for being on my committee. I would like to thank all my other professors. Without you I could have not completed this dissertation. The knowledge imparted to me by all my professors at DSU has prepared me well. Not only have I learned from the course work, but I have learned an enormous amount from fellow classmates. Truly, DSU has brilliant students! To my beautiful wife Michele, thank you for understanding the effort and time required for this doctorate. My beautiful daughter Shavaughn, you inspire me with your zest for life. Shavaughn has taught me the importance for exploring and just “going for it”. To my two sons, Joseph and Jack, it has been an honor seeing you grow into young men. My children inspire me to be the best version of me. To the greatest parents ever, Charles V. Frank Sr. and Martha Frank. No matter how difficult the endeavor, I always felt strength from your support. To my uncle Johnny who is constantly checking up on my well-being. Without the meals and snacks, I have no clue how I would have made it this far. When I look to heaven to my father, Charles V. Frank Sr., I feel there is nothing I can not accomplish.
ABSTRACT

The Mirai botnet deploys a distributed mechanism with each Bot continually scanning for a potential new Bot Victim. A Bot continually generates a random IP address to scan the network for discovering a potential new Bot Victim. The Bot establishes a connection with the potential new Bot Victim with a Transmission Control Protocol (TCP) handshake. The Mirai botnet has recruited hundreds of thousands of Bots. With 100,000 Bots, Mirai Distributed Denial of Service (DDoS) attacks on service provider Dyn in October 2016 triggered the inaccessibility to hundreds of websites in Europe and North America (Sinanović & Mrdovic, 2017). A month before the Dyn attack, the source code was released publicly on the Internet and Mirai spread to half a million bots. Hackers offered Mirai botnets for rent with 400,000 Bots. Recent research has suggested network signatures for Mirai detection. Network signatures are suggested to detect a Bot brute forcing a new Bot Victim with a factory default user-id and password. Research has not been focused on the Bot scanning mechanism. The focus of this research is performing experimentation to analyze the Bot scanning mechanism for when a Bot attempts to establish a connection to a potential new Bot Victim with a TCP handshake. The thesis is presented: it is possible to develop a solution that can analyze network traffic to identify a Bot scanning for a potential new Bot Victim.

The three research questions are (a) Can the Bots be identified for summation? (b) Can the potential new Bot Victims be identified for summation? (c) Is it possible to monitor the Bot scanning mechanism over time? The research questions support the thesis. The Design Science Research (DSR) methodology is followed for designing and evaluating the solution presented in this study. The original Mirai Bot code is used as a research data source to perform a Bot scanner code review. A dataset containing Bot scanning network activity, recorded by the University of Southern California (USC), is utilized as the research data source for experimentation performed with the Mirai Bot Scanner Summation Prototype solution. The Bot scanner code review is performed to identify the Bot scanning functionality and network communications with a potential new Bot Victim. A sampling from the Bot scanning dataset is confirmed from the analysis performed by the code review. The solution
created in this study, the Mirai Bot Scanner Summation Prototype, evaluates a Bot scanning
dataset. Researchers can use the prototype to tabulate the number of Mirai Bots, the number
of potential new Bot Victims, as well as the number of network packet types associated with a
Bot attempting to connect to a potential new Bot Victim. Using a database, permanent
storage is utilized for counting Bots, potential new Bot Victims, and network packet types.
Reporting as well as line-graphs is provided for assessing the Bot scanning mechanism over a
time period. Single case experimentation performed with the Mirai Bot Scanner Summation
Prototype provides answers to the research questions (a) Bots are identified for summation;
(b) Potential new Bot Victims are identified for summation; (c) the Bot scanner is monitored
over time. A comparison to a NIDS solution highlights the advantages of the prototype for
summating and assessing the Bot scanning dataset. Experimentation with the Mirai Bot
Scanner Summation Prototype and NIDS verifies it is possible to develop a solution that can
analyze network traffic to identify a Bot scanning for a potential new Bot Victim. Future
research could include adding the additional functionality to the Bot Scanner Summation
Prototype for evaluating a Bot scanner dataset for non-potential Bot Victims.
DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Charles V. Frank Jr.
# TABLE OF CONTENTS

MIRAI BOT SCANNER SUMMATION PROTOTYPE.............................................................. I

DISSERTATION APPROVAL FORM.............................................................................II

ACKNOWLEDGMENT ................................................................................................. III

ABSTRACT .................................................................................................................. IV

DECLARATION .......................................................................................................... VI

TABLE OF CONTENTS ............................................................................................. VII

LIST OF TABLES .................................................................................................... XII

LIST OF FIGURES ................................................................................................ XIII

INTRODUCTION ...................................................................................................... 1

  BACKGROUND OF THE PROBLEM ................................................................. 3
  STATEMENT OF THE PROBLEM ................................................................. 5
  STATEMENT OF THE PURPOSE ................................................................. 6
  THESIS AND RESEARCH QUESTIONS .................................................... 7
    Bots identified. ................................................................................................. 8
    Potential new Bot Victims identified. ............................................................. 9
    Bot scanning mechanism monitored ............................................................ 9

RESEARCH METHODOLOGY .............................................................................. 10

REPOSITORY OF CODE ...................................................................................... 12

  Prototype folder ............................................................................................... 13
  Splunk folder .................................................................................................... 14
  Suricata folder .................................................................................................. 14

RATIONALE AND SIGNIFICANCE .................................................................... 15

ASSUMPTIONS AND LIMITATIONS ............................................................. 17

  Assumptions .................................................................................................... 17
  Limitations ....................................................................................................... 17

KEY TERMS ........................................................................................................ 17

NATURE OF THE STUDY .................................................................................. 19

SUMMARY ........................................................................................................ 20

ORGANIZATION OF THE DISSERTATION ................................................ 21

LITERATURE REVIEW ..................................................................................... 22
DDoS Attack Survey ...................................................................................................................... 22

Attack timeline .............................................................................................................................. 23

Attack classification ....................................................................................................................... 24

 Flooding attacks ............................................................................................................................ 25

Software attacks ............................................................................................................................ 26

Mirai attack types ............................................................................................................................ 26

Defense strategies .......................................................................................................................... 27

Recent enterprise impact ................................................................................................................ 27

DDoS attack summary ................................................................................................................... 28

Mirai and Variants .......................................................................................................................... 29

Initial Mirai attacks ......................................................................................................................... 29

Mirai source code publicly released .............................................................................................. 29

Mirai spreading mechanism .......................................................................................................... 30

Mirai for rent ................................................................................................................................ 30

Prominent Dyn attack .................................................................................................................... 31

Bitcoin miner variant ....................................................................................................................... 32

Satori variants ................................................................................................................................. 32

Mirai and variants summary .......................................................................................................... 33

Mirai Components ......................................................................................................................... 33

Operation and communication steps ............................................................................................. 34

Bot malware ................................................................................................................................ 37

 Malware attack module .................................................................................................................. 37

 Malware killer module .................................................................................................................... 37

 Malware scanner module ............................................................................................................... 37

Bot scanner statistics ...................................................................................................................... 38

Mirai components summary ........................................................................................................... 38

Mirai Mitigation ............................................................................................................................. 39

Weak passwords ............................................................................................................................... 39

Cataloguing vulnerable IoT devices .............................................................................................. 40

Inoculation ................................................................................................................................... 40

Protecting IoT devices from infection ............................................................................................ 40

Network traffic analysis ................................................................................................................. 41

Mirai mitigation summary .............................................................................................................. 42

Network Traffic Analysis Solutions .............................................................................................. 43

Packet sniffers ................................................................................................................................. 43

Network intrusion detection systems ............................................................................................. 44

Network traffic analysis solutions summary ................................................................................... 45

Chapter Summary .......................................................................................................................... 46
# METHODOLOGY

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESEARCH METHOD JUSTIFICATION</td>
<td>47</td>
</tr>
<tr>
<td>RATIONALE FOR RESEARCH APPROACH</td>
<td>48</td>
</tr>
<tr>
<td>Problem identification</td>
<td>49</td>
</tr>
<tr>
<td>Requirements specification</td>
<td>49</td>
</tr>
<tr>
<td>Design work</td>
<td>53</td>
</tr>
<tr>
<td>Summation</td>
<td>54</td>
</tr>
<tr>
<td>Persistent database</td>
<td>57</td>
</tr>
<tr>
<td>Assessment</td>
<td>58</td>
</tr>
<tr>
<td>Artifacts</td>
<td>62</td>
</tr>
<tr>
<td>Evaluation</td>
<td>63</td>
</tr>
<tr>
<td>Communication</td>
<td>63</td>
</tr>
<tr>
<td>Rationale for research approach summary</td>
<td>63</td>
</tr>
<tr>
<td>RESEARCH DATA SOURCES</td>
<td>63</td>
</tr>
<tr>
<td>Required research data sources</td>
<td>64</td>
</tr>
<tr>
<td>Mirai program code</td>
<td>64</td>
</tr>
<tr>
<td>Bot scanning network dataset</td>
<td>65</td>
</tr>
<tr>
<td>Research data sources summary</td>
<td>65</td>
</tr>
<tr>
<td>DATA ANALYSIS METHODS</td>
<td>66</td>
</tr>
<tr>
<td>Code review</td>
<td>66</td>
</tr>
<tr>
<td>Bot scanning network dataset</td>
<td>68</td>
</tr>
<tr>
<td>Sample network packet analysis</td>
<td>69</td>
</tr>
<tr>
<td>Performing experimentation</td>
<td>70</td>
</tr>
<tr>
<td>Data analysis methods summary</td>
<td>71</td>
</tr>
<tr>
<td>LIMITATIONS AND DELIMITATIONS</td>
<td>71</td>
</tr>
<tr>
<td>Delimitations</td>
<td>71</td>
</tr>
<tr>
<td>Limitations</td>
<td>72</td>
</tr>
<tr>
<td>CHAPTER SUMMARY</td>
<td>72</td>
</tr>
<tr>
<td>EXPERIMENTATION</td>
<td>73</td>
</tr>
<tr>
<td>BOT SCANNER CODE REVIEW</td>
<td>74</td>
</tr>
<tr>
<td>Scanner initialization</td>
<td>74</td>
</tr>
<tr>
<td>Random IP address</td>
<td>81</td>
</tr>
<tr>
<td>Bot scanner code review summary</td>
<td>82</td>
</tr>
<tr>
<td>BOT SCANNING SAMPLE NETWORK PACKET ANALYSIS</td>
<td>82</td>
</tr>
<tr>
<td>Sample PCAP file</td>
<td>83</td>
</tr>
<tr>
<td>SYN packet</td>
<td>84</td>
</tr>
<tr>
<td>Retransmission packet</td>
<td>85</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

Table 1 *Mirai Bot Scanner Summation Prototype Functional Requirements* .......... 49  
Table 2 *Bot Scanner Summation Prototype Nonfunctional Properties* .................. 51  
Table 3 *Additional Tool Requirements* ............................................................... 53  
Table 4 *Pandas DataFrame Summation Results* .................................................. 57  
Table 5 *Summation Database Table Fields* ......................................................... 58  
Table 6 *Mirai Bot Scanner Summation Prototype Python Scripts* ....................... 62  
Table 7 *Detail and Summary Functions* ............................................................... 93  
Table 8 *Graphical Functions* ........................................................................... 95  
Table 9 *Method for NIDS Evaluation* ................................................................. 110  
Table 10 *Splunk Bot Total Report Sample* ......................................................... 114  
Table 11 *Splunk Network Packet Totals Report Sample* ..................................... 117  
Table 12 *Splunk Subnet Report Sample* ............................................................. 119  
Table 13 *Methods Comparison* ....................................................................... 120  
Table 14 *Performance Comparison* .................................................................. 121  
Table 15 *Solution Totals Comparison* ............................................................... 123  
Table 16 *Tshark Packet Comparison* ................................................................. 124
**LIST OF FIGURES**

*Figure 1.* Dyn attack heat map ................................................................. 31  
*Figure 2.* Mirai operation and communication .............................................. 35  
*Figure 3.* Network patterns between the loader and the new Bot Victim ........ 42  
*Figure 4.* Mira Bot Scanner Summation Prototype architectural components .... 54  
*Figure 5.* Summation algorithm ................................................................. 56  
*Figure 6.* Assessment reporting algorithm .................................................. 59  
*Figure 7.* Assessment graphing algorithm .................................................... 61  
*Figure 8.* Sublime text editor ..................................................................... 67  
*Figure 9.* Tail end of Bot scanner code ....................................................... 67  
*Figure 10.* Example of a daily packet capture ............................................. 68  
*Figure 11.* Analyzing a PCAP file with Wireshark ........................................ 69  
*Figure 12.* Analysis of retransmission packet with Wireshark ....................... 70  
*Figure 13.* Assign random IP address to IP header ...................................... 75  
*Figure 14.* Assign TCP header destination .................................................. 76  
*Figure 15.* Read packets from raw socket and verify IP header ..................... 77  
*Figure 16.* Verify TCP header ................................................................. 78  
*Figure 17.* Connection table parameters .................................................... 79  
*Figure 18.* Setup connection module ......................................................... 80  
*Figure 19.* Get random IP address ............................................................. 81  
*Figure 20.* 2016-11-20.pcap file analysis .................................................... 84  
*Figure 21.* SYN packet ........................................................................... 85  
*Figure 22.* TCP retransmission packet ....................................................... 86  
*Figure 23.* Gathering the packet source and destination IPs .......................... 88  
*Figure 24.* Determining the Bots and potential new Bot Victims .................. 89  
*Figure 25.* Totals inserted into MongoDB ................................................... 91  
*Figure 26.* PCAP processing time ............................................................... 91
CHAPTER 1

INTRODUCTION

In 1999, the term Internet of Things (IoT) was proposed by Kevin Ashton to describe how the physical world is connected to the Internet thru sensors (Gupta, Mudgal, & Mehta, 2016). The definition of IoT has garnered a great deal of attention, with numerous standards bodies and researchers posting different definitions (James A. Jerkins & Stupiansky, 2018). (Minerva, Bir, & Rotondl, 2015) study definitions for IoT. Minerva et al. (2015) state, “Different definitions and architectural models for IoT reflect different perspectives and support different business interests.” (p. 7). A common thread among all definitions is that IoT devices have the ability to collect and process information and communicate over the Internet (James A. Jerkins & Stupiansky, 2018). The Institute of Electrical and Electronics Engineers (IEEE), the world’s largest technical professional organization dedicated to advancing technology for the benefit of humanity, defines IoT as,

a self-configuring, adaptive, complex network that interconnects ‘things’ to the Internet through the use of standard communication protocols. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration. (Minerva et al., p. 74).

Current research is focused on securing IoT devices (Frank, Nance, Jarocki, & Pauli, 2017). Frank et al. (2017) estimate there are currently 15 billion Internet of Things (IoT) devices. By 2020, the estimate is projected to be as high as 50 billion IoT devices. Recent research conducted by the authors (Aman, Chua, & Sikdar, 2017) study trusting the data originating from IoT. With the rapid growing number of IoT devices deployed for smart cities, smart homes, smart hospital care, smart vehicles, and etc., the amount of data and the sensitivity of the data collected make the IoT devices prime targets for cyber-attacks.
Emphasized by IoT security researchers Aman et al. (2017), main security challenges faced by IoT include authentication, data integrity, data provenance, privacy, and access control. Authentication refers to the validation of the IoT device. Data integrity refers to the accuracy and consistency of the data collected via the IoT device. Data provenance is concerned with the historical record of the data and its origin. In context of this study, privacy refers to protecting the exposure of sensitive information collected from the IoT device. Access control refers to the authorizations assigned to the users of the IoT device. Emerging technologies such as blockchain and Software Defined Networks (SDNs), are possible security solutions for IoT (Mohan et al., 2018).

IoT systems, such as smart vehicles, must be concerned with securing the IoT devices that comprise the system and must be concerned with protection from unauthorized access. If the IoT devices allow for unauthorized access, then the IoT devices are vulnerable to cyber-attacks and the IoT system as a whole could become compromised. Recent research focuses on the security vulnerabilities of a smart car (Park et al., 2016). Sniffing the communication between the smart car and the cloud, the authors Park et al. (2016) analyzed that dash cam communication is not encrypted. Communication between the dash cam and the cloud revealed a packet is being sent with the factory default “admin” user-id and “000000” password. Concerning factory default credential settings, J. A. Jerkins et al. (2018) explain,

One of the most frequently exploited design flaws in IoT devices is embedding default credentials into the device’s firmware so that the owner cannot restrict remote access by changing or removing the credentials. (“IoT insecurity”, para. 1).

On October 2016 the Mirai botnet executed a Distributed Denial of Service (DDoS) attack against service provider Dyn that took down hundreds of websites (Kolias, Kambourakis, Stavrou, & Voas, 2017). An IoT botnet is a collection of compromised IoT devices, infected with malware, typically for executing DDoS attacks (Sinanović & Mrdovic, 2017). Mirai deploys a distributed spreading mechanism where Bots spread to webcams, DVRs, and routers with factory default user-ids and passwords (Kolias et al., 2017). Recent research has suggested network signatures for detecting the spreading mechanism (J. A.
Jerkins, 2017; Kolias et al., 2017; Sinanović & Mrdovic, 2017). The Mirai spreading mechanism consists of a Bot finding a new Bot Victim and then taking Command and Control (C&C) of the new Bot Victim thru the upload of the malware (Kolias et al., 2017). This study applies the knowledge base of previous Mirai research to provide an analysis of the network traffic associated with a Bot scanning for a potential new Bot Victim.

Chapter 1 of this study consists of the following sections (a) a discussion for the background of the problem; (b) a discussion on the statement of the problem for this study; (c) a discussion on the statement of the purpose of this study; (d) a discussion for the thesis statement and research questions; (e) a discussion for the research methodology of this study; (f) a discussion for the rational and significance of this research; (g) assumptions and limitations are presented; (h) key terms are defined; (i) a discussion for the nature of the study; (j) a discussion for the organization of the dissertation. The following section provides a discussion of the background for the Mirai botnet.

Background of the Problem

On October 21, 2016, 100,000 Mirai Bots attacked service provider Dyn and caused the un-availability of several high-profile websites such as GitHub, Twitter, Netflix and many others (Sinanović & Mrdovic, 2017). A month before the Dyn cyber-attack the Mirai code was exposed publicly on the hacking community web forum Hackforums (J. A. Jerkins, 2017). Within only two months of the source code’s release, the number of Bots more than doubled to half a million (Kambourakis, Kolias, & Stavrou, 2017). Hackers offer Mirai for rent with as many as 400,000 bots (Kolias et al., 2017). Recent research has identified the vulnerability in IoT devices that allows for the spreading of Bots (J. A. Jerkins, 2017; Kambourakis et al., 2017; Kolias et al., 2017).

Research performed by J. A. Jerkins (2017) focuses on how the Bot brute forces a new Bot Victim. The source code is reviewed to create a Mirai variant which will use the same Bot brute forcing mechanism to catalog IoT devices that are vulnerable to Mirai Command and Control (C&C). Reviewing the Mirai code, J. A. Jerkins (2017) emphasizes,

The Mirai botnet’s success was primarily due to the large number of available IoT devices with remote access credentials stored in their firmware. These devices exposed
Telnet, SSH, and web services to the Internet on their outside interfaces protected only by the firmware credentials. The credentials, in the form of username and password, cannot be changed without new firmware from the manufacture. Furthermore, the remote access services cannot be disabled without modifying the firmware of the IoT device or altering the running code on the device. (“The rise of IoT Botnets”, para. 2).

A code review has been performed concentrating on the Bot scanning mechanism (Conley, 2017). Besides generating a random non-government IP address, the Bot scanner code contains functionality for creating a network socket and performing a TCP handshake. The socket is a network interface that allows the Bot to communicate with TCP/IP over the internet (Koutsoubelias & Lalis, 2013). The TCP handshake exchanges SYN and ACK packets to establish connection between hosts (Arlitt & Williamson, 2005). The Bot attempts to establish a connection to a new Bot Victim on telnet ports 23 and 2323. Also, protocols, such as telnet, establish connection with the TCP handshake. The code review performed by Conley (2017) supports the research performed by (Šemič & Mrdovic, 2017). Semic and Mrdovic (2017) discovered that after the Bot establishes a connection, a telnet handshake occurs.

The research conducted by Kambourakis et al. (2017) reviews the code for details concerning the Bot and the C&C infrastructure. The Bot is responsible for identifying a new Bot Victim (Kambourakis et al., 2017). Kambourakis et al. (2017) explain, “The malware employs a brute-force dictionary-based technique for “guessing” passwords based on a hard-coded list. That inventory contains 62 username/password dyads.” (p. 269). Emphasizing network signatures can be created, Kambourakis et al. (2017) suggest that network traffic can be monitored on standard ports 23, 2323, and 22, which are bombarded with authorization attempts to the IoT device.

Current research performed by the authors Kolias et al. (2017) study Mirai communication and operation. The Mirai communication and operation consists of (a) the Bot brute forcing a new Bot Victim (b) the bot communicating the logon information back to the C&C server (c) the C&C server uploading the malware onto the new Bot Victim and (c) the C&C server commanding the Bots to execute a DDoS attack. Kolias et al. (2017) explain the brute forcing as containing two major responsibilities. For the first task, the Bot is continually
generating a random IP address and attempting to connect to a potential new Bot Victim with that random generated IP address. For the second major task, when a connection is successful the Bot will attempt a telnet remote access logon with a list of factory default user-ids and passwords. A new Bot Victim is identified when the Bot can logon with the factory default user-id and password. Analyzing the code and network traffic, Kolias et al. (2017) suggests, “Almost all stages of infection leave a footprint that can be recognized through basic network analysis.” (p. 82).

The authors (Sinanović & Mrdovic, 2017) perform static and dynamic analysis to classify Mirai DDoS attacks. Network signatures are experimented with for detecting Mirai DDoS attacks. A signature is suggested for detecting a Bot from an external network brute forcing a new Bot Victim. Based upon the research performed to analysis Mirai DDoS attacks, Sinanovic and Mrdovic (2017) conclude that Intrusion Detection Signatures (IDS) can be created for all parts of Mirai operation. Researchers Sinanovic and Mrdovic (2017) suggest creating a virtual test environment to perform experimentation to study how a Mirai Bot finds and compromises a vulnerable IoT device.

Recent research has reviewed the Mirai code and analyzed the Mirai network traffic. Mirai components are static and its behavior can be detected thru analyzing network traffic. Network signatures have been suggested for identifying a Bot brute forcing a new Bot Victim (J. A. Jerkins, 2017; Kambourakis et al., 2017; Kolias et al., 2017; Sinanović & Mrdovic, 2017). Even though network signatures have been suggested for detecting a Bot brute forcing a new Bot Victim, experimentation has not been focused on analyzing the network traffic for the Bot scanning mechanism. The next section states the research problem.

**Statement of the Problem**

The Mirai botnet deploys a distributed mechanism with each Bot continually scanning for a potential new Bot Victim (J. A. Jerkins, 2017; Kambourakis et al., 2017; Kolias et al., 2017). A Bot continually generates a random IP address to scan the network for discovering a potential new Bot Victim (Kambourakis et al., 2017). Once the Bot establishes a connection with the potential new Bot Victim, a telnet handshake occurs (Šemić & Mrdovic, 2017). A Bot remotely accesses the Bot Victim with telnet, providing a factory default user-id and password for logon (J. A. Jerkins, 2017). With 100,000 bots, Mirai DDoS attacks on service
provider Dyn in October 2016 triggered the inaccessibility to hundreds of websites in Europe and North America (Sinanović & Mrdovic, 2017). A month before the Dyn attack, the source code was released publicly on the Internet and Mirai spread to half a million Bots (Kambourakis et al., 2017). Hackers offered Mirai botnets for rent with 400,000 Bots (Kolias et al., 2017). Recent research has suggested network signatures for Mirai detection (Kambourakis et al., 2017; Kolias et al., 2017; Sinanović & Mrdovic, 2017). Based upon network analysis, Kolias et al. (2017) suggest network signatures can be created for Mirai detection. Kambourakis et al. (2017) study the Mirai malware code for Bot brute forcing and suggest signatures can be created for network detection. Performing network analysis of Mirai DDoS attacks, Sinanović and Mrdovic (2017) suggest future research experimenting with network signatures to detect a Bot brute forcing a new Bot Victim. Unfortunately, research provided in the literature has not performed experimentation focused on analyzing the network traffic for the Bot scanning mechanism, which is the focus of this research. The next section provides the purpose of this study.

**Statement of the Purpose**

The primary purpose of this research is to develop a prototype that summates the network traffic generated from the Bot scanning mechanism. The Bot brute forces a new Bot Victim thru remote access with telnet using a list of factory default user-ids and passwords. Before the Bot can remotely access the new Bot Victim, the Bot generates a random IP address and tries to connect to a potential Bot Victim associated with the random generated IP address. The solution, the Mirai Bot Scanner Summation Prototype, focuses on evaluating the Bot scanning mechanism, from network traffic, by counting the Bots and potential new Bot Victims and network packet types. The network traffic will contain the source IP of the Bot and the destination IP of the potential new Bot Victim.

An analysis of the network traffic generated during experimentation with the Mirai Bot Scanner Summation Prototype identifies the Bot scanning for a potential new Bot Victim. The number of Bots is identified. The amount of potential new Bot Victim is identified. The analysis supports the thesis. Furthermore, the analysis of the Bot scanning network traffic performed from experimentation with the Bot Scanner Summation Prototype answers additional questions that are not part of the primary focus.
Once the experimentation has completed and analysis of the Bot scanning mechanism has concluded with the Mirai Bot Scanner Summation Prototype solution, the results will be compared to existing peer-reviewed journals that focus on IoT. The next section will define the thesis and research questions.

**Thesis and Research Questions**

As stated in the Statement of the Problem subsection, a Bot is continually scanning for a potential new Bot Victim (Kambourakis et al., 2017). Static and dynamic analysis suggests Mirai components are recognizable. Network analysis has suggested signatures for Mirai detection (Kolias et al., 2017). The objective of the research is to perform experimentation to create a Mirai Bot Scanner Summation Prototype focused on analyzing the network traffic for summatting the Bot scanning mechanism.

In this dissertation, the following thesis is presented: *it is possible to develop a solution that can analyze network traffic to identify a Bot scanning for a potential new Bot Victim.* Analyzing the Bot scanning mechanism, a Bot can be identified for summation, a potential new Bot Victim can be discovered for summation, and the Bot scanning can be monitored over time.

For the literature review of this study, network traffic analysis solutions were studied as possible solutions for experimentation. The network traffic analysis solutions reviewed, packet sniffers and NIDS, do not meet the requirements for summatting the Bot scanning mechanism. Packet sniffers and NIDS lack a persistent database that is necessary for the summation process for tabulating Bots, potential new Bots Victims, and SYN and retransmission network packets. Also, packet sniffers and NIDS contain limitations for creating reports and graphs required for monitoring the Bot scanning mechanism over time. To design an original solution that meets the requirements for experimentation a code review of the Bot scanning mechanism is performed to determine the network interfaces and communication between a Bot and a potential new Bot Victim. The Mirai Bot Scanner Summation Prototype, the original solution of this study, searches thru the network packets from a Bot scanning dataset. Each packet is evaluated to identify the Bot as well as the potential new Bot Victim. The summation is tabulated based upon the Bot and potential new
Bot Victim identification. The Bot scanning is summated for the date range of the Bot scanning dataset.

Current research has analyzed the Mirai malware code for its functionality (Sinanović & Mrdovic, 2017). The Mirai malware residing on the Bot contains the functionality for brute forcing a new Bot Victim (Sinanović & Mrdovic, 2017). For this study, a code review performed on the Mirai malware for the Bot scanning mechanism supports the research conducted by (Conley, 2017; Šemić & Mrdovic, 2017). The code review has verified the Bot scanning mechanism. The Mirai Bot Scanner Summation Prototype performs experimentation to evaluate a network dataset consisting of the Bot scanner gathered by the University of Southern California (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). The experimentation performed with the Mirai Bot Scanner Summation Prototype will support the thesis and answer the following three research questions:

1. Can the Bots be identified for summation?
2. Can the potential new Bot Victims be identified for summation?
3. Is it possible to monitor the Bot scanning mechanism over time?

Answering the research questions will evaluate the experimentation performed with the Mirai Bot Scanner Summation Prototype. The research questions will help answer if it is possible to (a) identify a Bot for summation; (b) identify a potential new Bot Victim for summation; (c) determine the Bot scanning mechanism over time. Evaluation performed from the experimentation with the Mirai Bot Scanner Summation Prototype will be validated based upon answering the research questions.

**Bots identified.** Researchers Kolias et al. (2017) describe the operational and communication aspects of the Bot brute forcing a new Bot Victim. Before a Bot can remotely access a new Bot Victim with telnet, the Bot scans for a potential new Bot Victim. The scanning is comprised of generating a random IP and then connecting to that IP on ports 23 or 2323 (Conley, 2017; Kolias et al., 2017). A Bot connection to an potential new Bot Victim is established with the TCP handshake (Arlitt & Williamson, 2005). The Bot sends a SYN packet for establishing the TCP handshake, to the potential new Bot Victim on port 23 or 2323. The source IP of the SYN packet represents the Bot. Analysis performed in this study verifies the network packets associated with the Bot scanner dataset for containing SYN

**Potential new Bot Victims identified.** A Bot randomly generates an IP address with the hopes of that IP address becoming a potential new Bot Victim (Kolias et al., 2017). When the Bot tries to establish a connection to a potential new Bot Victim, a TCP handshake is initiated with a SYN network packet. As part of the TCP handshake initiation, the destination IP is expected to respond back with an ACK packet (Arlitt & Williamson, 2005). The destination IP may not receive the SYN packet due to some network security mechanism, such as a blocking firewall, or due to not listening on the SYN packet destination port. Also, a non-existent destination IP may not respond back with an ACK packet. If the Bot does not receive an ACK before the retransmission timer expires, the Bot will assume the ACK has been lost and will retransmit the SYN network packet. A potential new Bot Victim responds back to the Bot with an ACK. A potential new Bot Victim does not cause the Bot to retransmit ACK network packets. The Bot scanner dataset contains SYN and retransmission packets. The Bot Scanner Summation Prototype will linearly search thru the dataset to tabulate potential new Bot Victims from the SYN packets. Retransmission packets will not be included for the tabulation of potential new Bot Victims.

**Bot scanning mechanism monitored.** The dataset is collected from Bot scanning activity beginning 06/01/2016 and ending on 03/30/2017. The dataset contains a PCAP file for each day of Bot scanning network activity. The Bot Scanner Summation Prototype summates the number of Bots, potential new Bot Victims, and SYN packets per PCAP file. The daily summations are stored in a persistent database. The number of Bots, potential new Bot Victims, and SYN network packets can be tabulated within a time period by the Bot Scanner Summation Prototype. Based upon the date-range, the Bot Scanner Summation Prototype searches the persistent database. Searching the persistent database, the Bot Scanner Summation Prototype is able to summate the Bot scanning mechanism over that time-period. The number of Bots, potential new Bot Victims as well as the SYN packets are tabulated based upon the date range. The rate at which Bots attempt to connect to potential new Bot Victims can be calculated as well as the rate of potential new Bot Victims. The Bot Scanner
Summation Prototype provides reports and graphs for evaluating the Bot scanning mechanism. The next section discusses the research methodology.

**Research Methodology**

Design Science Research (DSR) focuses on the development of solutions for practical problems (Cleven, Gubler, & Huner, 2009; Kampling, Klesel, & Niehaves, 2016; Offermann et al., 2009). The main DSR requirements are rigor and relevance (Offermann et al., 2009). Artifact development commonly covers the phases (a) problem identification; (b) requirements specification; (c) design; (d) evaluation; (e) communication. (Cleven et al., 2009). Experimental research is well suited for DSR (Kampling et al., 2016). Thesis are evaluated using laboratory experiments (Offermann et al., 2009). Evaluation can be qualitative or quantitative (Cleven et al., 2009). Evaluation for this study is quantitative since the Bot scanner mechanism is summated. This study follows the DSR methodology for artifact development.

DSR researchers Cleven et al. (2009) describe the focus for artifact development as being either technical, organizational, or strategic. Examples of technical DSR artifacts could include computer programs, algorithms, and databases. Organizational artifacts could include process models and methods for organizational re-design. An example of a strategic artifact is a design for a decision support system or a road map for software development. This study developed technical artifacts for developing the Bot Scanner Summation Prototype to summate a Mirai Bot scanning dataset.

The first phase for artifact development is problem identification. Statement of the research problem is one of the most important parts of research (Ellis & Levy, 2008). The authors Ellis and Levy (2008) focus on the research problem and emphasize the importance and impact of the research problem for problem-based research. The research problem addresses and delimits the research questions (Ellis & Levy, 2008). The research questions determine the methodology and then the methodology produces results (Ellis & Levy, 2008). Cleven et al. (2009) explain, “Regarding an identified research problem, decisions concerning the artifact type and focus, object, and reference point may further be made” (“Configuring evaluation design criteria in the DSR process”, para. 3). The Statement of Problem subsection for this study has identified the problem and focus. Once the problem has been
identified, the next step in the DSR process for artifact development is requirements specification.

The requirements specification phase focuses on the requirements for the Mirai Bot Scanner Summation Prototype (Offermann et al., 2009). A functional requirement defines the behavior of an artifact. A nonfunctional property is a quality of the system, not behavior (Wieringa, 2014). An example of a necessary functional requirement for the Mirai Bot Scanner Summation Prototype is to evaluate the Mirai Bot scanner network traffic contained in the Bot scanner dataset for summating Bots and potential new Bot Victims. An example of a nonfunctional property is that the Bot Scanner Summation Prototype should be usable on a PC running the Windows 10 Operating System (OS) with at least 12 Gigabytes (GB) of memory, 500GB hard drive, and an Intel Core i5 2.20 Gigahertz (GHz) processor. A detailed list of the functional requirements and nonfunctional properties will be provided in Chapter 3 Methodology. Once the specifications and nonfunctional properties have been defined, the design work phase begins.

In the design work phase, the solution is designed (Offermann et al., 2009). Stated by Offermann et al. (2009), “It is divided into the steps “artefact design” and supporting “literature research.” (“Solution design”, para. 1). The Mirai Bot Scanner Summation Prototype consists of technical artifacts. The Mirai Bot Scanner Summation Prototype contains Python program scripts that search thru the Bot scanner dataset to summate Bots, potential new Bot Victims, and network packet including TCP SYN and retransmission packets. The Bot and packet summations for each day are permanently stored in a database. Each research question will have an associated defined Python function within the Mirai Bot Scanner Summation Prototype. The defined function contains the logic to answer the research question. There are three defined functions to answer the research questions for (a) calculating the number of Bots; (b) calculating the number of potential new Bot Victims; (c) monitoring the Mirai spreading mechanism over time. Chapter 2 of this study contains a literature review of the Mirai botnet as well as a review of competing solutions for analyzing network traffic datasets. Once the design has completed, the next step in DSR is evaluation.

Current research performed by Venable et al. (2012) presents a framework for an evaluation strategy. The evaluation for the Bot Scanner Summation Prototype is artificial since all of the artifacts are purely technical. Also, the evaluation strategy for this study is
expost since the Bot Scanner Summation Prototype evaluates a recorded Bot scanner dataset by summating the Bot scanning network activity. Evaluation is an iterative process and may require review and re-evaluation (Offermann et al., 2009). During the experimentation, supplementary code reviews may be needed to gather additional artifacts concerning the Bot scanning mechanism. The logic for the defined functions within the Bot Scanner Summation Prototype solution may need to be modified based upon experimentation or additional requirements gathering. Concerning the relationship between experiments and DSR, Kampling et al. (2016) emphasizes that experimentation is the central method for evaluation of artifacts in DSR, but few studies focus on the use of experiments.

Experimentation will be performed from a dataset gathered by the University of Southern California (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). The dataset contains only Bot scanning activity collection starting on June 1, 2016 and ending on March 31, 2017. The dataset contains Mirai identified TCP SYNs sent to ports 23 and 2323. Through 2016-11-20, only traffic to IP address (130.152.184.2) is included in the dataset. From 2016-11-20 onward traffic to 192.228.79.0/24 is added.

The dataset contains the Bot trying to establish a connection to a potential new Bot Victim over ports 23 and 2323, which are telnet ports (Kolias et al., 2017). The user executes functions within the Bot Scanner Summation Prototype to answer the research questions. The research questions evaluate the effectiveness of the Mirai Bot Scanner Summation Prototype for evaluating Bot scanner network traffic.

Once the experimentation is complete, the final stage is communication. The findings will be communicated within this dissertation. A research paper will explain the findings of this research and the findings will be presented at a conference and a research paper will be published in a peer-reviewed journal. The following section details repository of the code and configuration files of this study.

**Repositories of Code**

Github is a popular software development platform for version control with Git (Github, 2019). Github allows developers to maintain a repository of program code as well as non-program files (Ray, Posnett, Filkov, & Devanbu, 2014). For this dissertation, a Github
repository was maintained for the Mirai Bot Scanner Summation Prototype that allows for
public access and download of the Python programs as well as the configuration files and
Excel spreadsheets utilized during experimentation (Infosecchazzy, 2019). In Chapter 2,
*Network Traffic Analysis Solutions*, Network Intrusion Detection System (NIDS) is reviewed,
discussing the Suricata NIDS for alerting on network traffic and the Splunk Security Incident
Event Management (SIEM) for reporting on the Suricata alert log. In Chapter 4, *Mirai Bot
Scanner Summation Prototype Code Review*, the prototype Python programs for summation
and assessment are reviewed, followed by a *Single Case Experiment* with the prototype and
the Bot scanning dataset. For a *Prototype Solution Comparison*, Suricata and Splunk are
configured for Bot summation and assessment, with the results being compared to the Mirai
Bot Scanner Summation Prototype. The repository contains programs, configuration files,
and spreadsheets for the prototype, Suricata, and Splunk (Infosecchazzy, 2019).

The Mirai Bot Scanner Summation Prototype repository consists of (a) a Word
document, *Ann_Senpai_Statement.docx*, which contains the Anna Senpai statement, posted
on an internet forum, describing the Mirai botnet (Gamblin, 2016) ; (b) an Excel spreadsheet,
*Bot_Scanning_Solution_Comparison.xlsx*, which contains a comparison between the
prototype and Splunk for Bot and network packet summation totals during experimentation;
(c) a Prototype folder containing Mirai Bot Scanner Summation Prototype Python programs;
(d) a Splunk folder consisting of reports generated from Splunk searches; (e) a Suricata folder
containing configuration files for alerting on when a Bot attempts to connect to a potential
new Bot Victim.

**Prototype folder.** The prototype folder contains a repository of the Python programs
required for the summation and assessment components of the Mirai Bot Scanner Summation
Prototype. The folder consists of the following files:

- *ANSWER_RESEARCHQUESTIONS.PY FORMATTED TEXT OUTPUT.docx*,
  which contains the assessment for all of the PCAP files of the dataset;
- *Analyze_PCAP_Files.py*, a Python module that enumerates the PCAP files for
  summation;
- *Answer_Research_Questions.py*, a module that calls the functions to answer the
  three research questions;
- BotScanner.py, which contains the functions to perform summation with Mirai Bot Scanning dataset;
- BotScannerResults.py, which contains the functions for assessment of the Mirai Bot Scanning dataset;
- Packet_Summary_Questions.py, which provides a summary of the runtime for summatng the dataset.

The prototype folder contains the Python programs that encompass the summation and assessment components of the Mirai Bot Scanner Summation Prototype. During experimentation, the Python programs completed the summation and assessment of the Bot scanning dataset. The next sub-section describes the contents of the Splunk folder.

Splunk folder. The Splunk folder contains reports, generated during the experimentation of this dissertation, from Splunk searches of the Suricata alert log, with the log containing alerts of Bots attempting to connect to potential new Bot Victims. The folder contains (a) Packet_Counts_Suricata_Splunk.csv, which contains network packet counts per PCAP date; (b) Subnets_Suricata_Splunk.csv, which contains potential new Bot Victim subnets per PCAP date; (c) Total_Bots_Suricata_Splunk.csv, which contains Bot and potential new Bot Victim counts per PCAP date. The Splunk folder contains the Splunk search results that allow for summation comparison with the prototype summation results. The next subsection presents the Suricata folder.

Suricata folder. The Suricata folder contains the programs and configuration files necessary to produce an alert log, from the Bot scanning dataset, of Bots attempting to connect to potential new Bot Victims. The folder contains (a) Alert_Dataset.bat, which is a batch file for calling Suricata to alert on each PCAP file of the Bot scanning dataset; (b) Convert_Pcap.py, which converts the PNCAP files of the Bot scanning dataset to PCAP format; (c) mirai-bot-scanning.rules, which contains the Suricata rule for detecting when a Bot is attempting to connect to a potential new Bot Victim; (d) suricata.yml, which contains the configuration for the Suricata alert log. The Suricata folder contains the programs and configuration files that allow Suricata to produce an alert log that is searched by Splunk for summatng Bots, potential new Bot Victims, and network packets. The next section details the rationale and significance for this research.
Rationale and Significance

The rationale for this research comes from the fact that according to Kambourakis et al. (2017), “after its source code was disclosed on Sept. 30, 2016, Mirai botnets managed to control remotely nearly half a million IoT devices, assembling a mighty botnet.” (p. 267). A month after the source code was disclosed, the prominent Dyn attack occurred with 100,000 Bots executing a DDoS attack against DNS service provider Dyn which caused the unavailability of hundreds of web-sites. (Sinanović & Mrdovic, 2017). According to Kambourakis et al. (2017), since the release of the Mirai source code Mirai variants are created daily. The authors Kolias et al. (2017) state,

Today, Mirai mutations are generated daily, and the fact that they can continue to proliferate and inflict real damage using the same intrusion methods as the original malware is indicative of IoT device vendors’ chronic neglect in applying even basic security practices. (p. 81).

At the end of Feb. 2017, a Mirai variant executed a 54-hour long app-layer DDoS attack against a US College (Kambourakis et al., 2017). According to Kambourakis et al. (2017) nearly 900 thousand customers of Deutsche Telekom Internet Service Provider (ISP) were denied Internet access after their routers being enslaved as Bots from a Mirai variant. Mirai is successful in spreading to IoT devices due to remote access credentials stored in the IoT firmware (J. A. Jerkins, 2017). The Bot scans to connect to a potential new Bot Victim (Kolias et al., 2017). The Bot identifies a new Bot Victim by remotely accessing the new Bot Victim with a factory default user-id and password (Kolias et al., 2017). Referring to insecure IoT interfaces, at least six out of ten IoT devices tested contained security issues with their web access interfaces. These security issues were centered around poorly managed sessions and weak logon credentials (Atwady & Hammoudeh, 2017).

Recent research focuses on IoT security challenges (Medwed, 2016). The author Medwed (2016) emphasizes, “Exploits reported at a steady pace clearly suggest that security is a major challenge when the world wants to successfully switch from an IoT hype to a real IoT deployment.” (“ABSTRACT”, para. 2), Current research is focused on mitigating IoT device vulnerabilities (Hadar, Siboni, & Elovici, 2017). Researchers Hadar et al. (2017)
explain that anti-virus for IoT devices is not practical due to limited computing power on the actual IoT device. Recent research has shown that Mirai does not attempt to avoid detection (Kolias et al., 2017).

Recent research has focused on studying network traffic during C&C of the upload of the malware onto the new Bot Victim (Kolias et al., 2017). Additional research has been focused on classifying the DDoS attacks commanded by the C&C server (Sinanović & Mrdović, 2017). Research has suggested network signatures for detecting a Bot brute-forcing a new Bot Victim (Kambourakis et al., 2017; Kolias et al., 2017; Sinanović & Mrdović, 2017). Although network signatures have been suggested for brute forcing, experimentation has not been performed to study the brute-forcing. The limited amount of peer-reviewed research focused the Mirai botnet indicates the lack of research focused on IoT botnets (Kolias et al., 2017).

Current research on the Mirai botnet has not focused on the Bot scanner to study how a Bot connects to a potential new Bot Victim. This Bot Scanner Summation Prototype performs experimentation with the Bot scanner. Experimenting with a Bot scanner dataset, the Bot Scanner Summation Prototype reveals the number of Bots, number of potential New Bot Victims, and the number of Bots scanning over time. Answering the research questions with the experimentation provides deeper insight and knowledge into the number of Bots scanning, the number of potential new Bot Victims, as well as the network communication and protocols required for a Bot to connect to a potential new Bot Victim.

A comparison of network packet analysis tools demonstrates the significance and elegance of the Mirai Bot Scanner Summation Prototype solution. Packet sniffer stools and NIDS were compared. Packet sniffers and NIDS do not contain textual and graphical output capabilities needed for evaluation of the dataset in order to answer the research questions. Also, packet sniffers and NIDS do not contain a database for permanent storage of the summation of the Bot scanner per day. Packet sniffers and NIDS do not contain the necessary functionality for calculating the number of Bots, potential Bot Victims, and are not able to monitor the Bot scanner over time.

The Bot Scanner Summation Prototype provides insight into the Bot scanner which has not been previously provided by current Mirai research. Considering that variants deploy the same Bot scanning methods as the original Mirai botnet, future research could include
experimentation with variants. Several assumptions have been made to perform this research and the next section discusses those assumptions.

Assumptions and Limitations

This selected research topic filled a void for studying the Bot scanner. Although current research has studied network traffic, the focus of those studies has not been on a Bot scanning for a potential new Bot Victim. It is possible that experimentation exists for the Bot scanner, but there was a lack of references in peer-reviewed journals during the literature review.

Assumptions. The scope of this study is the original Mirai code that was posted on an internet forum. The Mirai botnet does not encrypt network traffic.

Limitations. The captured dataset is limited to the Bot scanner network traffic.

Botnet research and IoT have produced several terms that have specific meaning. To eliminate confusion in this research, the following section will contain a definition of terms.

Key Terms

The following terms will be continually used throughout this research. The definitions will help to eliminate any confusion with these terms.

Bot: An IoT device that is under Mirai C&C (Kolias et al., 2017).

Bot scanner: The Bot is constantly generating random IP addresses, attempting to connect to the random IP address, and trying to remotely access the potential new Bot Victim with telnet and a factory default user-id and password (J. A. Jerkins, 2017).

Bot scanner dataset: The dataset contains Bot scanning activity collection by the University of Southern California starting on June 1, 2016 and ending on March 20, 2017 (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016)

Bot Victim: An IoT device with default credentials that is vulnerable to Mirai malware infection (J. A. Jerkins, 2017).
**C&C**: Command and Control refers to the Mirai server responsible for loading the malware onto the new Bot Victim and instructing the Bots to execute a DDoS attack (Kolias et al., 2017).

**DDoS**: Distributed Denial of Service refers to an attack that floods an internet service with network traffic from different sources to render that internet service unavailable (Sinanović & Mrdovic, 2017).

**DSR**: Design Science Research is the research methodology followed by this study for experimentation. Design Science Research focuses on the development of solutions for practical problems (Cleven et al., 2009; Kampling et al., 2016; Offermann et al., 2009).

**IEEE**: The Institute of Electrical and Electronics Engineers, the world’s largest technical professional organization dedicated to advancing technology for the benefit of humanity, is conducting on-going research towards the definition of IoT (Minerva et al., 2015).

**IoT**: “a self-configuring, adaptive, complex network that interconnects ’things’ to the Internet through the use of standard communication protocols. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration.” (Minerva et al., p. 74).

**IP address**: A unique network identification for the IoT device (Kambourakis et al., 2017).

**Mirai**: An IoT botnet that spreads to IoT devices with factory default credentials (J. A. Jerkins, 2017).

**Mirai Bot Scanner Summation Prototype**: The solution presented in this study for performing experimentation with the Bot scanner dataset for assessing the summation of Bots, potential new Bot Victims, and network packets.

**Mirai variant**: The original Mirai code is modified to include different functionality (Kolias et al., 2017).

**NIDS**: A Network Intrusion Detection System applies signatures or use anomaly detection for alerting on malicious network traffic (Kumar & Singh, 2012).
**PCAP file:** A standard format for captured network traffic that can be read by network monitors or network packet sniffers, such as Wireshark (Buczak et al., 2016).

**SIEM:** A Security Incident Event Management solution assesses the alert log from a NIDS (Nagaraja & Kumar, 2018).

**TCP handshake:** A connection between a Bot and a potential new Bot Victim is established thru the exchange of SYN and ACK packets (Arlitt & Williamson, 2005).

The next section presents the nature of the study.

**Nature of the Study**

The design of this study was to perform experimentation to support the thesis. A solution, the Mirai Bot Scanner Summation Prototype, was evaluated to analyze network traffic to summate the Bot scanning mechanism. Recent peer-reviewed research has not been focused on perform experimentation for the Bot scanning mechanism (Kambourakis et al., 2017; Kolias et al., 2017; Sinanović & Mrdovic, 2017).

The DSR methodology was followed to develop the Mirai Bot Scanner Summation Prototype. A code review was performed for the Bot scanner, to identify the communication between a Bot and a potential new Bit Victim. The dataset was sampled, to characterize the Bot scanning network packets. The solution, the Mirai Bot Scanner Summation Prototype, was developed to perform quantitative analysis of the dataset. The solution calculates the number of Bots, number of potential new Bot Victims, and the number of network packet types including SYN and re-transmission packets. Based upon the calculations per day, the Bot spreading mechanism was monitored for the date range of the Bot scanning dataset.

The Mirai code was exposed publicly on the hacking community web forum Hackforums (J. A. Jerkins, 2017). The code review for this study reviews the original Mirai code posted on Hackforums for the Bot scanner. According to Kambourakis et al. (2017), since the release of the Mirai source code Mirai variants are created daily. Variant code is not reviewed.

The dataset contains Bot scanning activity collection starting on June 1, 2016 and ending on March 20, 2017 (*Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016*). This dataset contains Bot scanning
activity limited to the TCP SYN network packet sent to a potential new Bot Victim for connecting to telnet ports 23 and 2323. Recent research studying Mirai network traffic has performed experimentation from a controlled testing environment (Kambourakis et al., 2017; Koliás et al., 2017; Sinanović & Mrdović, 2017). Even with limitations, experimentation performed by the Bot Scanner Summation Prototype solution with this dataset can answer the research questions and confirm the thesis, from actual captured Bot scanning network activity. The next section provides a summary for this chapter.

**Summary**

Depending upon the definition of IoT, there are billions of IoT devices. By 2020, the estimate for IoT devices is as high as 20 billion. IoT devices provide the needed applications for gathering data for smart cities, smart cars, smart homes, health care, and so on. Applications for IoT seems limitless. IoT devices are vulnerable due to factory default weak passwords. The IoT device is manufactured with a default user-id and password. Research has shown that IoT devices are vulnerable to being compromised due to weak passwords associated with the factory default user-id.

On October 2016 the Mirai botnet executed a prominent DDoS attack against service provider Dyn, rendering hundreds of websites un-available. It is estimated that 100,000 Mirai Bots attacked Dyn. With the public release of the Mirai code, Mirai has been offered as-a-service and variants are created daily. A hacker, BestBuy, brags of offering Mirai as-a-service consisting of 400,000 bots. Mirai variants have not only included DDoS attacks but also perform SQL Injection attacks and bitcoin mining. Variants have expanded the Bot spreading mechanism to include exploits for firmware vulnerabilities. This study focuses on the original Mirai botnet spreading mechanism.

The Mirai Bot spreads the malware to IoT devices with factory default user-ids and passwords. The Mirai spreading mechanism consists of a Bot finding a new Bot Victim and then taking Command and Control (C&C) of the new Bot Victim thru the upload of the malware. A component of the Mirai spreading mechanism is the Bot connecting to a potential new Bot Victim with a TCP handshake over telnet ports 23 and 2323. When trying to establish the TCP handshake, the potential new Bot Victim returns a SYN packet waiting for an ACK packet from the Bot for acknowledgement. Once the Bot receives the SYN packet, a
potential Bot Victim has been identified, and then the Bot attempts to remotely access the potential Bot Victim via telnet with known factory default user-ids and passwords. If the Bot brute-force is successful, then C&C uploads the malware onto the Bot Victim.

Recent peer-reviewed research has focused on experimentation the Mirai DDoS attacks, the C&C upload of the malware, and has suggested research for detecting the Mirai spreading mechanism with Snort IDS signatures. This study follows the DSR methodology for developing and experimenting with a solution for detecting a Bot connecting to a potential new Bot Victim. The artificial solution, the Bot Scanner Summation Prototype, evaluates a Bot scanner dataset to answer the research questions. The Bot Scanner Summation Prototype provides the artifacts to calculate the number of Bots, potential new Bot Victims, and network packet types over a user-supplied date-range of the dataset. The Bot Scanner Summation Prototype provides textual as well as graphical analysis. Compared with existing solutions, such as network packet analyzers, network forensic tools, and NIDS, the Mirai Bot Scanner Summation Prototype contains the artifacts necessary to answer the research questions. The next section discusses the organization of the remainder of this study.

Organization of the Dissertation

Chapter 2 is a literature review related current research performed with the Mirai botnet as well as provide a review and comparison of existing solutions to the solution presented in this study, the Mirai Bot Scanner Summation Prototype. Chapter 3 details the DSR methodology for this study. Chapter 4 presents the findings and results from the experimentation performed with the Mirai Bot Scanner Summation Prototype. Chapter 5 completes this study by revealing conclusions, recommendations, and observations relative to suggested additional research defined by this study.
CHAPTER 2

LITERATURE REVIEW

In Chapter 1 a definition of IoT is provided from the IEEE (Minerva et al., 2015). A current study performed by researchers Minerva et al. (2015) presents many different definitions and architecture models for IoT. Currently, there is an estimated 15 billion IoT devices. By 2020, the estimate is as high as 50 billion (Frank et al., 2017). The rapid growth of IoT devices has paved the way for systems such as smart cities, smart hospital care, and smart vehicles etc. (Aman et al., 2017). IoT systems face security challenges that include authentication, data integrity, data provenance, privacy, and access control (Aman et al., 2017). One of the most frequently exploited vulnerabilities in IoT devices is factory default credentials (James A. Jerkins & Stupiansky, 2018). On October 2016 Mira executed a DDoS attack against service provider Dyn that took down hundreds of websites (Kolias et al., 2017). Recent research has suggested network signatures for detecting Mirai (J. A. Jerkins, 2017; Kolias et al., 2017; Sinanović & Mrdovic, 2017).

The focus of this study is to apply the knowledge base of previous peer-reviewed Mirai research to perform experimentation with a Bot scanning network dataset to identify a Bot scanning for a potential new Bot Victim. The literature review will consist of six sections (a) a survey of DDoS attacks is presented; (b) a discussion on a survey for Mirai and variants; (c) a discussion on the Mirai components; (d) a discussion on current research focused on Mirai mitigation; (e) a comparison of solutions for network traffic analysis; (f) a discussion for the summary of the literature review. The following section presents a survey for DDoS attacks.

DDoS Attack Survey

A DDoS attack attempts to render a target unavailable by overwhelming it with network traffic from multiple sources. The target could be an internet website, a server,
To create an effective DDoS attack, three operational and communication steps are needed: Scanning, Propagation and C&C (Srivastava, Gupta, Tyagi, Sharma, & Mishra, 2011). Initially, the attacker or even the bots themselves scan to find vulnerable hosts to join the botnet. Propagation is the process of spreading the malware onto vulnerable hosts to form bots. The attacker commands the bots to execute the DDoS attacks against a target.

This section presents a survey of DDoS attacks. It consists of the following subsections (a) a discussion of a timeline for DDoS attacks; (b) a discussion for the classification of DDoS attacks; (c) a discussion concerning DDoS defense mechanisms; (d) a discussion of the impact of recent DDoS attacks on enterprises; (e) a summary of DDoS attacks.

**Attack timeline.** Researchers Deshmukh et al. (2015) provide a timeline of DDoS attacks. DDoS attacks were first witnessed starting in 1999 and continue on to this very day. Some of the major DDoS attacks described by researchers Deshmukh et al. (2015) are as follows:

- In 1999, a Trinoo network was commanded to flood a single system at the University of Minnesota, rendering the University of Minnesota network unusable for more than two days.
- In 2000, a 15-year-old boy launched a DDoS attack on Yahoo CNN, eBay, Dell, and Amazon.
- In 2003, the Mydoom e-mail worm was used to shut down the SCO group website for two weeks (Wong, Bielski, McCune, & Wang, 2004).
- In December 2007 Russian government websites suffered DDoS attacks.
- In November 2008, the Conficker worm propagated thru vulnerabilities in the Microsoft OS (L. Zhang, Yu, Wu, & Watters, 2011).
- On July 4th, 2009, 27 US government websites experienced a DDoS attack. Among the US government site attacked were the White House, Federal Trade Commission (FTC), Department of Transportation, and the Department of the Treasury.
• On 2010 DDoS attacks were launched on credit-card websites for MasterCard, PayPal, and Visa. This string of attacks is known as "Pay Back" and was launched in retaliation for stopping to give credit services to WikiLeaks.

• In 2011 the LulzSec hacktivist group attacked the website of the US Central Intelligence Agency (CIA).

• 2012 witnessed many DDoS attacks on US banks involving use of the Itsoknoproblembro DDoS tool.

Recent DDoS attacks occurring in 2016 include IoT and an attack on a cryptocurrency start-up (Pritchard, 2018). Security researcher Pritchard (2018) listed the Mirai DDoS attacks that occurred in 2016. A review of Mirai DDoS attacks will be presented in the next section for Mirai and variants. Starting on April 5, 2017, over a four-day period the BrickerBot IoT botnet launched thousands of Permanent Denial of Service (PDoS) attempts from various locations around the world. On November 2, 2017 the Electroneum cryptocurrency start-up’s website was a victim of a DDoS attack. The DDoS attack locked caused Electroneum to lock investors out of their accounts while the DDoS attack was mitigated. The Financial Conduct Authority, the UK financial markets regulator, emphasized to the investors that Initial Coin Offerings (ICOs) offer no protection. On February 28, 2018, GitHub was hit by a Memcached amplification attack of 1.35 TBs, which is the largest-ever DDoS attack (Paganini, 2018). The following sub-section discusses the classification of DDoS attacks.

**Attack classification.** DDoS attacks can be classified into two categories: flooding and software (Srivastava et al., 2011). A flooding attack occurs when the target is overwhelmed with network traffic thus exhausting the bandwidth with a high number of packets. A software attack attempts to exploit a software vulnerability with a small number of malformed packets. The main types of DDoS flooding attacks are: SYN, Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), DNS, Hypertext Transfer Protocol (HTTP), and Amplification (Cloudflare, 2018; Deshmukh & Devadkar, 2015; Srivastava et al., 2011). The main types of software attacks can be classified as: Ping of Death, Teardrop Attack, and Land Attack (Srivastava et al., 2011). Focusing on the Mirai botnet, Bots are commanded for executing ten different types of DDoS flooding attacks (Sinanović & Mrdovic, 2017). The next sub-section presents a review of flooding attacks.
**Flooding attacks.** In a *SYN flooding attack*, the victim is flooded with SYN packets containing spoofed IP addresses. The connections on the targeted victim are left open waiting for a response from the spoofed IP. If the victim is flooded with enough SYN packets, the victim could hang or crash (Srivastava et al., 2011).

In an *ICMP flooding attack*, forged ICMP echo packets, containing the targeted victim IP, are sent to broadcast addresses of vulnerable networks. All of the systems on the vulnerable networks reply to the targeted victim with ICMP ECHO packets. A flood of ICMP ECHO packets exhausts the bandwidth available to the targeted victim (Nikolskaya, Ivanov, Golodov, Minbaleev, & Asyaev, 2017).

In a *UDP flooding attack*, a deluge of UDP packets are sent to a random port on the victim system. Realizing that no applications are associated with the port, the victim generates an ICMP packet with an unreachable destination. Flooded with enough UDP packets, the targeted victim could become unresponsive or crash (Srivastava et al., 2011). Other variations of UDP flooding attacks include *Fragmentation, DNS flooding, VoIP flooding*, and *Media data flooding* (Nikolskaya et al., 2017).

In a *DNS flood attack*, a domain’s DNS servers are flooded in an attempt to disrupt DNS resolution. DNS flood attacks are a new type of attack corresponding with the formation of IoT botnets. DNS flood attacks take advantage of the high bandwidth connections of IoT devices to directly overwhelm the DNS servers of major DNS service providers. Disrupting DNS resolution compromises a website's ability to respond to legitimate traffic (Cloudflare, 2018).

In a *HTTP flooding attack*, a victim website is overwhelmed with HTTP requests. There are two types of HTTP attacks: *GET* and *POST*. A *HTTP GET attack* requests images, files, or other assets from the website. In a *HTTP POST attack*, typically a form is submitted on a website. The server must handle the incoming request and query a database. A HTTP POST attack is more computing intensive than a HTTP GET attack. Once the victim has been deluged with the HTTP requests, the website is unable to respond to normal traffic (Cloudflare, 2018).

In an *Amplification flooding attack*, a large number of packets, forged with the victim IP, are sent to a broadcast IP address of a network. The systems in the broadcast address range send a reply, resulting in a flood of network traffic targeted at the victim. This type of
attack exploits the broadcast address feature found in network routers. *Smurf* and *Fraggle* are examples of amplification flooding attacks (Nikolskaya et al., 2017). The next-subsection reviews DDoS software attacks.

**Software attacks.** In a *Ping of Death* attack, the attacker uses of the ping command to send an IP packet larger than the 65,536 bytes allowed by the IP protocol. Not knowing how to handle the oversized packet, systems hang or crash. Nowadays systems are safe from this type of attack (Srivastava et al., 2011). For a *Teardrop Attack*, an attacker sends two packet fragments that cannot be reassembled properly. Making use of a bug in the TCP/IP fragmentation re-assembly of operating systems, the system could reboot or shutdown (Srivastava et al., 2011). In a *Land Attack* an attacker sends a malicious packet with the same source and destination IP address. Whenever the victim system replies to the malicious packet an infinite loop is created resulting in a slow-down of the system that can lead to a reboot. Other fields can be manipulated to form malicious IP packets in an attempt to cause a DoS attack (Deshmukh & Devadkar, 2015). The next sub-section discusses Mirai DDoS attack types.

**Mirai attack types.** The Mirai malware spreads to vulnerable IoT devices to form a botnet that is commanded and controlled for DDoS attacks (Sinanović & Mrdovic, 2017). The botmaster commands the DDoS attacks from the C&C server (Kambourakis et al., 2017). The C&C communicates with the Bot to specify the target and type of DoS attack. Once the Bot receives the attack information from the C&C server, the DDoS attack commences with the Bot attacking the victim. The botmaster is presented with a virtual terminal on the C&C server that contains ten types of DDoS attacks that can be commanded (Sinanović & Mrdovic, 2017).

*UDP* and *UDPPLAIN* attack commands issued from the C&C server instruct the Bots to generate UDP packets with a random payload and random source IP address. *SYN, ACK,* and *STOMP* attack commands generate SYN or ACK flood attacks. The *HTTP* attack command initiates a HTTP flooding attack. The *VSE* command generates a Valve Source Engine query UDP Amplification flood attack. The *DNS* command targets the authoritative DNS server with a DNS flooding attack. The *GREETH* and *GREIP* commands flood the victim with malicious Generic Routing Encapsulation (GRE) encapsulated Ethernet and IP packets. The next sub-section discusses DDoS defense strategies.
**Defense strategies.** DDoS defense strategies can be classified into the following categories: prevention, detection, and response (Deshmukh & Devadkar, 2015). Prevention methods try to stop DDoS attacks from being initiated. Ingress and egress packet filters on routers can help to prevent traffic from malformed packets (Srivastava et al., 2011). A firewall can block malicious network packets from entering or leaving a network. A Web Application Firewall (WAF) can prevent malicious HTTP requests from reaching the website (Sheth & Thakker, 2013). Other prevention techniques include disabling IP broadcasting, disabling unused services, and applying security patches (Deshmukh & Devadkar, 2015).

Detection aims to detect an ongoing attack as soon as possible without disrupting legitimate traffic. Signature-based Network Intrusion Detection Systems (NIDS) apply signatures to detect network traffic of known DDoS attacks. Anomaly-based NIDS establish a baseline and then detect anomalies, caused by the DDoS attack, in the network traffic (Srivastava et al., 2011). When DDoS attack is detected, a response could be to block the attack or trace the origin of the attack. Response can be accomplished using an Access Control List (ACL). The ACL on the router could prevent malicious packets from being routed to the targeted victim (Deshmukh & Devadkar, 2015). Updated firewall rules can block malicious network traffic (Sheth & Thakker, 2013). Tracing back the source of the attack is difficult because of spoofed IP addresses (Srivastava et al., 2011). The following sub-section presents the recent impact of DDoS.

**Recent enterprise impact.** Neustar is a leading DDoS protection company (Neustar, 2018). Neustar queried 1,010 executives to find out how DDoS attacks affect their organizations and what measures are in place to counter these attacks. According to the 2017 Neustar Worldwide DDoS Attacks & Cyber Insights Research Report, DDoS attacks are on the rise and the enterprise can now expect the cost of at least $2.5 million every time they become a victim (Neustar, 2017). Included in the Neustar report:

- 84 percent included in the research have experienced at least one DDoS attack in the last 12 months, up from 73 percent in 2016;
- 86 percent of these businesses were struck with multiple DDoS attacks over the past 12 months;
63 percent said the loss of revenue at peak times caused by DDoS disruption can sometimes reach beyond $100,000 an hour;

45 percent of DDoS attacks were greater than 10 Gbps per second;

15 percent of attacks were greater than 50 Gbps which is almost double the rate reported in 2016.

A recent Kaspersky Lab study indicates DDoS attacks are on the rise in the first quarter of 2018 (Alexander Khalimonenko, 2018). There is a significant increase in the total number and duration of DDoS attacks compared to the fourth quarter of 2017. SYN attacks remain the most popular form of DDoS attacks. There is a six-times increase in sustained attacks lasting longer than 50 hours. The Kaspersky study estimates DDoS attacks originating from China were at 59.42%, followed by the US at 17.83%, and South Korea at 8%.

The Neustar report shows the deep financial costs for DDoS attacks and shows that DDoS attacks are becoming increasingly costly and more powerful. The Kaspersky study for the first quarter of 2018 indicates an increase in the occurrence of DDoS attacks as well as an increase in the number of sustained attacks. The next section discusses a summary of DDoS attacks.

**DDoS attack summary.** DDoS attacks attempt to overwhelm the victim website with network traffic. The first DDoS attack occurred in 1999 to overwhelm the University of Minnesota network. DDoS attacks have progressed to the current internet landscape where IoT botnets are utilized for targeted DDoS attacks. DDoS attacks can be categorized into flooding and software attacks. Flooding attacks send a deluge of malicious packets to the targeted server in hopes of crashing it. Software attacks attempt to exploit software vulnerabilities. Mitigation strategies include trying to prevent, detect, and respond to DDoS attacks. Patching software vulnerabilities and ingress and egress packet filters on routers can help to prevent DDoS attacks. NIDS can detect DDoS attacks based upon signatures and anomalies. Recent studies indicate that DDoS attacks targeted at enterprises are rising as well as the costs. The next section discusses the Mirai botnet and variants.
Mirai and Variants

The previous section presented a discussion on a survey of DDoS attacks. DDoS attacks started as early as 1999 and continue to occur today. There are two main classifications for DDoS attacks: flooding and software. Flooding attacks attempt to deluge the targeted victim with network traffic. Software attacks exploit known vulnerabilities contained in program code. Mitigation techniques attempt to detect and prevent DDoS attacks. Because of spoofing IP addresses, it is difficult to traceback the attack. Recent studies have shown the increase of DDoS attacks and their steep financial impact on enterprises. Concerning Mirai, Bots are commanded to execute ten different types of DDoS flooding attacks.

Mirai was identified in August 2016 by the white hat security research group MalwareMustDie (Kolias et al., 2017). In Japanese, the definition for Mirai is “the future” (Kolias et al., 2017). This section consists of the following sub-sections (a) a discussion of initial Mirai attacks; (b) a discussion of the Mirai source code that was publicly released; (c) a discussion concerning the Bot spreading mechanism; (d) a discussion of Mirai for rent; (e) a discussion of the prominent Mirai Dyn attack; (f) a discussion of a bit coin miner variant; (g) an overview of Satori variants; (h) a summary of Mirai and variants.

Initial Mirai attacks. September 2016, the Mirai botnet performed DDoS attacks on internet services. Researchers Kolias et al. (2017) emphasize the strength of the Mirai botnet:

Krebs was hit with 620 Gbps of traffic, “many orders of magnitude more traffic than is typically needed to knock most sites offline.” At about the same time, an even bigger DDoS attack using Mirai malware—peaking at 1.1 Tbps—targeted the French webhost and cloud service provider OVH (p. 81).

After the initial attacks on Krebs website and French webhost and cloud service provider OVH, the Mirai source code was publicly released (Sinanović & Mrdovic, 2017). The following sub-section describes the public release of the Mirai code.

Mirai source code publicly released. The author, Anna-Senpai released details on the Mirai architecture as well as how to compile the Mirai code and operate the C&C (Infosecchazzy, 2019). Concerning the number of Bots, Anna-Senpai believes that originally
Mirai was able to spread to 380,000 Bots but after the initial attacks on Krebs Internet Service Providers (ISP) have been able to shutdown Bots and Mirai has lessened to 300,000 Bots (Infosecchazzy, 2019).

Mirai was able to spread to 380,000 bots from telnet only. Telnet is an un-encrypted protocol that allows the Bot to remotely access the new Bot Victim with a factory default user-id and password (Kolias et al., 2017). Concerning the number of Bots, Anna-Senpai is claiming that Mirai contained a maximum of 380,000 Bots during the Kreb attack. After the Kreb attack, Mirai contained 300,000 Bots. Since the attack, the number of Bots reduced by 80,000. Anna-Senpai emphasizes this reduction in Bots is due to ISPs shutting down Bots. The next sub-section describes the Mirai spreading mechanism.

**Mirai spreading mechanism.** Mirai spreads by infecting IoT devices such as webcams, digital video recorders, and home wireless routers that run the BusyBox operating system (Kolias et al., 2017). Mirai spreads to IoT devices with remote access credentials stored in their firmware (J. A. Jerkins, 2017). The Bot contains 62 username/password dyads for remotely accessing the new Bot Victim (Kambourakis et al., 2017). The “real-time-load”, described by Anna Senpai (Infosecchazzy, 2019), is the distributed mechanism where a Bot identifies a new Bot Victim, via telnet with the factory default user-id and password, for upload of the malware by C&C (Kolias et al., 2017). Once the malware has been uploaded by the C&C, the new Bot Victim becomes a Bot, and the iterative process begins for finding a new Bot Victim. The next sub-section describes Mirai for rent.

**Mirai for rent.** After the public release of the Mirai source code, a hacker offers Mirai for rent with as many as 400,000 Bots (Kolias et al., 2017). The hacker was interviewed by security blogger (Cimpanu, 2016). Cimpanu (2016) was able to gather information on how the hacker BestBuy was able to spread Mirai to contain 400,000 Bots. The Mirai source code was modified to include Secure Shell (SSH) for remote access to IoT devices as well as a zero-day vulnerability.

The hacker BestBuy created a variant of Mirai for rent. SSH was added for remote access. SSH utilizes an encrypted protocol for remote access (Venkatachalam, 2007). In addition to telnet, the Bot utilizes the encrypted SSH protocol to remotely access a new Bot Victim with the factory default user-id and password. Support was added to exploit a zero-day vulnerability. Experimentation performed by security researcher Whittier-Jones (2018)
shows that IoT devices can be exploited by many old and patched security exploits. During the interview, Cimpanu (2016) gathered additional information for a mitigation technique implemented by BestBuy. BestBuy was able to bypass DDoS mitigation systems by spoofing Bots. Spoofing a Bot includes faking the Bot IP address.

This variant of Mirai implemented the spoofing to fake the Bot IP address to avoid detection (Takemori, Fujinaga, Sayama, & Nishigaki, 2009). The Mirai for rent variant contains three features that the Original Mirai does not. SSH functionality was implemented for remote access. A zero-day vulnerability was exploited. The Bot’s IP address was spoofed. With these added features, the Mirai for rent variant spread to 400,000 Bots. The next sub-section describes the prominent Dyn attack.

**Prominent Dyn attack.** On October 2016 Mirai performed a DDoS attack on service provider Dyn (Kambourakis et al., 2017). With 100,000 bots, Mirai DDoS attacks on service provider Dyn triggered the inaccessibility to hundreds of websites in Europe and North America (Sinanović & Mrdovic, 2017). A heat map is a graphical representation of data represented as colors (Wilkinson & Friendly, 2009). Shown below in Figure 1 is a heat map depicting the wide-spread internet outage.

![Figure 1. Dyn attack heat map.](image)

Taken from ("Level3 outage map," 2018).
Shown above in Figure 1, the heat map depicts a large concentration of web-sites unavailable in the northeast and west coast of the United States. High traffic websites such as Twitter, Netflix, Reddit, GitHub and many others became unavailable for several hours (Kolias et al., 2017). During the press briefing the US Press Secretary Josh Earnest was questioned about the Internet outage. His response, "the DHS is tracking it." ("Press Briefing by Press Secretary Josh Earnest, 10/21/2016," 2016). The next sub-section explains a bitcoin miner variant.

**Bitcoin miner variant.** In March 2017, a variant appeared with bitcoin miner functionality (Kambourakis et al., 2017). Bitcoin is a popular cryptocurrency (Dziembowski, 2015). Bitcoin mining is a computational process that is used to verify Bitcoin transactions for profit (Vilim, Duwe, & Kumar, 2016). The variant utilizes the power accumulated from IoT devices to mine the digital currency for financial gains. Besides bitcoin mining, the ELF Linux Mirai variant has the abilities to execute Structured Query Language (SQL) injection attacks.

SQL injection occurs when an attacker inserts a malicious SQL query into the web application to manipulate data or gain access to the database (Yeole & Meshram, 2011). For this variant, the Bot is trying to execute the SQL injection attack. Not only is the Bot performing an SQL injection attack, in addition the Bot is mining for bitcoins. The original Mirai did not perform SQL injection attacks nor did not mine for bitcoins for profit. This variant shows Mirai evolving into a different attack other than a DDoS attack, which is SQL injection. Not performing an attack, such as a DDoS or SQL injection, the Bots are mining for bitcoins. The following sub-section provides an overview of Satori variants.

**Satori variants.** Recently, on December 12, 2017, the Satori variant was identified (Arzamendi, Bing, & Soluk, 2018). The first version of Satori distinguished itself from Mirai in that its spreading mechanism exploits two vulnerabilities in IoT devices: (1) a “zero-day” in Huawei’s home gateway and (2) a command execution vulnerability in Realtek’s Universal Plug and Play (UPNP) Simple Object Access Protocol (SOAP) interface (Arzamendi et al., 2018). Both Satori versions were clearly intended to target specific types of devices, unlike Mirai, which infects any device running BusyBox via telnet with a factory default username and password (J. A. Jerkins, 2017). Other versions of Satori are similar to Mirai and use telnet to propagate (Arzamendi et al., 2018).
The fourth variant of Satori, is the first known Argonaut RISC Core (ARC) malware (Arzamendi et al., 2018). Adding ARC malware greatly expands the potential for bots. ARC processors have been licensed by more than 190 companies and are deployed in more than 1.5 billion products a year (Arzamendi et al., 2018). The Satori variant shows how a Mirai variant is going beyond the brute-force spreading mechanism of the “real-time loader” by exploiting two firmware vulnerabilities. Also, Satori is expanding to an additional large set of potential bots from the ARC processor. The next sub-section provides a summary of Mirai and variants.

**Mirai and variants summary.** Literally, Mirai variants are created daily (Kolias et al., 2017). With many variants created since the Mira code was publicly released, it would be impossible to discuss each one. The variants chosen provide insight into the evolution of Mirai variants. Mirai first appeared on the scene to perform DDoS attacks on internet services. Anna-Senpai released the code to a public forum shortly after. After the code was released, variants started to be created. Some of the variants are similar to Mirai and are telnet-based botnets that execute DDoS attacks. The Mirai for rent variant added an additional remote access method for SSH, exploited a zero-day vulnerability, and spoofed Bots. Other variants, such Satori, are not telnet-based but exploit vulnerabilities for propagation. Satori included IoT devices built upon the ARC processor. A variant was discovered that mines for bitcoins for profit. Variants are utilizing new methods for propagation, exploiting firmware vulnerabilities, developing strategies to avoid detection, and are performing DDoS as well as non-DDoS attacks. In December 2017, three hackers pleaded guilty to computer-crimes charges for creating and distributing Mirai ("Justice Department Announces Charges and Guilty Pleas in Three Computer Crime Cases Involving Significant DDoS Attacks," 2017). To grasp a deeper understanding of Mirai, with a particular focus on how a Bot finds a new Bot Victim, the next section will discuss the Mirai components.

**Mirai Components**

The previous section discussed Mirai and variants. In September 2016, the Mirai botnet performed DDoS attacks on internet services. After the initial attacks on Krebs website and French webhost and cloud service provider OVH, the Mirai source code was
publicly released (Sinanović & Mrdovic, 2017). The author, Anna-Senpai released details on
the Mirai architecture as well as how to compile the Mirai code and operate the C&C
(Infosecchazzy, 2019). The “real-time-load” is explained as a distributed mechanism where a
Bot identifies a new Bot Victim, via telnet with the factory default user-id and password, for
upload of the malware by C&C (Kolias et al., 2017). After the public release of the Mirai
source code, a hacker offers Mirai for rent with as many as 400,000 Bots (Kolias et al., 2017).
On October 2016 Mirai performed a DDoS attack on service provider Dyn (Kambourakis et
al., 2017). With 100,000 bots, Mirai DDoS attacks on service provider Dyn triggered the
inaccessibility to hundreds of websites in Europe and North America (Sinanović & Mrdovic,
2017). In March 2017, a variant appeared with bitcoin miner functionality (Kambourakis et
al., 2017). Recently, on December 12, 2017, the Satori variant was identified (Arzamendi et
al., 2018). The Satori variant shows how a Mirai variant is going beyond the brute-force
spreading mechanism of the “real-time loader” by exploiting two firmware vulnerabilities.
This next section discusses the Mirai components consisting of (a) a discussion of the Mirai
operation and communication steps; (b) a discussion of the Mirai malware; (c) a discussion of
bot scanner statistics; (d) a summary focused on the Mirai components related to the focus of
this study. The next sub-section discusses Mirai operational and communication steps.

**Operation and communication steps.** Researchers Kolias et al. (2017) study Mirai
operation and communication. Kolias et al. (2017) explain Mirai being comprised of four
components (a) the Bot which is responsible for searching and finding new Bot Victims as
well as executing DDoS attacks; (b) the C&C server which provides a CLI to check
component status and command a DDoS attack; (c) the loader which uploads the malware
onto the new Bot Victim; (d) the report server which maintains a database with details about
the Bots. Below, Figure 2 shows the key steps for the operation and communication for the
Mirai components.
In Figure 2 the components are depicted along with the step that corresponds with the operation and communication between components. Steps [1. Brute force, 2. Report, 3. Check status, 4. Infect command, and 5. Malicious binary] constitute the distributed spreading mechanism, the “real-time-load”, which is described by Anna-Senpai (Infosecchazzy, 2019). Step 6. Attack command represents the C&C server commanding the Bot for executing a DoS attack. Step 7. Attack represents the Bot participating in a DDoS attack. The operational and communication steps are as follows:

- **In Step 1**, the bot brute-forces a potential new Bot Victim to discover IoT devices configured with factory default user-ds and passwords. There are 62 possible factory default user-id and password combinations.
- **For Step 2**, when a potential new Bot Victim is discovered by the Bot via remote access with a factory default user-id and password, the Bot forwards various new Bot Victim characteristics to the report server through a different port. The characteristics include the needed information, such as the IP address, user-id, password, and hardware architecture of the potential new Bot Victim, for the C&C loader server to upload the malware.
• In **Step 3**, via the C&C server, the botmaster frequently checks the report server to determine the new Bot Victims that require the upload of the malware as well as verifying the status of the Bots that comprise the botnet. Typically, a Tor server is utilized to disguise and keep the IP address of the C&C server anonymous.

• Concerning **Step 4**, after querying the report server for potential new Bot Victims that are not infected with the malware, the botmaster issues an infect command to the loader server, providing the necessary logon characteristics for remotely accessing the potential new Bot Victim.

• For **Step 5**, the loader remotely accesses the potential new Bot Victim and instructs the potential new Bot Victim to download and execute the malware. Once the malware is downloaded and executed it will attempt to protect itself from other malware by shutting down telnet and SSH services. Once the malware is loaded and executed, the new Bot can communicate with the C&C server to receive attack commands. For C&C communication between the C&C and the Bot, a domain name is hardcoded in the malware. The domain name is cnc.changeme.com. The botmaster can change the C&C IP address without modifying the malware and without extra communication between the C&C and the Bot.

• For **Step 6**, the botmaster commands all Bot instances to execute a DDoS against a targeted website. Parameters supplied by the C&C include the type and duration of attack, the IP addresses of the Bots, and the target IP address of the public facing website server.

• In **Step 7**, the Bots are commanded to start attacking the targeted server with one of 10 available DDoS attacks contained in the malware. The available DDoS attacks are Generic Routing Encapsulation (GRE), TCP, and HTTP flooding attacks.

**Step 1.** Brute force represents (a) a Bot scanning for a potential new Bot Victim; (b) a Bot identifying a new Bot Victim by remotely accessing the new Bot Victim with the factory default user-id and password. With the problem being discussed in Chapter 1 of this study, the focus for this study is a Bot scanning for a potential new Bot Victim, which is a sub-step described for brute-forcing. The next sub-section describes the Bot malware.
**Bot malware.** Recent research by (Sinanović & Mrdovic, 2017) performed static analysis of the malware resident on the Bot. The malware contains three main modules (a) attack; (b) killer; (3) scanner. The Attack module is responsible for executing the DDoS attack commanded from the C&C. The Killer module is responsible for killing processes that are associated with ports that Mirai utilizes. The Scanner module scans for potential new Bot Victims. The next sub-section describes the malware attack module.

**Malware attack module.** The Attack module parses the command received from the C&C server and launches the Denial of Service (DoS) attack. There are ten different DoS attack functions. Each DoS command contains its own procedure for generating the commanded DoS attack network traffic. The functionality of the malware attack module is not part of the focus for this study. The next sub-section describes the malware killer module.

**Malware killer module.** The Killer module kills processes associated with ports 22, 23 and 80 and prevents applications from using these ports again. Port 22 is associated with SSH. Port 23 is used by telnet. Port 80 is associated with Hypertext Transfer Protocol (HTTP). HTTP allows for the communication between web browsers and web servers (Mogul, 2002). By killing applications that are using ports 22, 23, and 80, Mirai is freeing up those ports for use. Therefore, the ports for telnet, SSH, and HTTP are available on the Bot. Killer continually scans memory trying to find and kill similar malware running on the Bot. The malware eliminates worms (such as the Anime, qBot, and Bashlight) with a technique known as memory scraping (Kambourakis et al., 2017). The functionality of the malware killer module is not part of the focus for this study. The next sub-section describes the malware scanner module, which is the focus of this study.

**Malware scanner module.** The Scanner module uses telnet and a random generated public IP address to search for a potential new Bot Victim. Researchers Kolias et al. (2017) study the IP addresses that are excluded from the random generated IP address. The US Postal Service, the Department of Defense, the Internet Assigned Numbers Authority, General Electric, and Hewlett-Packard are black-listed from the random generated IP list. Suggested by Kolias et al. (2017), Mirai avoids propagating to certain IP addresses in order to avoid detection from the US government. Besides black-listing government IP addresses, Mirai does not contain any other limitations for generating random IP addresses.
Besides generating a random IP address, the Bot scanner code contains functionality for creating a socket and performing a TCP handshake (Conley, 2017). A socket is a network interface that allows the Bot to communicate with TCP/IP over the internet (Koutsoubelias & Lalis, 2013). Once the socket has been initialized, a TCP Handshake is performed with the random generated IP address. Researchers study TCP across the internet (Arlitt & Williamson, 2005). The TCP handshake exchanges Synchronize (SYN) and Acknowledgement (ACK) packets to establish a connection between hosts (Arlitt & Williamson, 2005). The Bot attempts to establish a connection to a new Bot Victim on ports 23 and 2323. The research performed by Conley (2017) supports the research performed by (Šemić & Mrdovic, 2017). Semic and Mrdovic (2017) discovered that after the Bot establishes a connection, a telnet handshake occurs. The next sub-section presents bot scanner statistics.

**Bot scanner statistics.** When the Bot has discovered a potential new Bot Victim, a list of 62 factory default user-id and password combinations are used to gain remote access to the new Bot Victim. The new Bot Victim logon credentials are sent back to the reporting server (Sinanović & Mrdovic, 2017). When the bot successfully gains access to a new Bot Victim, the Bot reports the new Bot Victim IP and logon credentials to the C&C server for malware infection and then continues scanning for a potential new Bot Victim (J. A. Jerkins, 2017). The Bot initiates a maximum of 128 connections per second (Kambourakis et al., 2017). Therefore, the Bot can connect to 128 potential new Bot Victims per second. Once a minute, the Bot sends the new Bot Victim credentials to the report server on port 80 (Kambourakis et al., 2017). The Bot establishes a raw socket connection to C&C server to receive DoS commands (Kambourakis et al., 2017). The botmaster uses the Command Line Interface (CLI) on the C&C server to command attacks as well as manage Mirai (J. A. Jerkins, 2017). The next sub-section provides a summary of Mirai components focused on a Bot scanning for a potential new Bot Victim.

**Mirai components summary.** Recent research by (Sinanović & Mrdovic, 2017) performed static analysis of the malware resident on the Bot. The malware contains three main modules (a) attack; (b) killer; (c) scanner. With the focus for this study being a Bot scanning for a potential new Bot Victim, the Scanner module is the central focus for this study. The Bot attempts to establish a connection to a new Bot Victim on ports 23 and 2323.
The Scanner module uses telnet and a random generated public IP address to search for a potential new Bot Victim. When the Bot successfully gains access to a new Bot Victim, the Bot reports the new Bot Victim IP and logon credentials to the C&C server for malware infection and then continues scanning for a potential new Bot Victim (J. A. Jerkins, 2017). The next sub-section surveys peer-reviewed journals focused on Mirai mitigation strategies.

**Mirai Mitigation**

The previous section discusses Mirai components. Kolias et al. (2017) explain Mirai being comprised of four components (a) the Bot is responsible for searching and finding new Bot Victims as well as executing DDoS attacks; (b) the C&C server provides a CLI to check component status and command a DDoS attack; (c) the loader uploads the malware onto the new Bot Victim; (d) the report server maintains a database with details about the Bots. Being part of the first component described by Kolias et al. (2017), the focus for this study is a Bot scanning a potential new Bot Victim. The Bot malware is constantly generating random IP addresses and attempting to establish a connection to the potential new Bot Victim via a TCP handshake (Šemić & Mrdovic, 2017).

This next section discusses the Mirai mitigation consisting of (a) a discussion of IoT vulnerabilities related to weak passwords; (b) a discussion of cataloguing IoT devices vulnerable to Mirai infection; (c) a discussion inoculating IoT devices from Mirai infection; (d) a discussion protecting IoT devices from Mirai infection; (e) a discussion of network traffic analysis that suggests network signatures for Mirai detection; (f) a summary of Mirai mitigation.

**Weak passwords.** Recent research conducted by (Whitter-Jones, 2018) reviews IoT security. The IoT attack surface includes Plug and Play mechanics of devices that are vulnerable to brute forcing of default credentials and DoS attacks. There is a problem with the integration of IoT devices within an already established network domain that utilized Active Directories and policies. IoT devices do not have the capabilities of connecting to Active Directory. Group policies and password policies are not applied. Device setup by an individual could allow for weak passwords (Whitter-Jones, 2018). Concerning default credentials and access controls, Whitter-Jones (2018) emphasize that requiring consumers to change default passwords would provide a greater layer of security and would reduce the risk
of Mirai infection. Researcher Jerkins (2017) explains that manufacturers are not currently motivated by market forces or regulatory requirements to improve the security of IoT. Consumers are not concerned with security. The next sub-section describes cataloguing IoT devices vulnerable to Mirai infection.

**Cataloguing vulnerable IoT devices.** J. A. Jerkins (2017) creates a Mirai variant to catalog vulnerable devices. The device type, manufacturer, firmware version, and network address are catalogued. An email is sent to the owner of the network for the vulnerable device. There are legal challenges with this approach for securing IoT. Laws, statutes, and regulations exist concerning computer intrusion and abuse in the United States (J. A. Jerkins, 2017). Researchers Jerkins and Stupiansky (2018) clarify that one of the most frequently exploited design flaws is being unable to change device default credentials since the default credentials are embedded into the firmware. Another vulnerability is the lack of a method for manufacturers to provide security updates. The next sub-section describes inoculating IoT devices from Mirai infection.

**Inoculation.** James A. Jerkins and Stupiansky (2018) propose a method for limiting the spreading of malware through inoculation epidemics. A harmless virus would search for IoT devices that are vulnerable to malware infection. Once a vulnerable device was encountered, the harmless virus would inoculate the device by execution of a reboot (James A. Jerkins & Stupiansky, 2018). A simulation showed that inoculated devices slowed down the spread of the malware. Researchers James A. Jerkins and Stupiansky (2018) believe that if the inoculation cannot prevent the Mirai epidemic from occurring it can slow down the Mirai infection and provide time to react to the Mirai infection. This research prevents the spread of Mirai by restarting the IoT device. A simulation has shown that restarting the IoT device can inhibit the spread of Mirai. The research is not focused on preventing the infection of IoT devices from the Mirai malware. The next sub-section discusses protecting IoT devices from infection.

**Protecting IoT devices from infection.** Current research conducted by authors Frank et al. (2017) conduct experiments for protecting an IoT device from Mirai. A device hardening script puts a wrapper around the firmware that prevents the upload of the malware from the loader. Another script executing in the background detects open ports associated with Mirai. The ports are closed, and the programs associated with the ports are killed
preventing further processing. The combination of the hardening and detecting scripts protect
the IoT device from Mirai malware infection.

Contrasting the research conducted by James A. Jerkins and Stupiansky (2018), the
research performed by Frank et al. (2017) does not require the IoT device to be rebooted to
eliminate the Mirai malware infection. Future research suggests developing network
signatures for determining indicators of compromise for the IoT device (Frank et al., 2017).
The next sub-section presents Mirai network traffic analysis.

**Network traffic analysis.** Conley (2017) analyzed Mira network traffic with a
network packet sniffer. Network packets were gathered for destination port 23. The contents
of the packets contained the factory default user-id and password. The dynamic analysis
performed by Conley (2017) supports recent research performed that Mirai is a telnet-based
botnet that contains a list of factory default user-ids and passwords for remote access (J. A.
Jerkins, 2017; Kambourakis et al., 2017; Kolias et al., 2017; Sinanović & Mrdovic, 2017).
Mirai does not try to avoid detection (Kolias et al., 2017). Network or host-based signatures
can detect Mirai (Kambourakis et al., 2017). Since the focus of this study is counting the
number of Bots and potential new Bot Victims from a recorded Bot scanner dataset, this study
does not focus on experimenting with network signatures to detect the Bot scanning
mechanism. Even though the focus of this research is not experimenting with network
signatures for Bot detection, the Bot scanner code review and the analysis of the Bot scanner
network packets from the dataset reveal the necessary information to develop network
signatures for Bot scanning detection.

During Mirai infection network traffic on telnet ports 23 and 2323 can be monitored,
which is barraged with logon attempts to gain access to the IoT device (Kambourakis et al.,
2017). Mirai contains a list of factory default user-ids and passwords which signatures can
detect (Jonsdottir, Wood, & Doshi, 2017). Current research performed by (Šemić & Mrdovic,
2017) with a honeypot, shows that the Bot establishes a connection, performs the telnet
handshake, and successfully logs in with the factory default user-id and password. Once the
Bot logs in, the honeypot receives several successive inputs (a) enable; (b) system; (c) shell;
(d) sh. These inputs are to attain access to the new Bot Victim shell (Šemić & Mrdovic,
2017).
Mirai botnet researchers Kolias et al. (2017) study communication patterns between the loader and a new Bot Victim, during the upload of the malware. Shown below, Figure 3 shows the network traffic between the loader and the new Bot Victim.

![Figure 3. Network patterns between the loader and the new Bot Victim.](image)

Figure 3. Network patterns between the loader and the new Bot Victim.
Taken from (Kolias et al., 2017, p. 821).

Figure 3 shows that patterns of the network traffic are indicative of the malware infection. Session times vary, but the type of messages, packet sizes, and the sequence of messages form a pattern. Similar to research performed by Kolias et al. (2017) for analyzing network traffic, researchers Sinanović & Mrdovic (2017) perform dynamic analysis of the network communication between the C&C and Bot during a DDoS attack. Experimentation performed by Sinanović and Mrdovic (2017) was successful in detecting Bot DDoS with network signatures. A signature was suggested to prevent Mirai infection from an external network. Supporting the view of Kolias et al. (2017), Sinanović and Mrdovic (2017) suggest it is possible to create network signatures for all parts of Mirai operation. Network signatures seem to be the best and easiest way to detect and stop Mirai. Future research is suggested for creating a more complex test environment to see how a Bot brute-forces a new potential Bot Victim. The next sub-section provides a summary for this section.

**Mirai mitigation summary.** The IoT attack surface includes Plug and Play mechanics of devices that are vulnerable to brute forcing of default credentials and DoS
attacks (Whitter-Jones, 2018). Researcher Jerkins (2017) explains that manufacturers are not currently motivated by market forces or regulatory requirements to improve the security of IoT. James A. Jerkins and Stupiansky (2018) propose a method for limiting the spreading of malware through inoculation epidemics. Research performed by James A. Jerkins and Stupiansky (2018) prevents the spread of Mirai by restarting the IoT device. A simulation has shown that restarting the IoT device can inhibit the spread of Mirai. Current research conducted by authors Frank et al. (2017) conduct experiments for protecting an IoT device from Mirai. A device hardening script puts a wrapper around the firmware that prevents the upload of the malware from the loader. Another script executing in the background detects open ports associated with Mirai. The ports are closed, and the programs associated with the ports are killed from further processing. Network traffic has been analyzed and signatures are suggested for Mirai detection (J. A. Jerkins, 2017; Koliás et al., 2017; Sinanović & Mrdović, 2017). The next section discusses network traffic analysis tools.

**Network Traffic Analysis Solutions**

Solutions have been created to analyze network traffic. These solutions vary in their capability and intended purpose. Two major categories of network traffic analysis tools are: packet sniffers and network intrusion detection systems. This section will discuss the features and limitations of packet sniffers and network intrusion detection systems. A summary is provided discussing how the Mirai Bot Scanner Summation Prototype addresses these limitations. The next sub-section discusses packet sniffers.

**Packet sniffers.** A packet sniffer, also known as a network or protocol analyzer, is a program running on a network attached device that passively records all network traffic (Asrodia & Patel, 2012). Some popular packet sniffers include Wireshark, Tcpdump, and Windump. Tcpdump is a Unix and Linux command-line program for capturing network traffic. Packet sniffers save captured network traffic to a PCAP file. Packet sniffers contain two forms of analysis: real time and batch. Real time analysis is performed as the network packets are being captured. Batch analysis is performed from a captured PCAP file (Asrodia & Patel, 2012). Based upon the network packet type, the packet sniffer parses the network traffic to the appropriate packet fields. Using a packet sniffer, the user is able to determine the network protocol associated with the packet as well as the packet field values.
Although packet sniffers capture network traffic, they provide limited functionality for summing packets based upon packet fields, storing summation results in a persistent database, producing customized reports, and providing customized graphical output. Tcpdump and Windump do not provide the functionality for summation of packets, a persistent database of summation analysis, the ability to produce reports, and display line graphs. Wireshark provides a GUI that does provides for searching thru the network packets based upon filters of packet field values. Wireshark can count the number of packets based upon the filter values. Although Wireshark contains a GUI with filtering and summation capabilities, similar to Tcpdump and Windump, a persistent database in not available for storing the summation and packet analysis results required for the research performed in this study. Each instantiation of a packet sniffer requires performing the summation and analysis of the PCAP file since the results are not stored in a persistent database. With the Bot scanner dataset containing 304 PCAP files and total size of 3.88 GB, a persistent database of summation results per PCAP file is required to produce the reports and line graphs near real time. Also, Wireshark does not contain the reports or line graphs necessary for providing the information corresponding to the terms specified in this research for Bots and potential new Bot Victims. The next sub-section discusses network intrusion detection systems.

**Network intrusion detection systems.** Network Intrusion Detection Systems, known as NIDS, monitor the network traffic and produce alerts that are logged in a file on the operating system. Also, NIDS can alert on a captured PCAP file. The alerts are forwarded to a Security Information and Event Management (SIEM) system for further analysis and reporting (Nagaraja & Kumar, 2018). There are two categories of NIDS: signature-based and anomaly based. Signature-based NIDS compare the captured packets against a database of known vulnerabilities and malicious signatures to detect cyber-attacks (Kumar & Singh, 2012). Snort, Suricata, and Bro are examples of signature-based NIDS (Thongkanichorn, Ngamsuriyaroj, & Visoottiviseth, 2013). Anomaly-based NIDS detect anomalies with a comparison to base-lined network traffic (W. Zhang, Yang, & Geng, 2009). DarkTrace is an example of an anomaly-based NIDS that deploys artificial intelligence and anomaly detection for an immune system that prevents cyber-attacks (Darktrace, 2018).

The limitations for NIDS is similar to the limitations for packet sniffers. NIDS send alerts to a SIEM. NIDS lack the capabilities for summation of network packets, summing
packets based upon packet fields, storing summation results in a persistent database, producing customized reports, and providing customized graphical output. Therefore, NIDS only produce alerts and cannot perform analysis required for research presented in this dissertation. NIDS rely upon a SIEM for providing a persistent database for storing the alerts and allowing for additional analysis with customized reporting and customized dashboards containing graphs. Requirements for a solution to analyze the Bot scanner dataset include being able to execute the solution on a PC with Windows 10 and the solution should be self-contained, not requiring additional solutions for reporting and graphing. Based upon requirements, SIEM integration with NIDS does not meet the requirements for a solution of this dissertation. The next subsection provides a summary comparing network traffic analysis solutions to the Mira Bot Scanner Summation Prototype.

**Network traffic analysis solutions summary.** Packet sniffers record network traffic. Tcpdump and Windump are command-line solutions for packet capture and analysis. Wireshark provides a GUI for packet capture and analysis. Packet sniffers do not contain the filters for identifying a Bot or a potential new Bot Victim. The Mirai Bot Scanner Summation Prototype contains the functionality for analyzing the Bot scanner dataset. The user of the Mirai Bot Scanner Summation Prototype does not need to know the filters or logic for summation. The user provides a date range and the summation is performed. The Mirai Bot Scanner Summation Prototype contains a persistent database of summation results based upon each PCAP file analyzed. The persistent database will provide for near real time reports and graphs. The reports and graphs contain the terms defined in this dissertation, Bot and potential new Bot Victim, thus providing information that is easily comprehensible to a user who has read this dissertation. NIDS suffer from similar limitations as packet sniffers. NIDS do not provide the functionality for the summation of the Bot scanner mechanism per PCAP file, persistence of the summation analysis per PCAP file, and reports and graphs with terminology from this dissertation. NIDS produce alerts in a log file that is sent to a SEIM for persistence and analysis. The Bot Scanner Summation Prototype is a self-contained solution that meets all of the requirements for analyzing the Bot scanner dataset. The next section provides a chapter summary for the literature review.
Chapter Summary

Chapter 2 focused on the literature review, which indicated that Mirai is a concern. Although research regarding Mirai exists, an anomaly is present regarding performing experimentation for identifying a Bot scanning for a potential new Bot Victim. The topics discussed in the chapter include (a) a discussion of DDoS attacks; (b) a discussion on a survey for Mirai and variants; (c) a discussion on the Mirai components; (d) a discussion on current research focused on Mirai mitigation; (e) a discussion of network traffic analysis solutions. The completed literature review answered questions regarding a Bot scanning for a potential new Bot Victim. Chapter 3 developed the design for the quantitative research method for this study.
CHAPTER 3

METHODOLOGY

The focus of this study is to perform experimentation with the Mirai Bot Scanner Prototype from a recorded Mirai Bot scanner dataset for tabulation of Bots scanning for potential new Bot Victims. Chapter 2 surveyed existing literature that applies to the background of this study and supports the focus of this study. DDoS attacks were reviewed. Mirai and variants were discussed. Mirai command and communication was studied. Mirai mitigation techniques were discussed. Network traffic analysis tools were reviewed and compared to the Mirai Bot Scanner Summation Prototype. In the following sections, Chapter 3 presents (a) the research method justification; (b) rationale for the research approach; (c) research data sources; (d) data analysis methods; (e) limitations and delimitations; (f) summary of the research methodology. The following sub-section discusses the research method justification.

Research Method Justification

DSR focuses on the development of solutions for practical problems (Cleven et al., 2009; Kampling et al., 2016; Offermann et al., 2009). DSR provides a framework for the development and evaluation of the Bot Scanner Summation Prototype (Cleven et al., 2009). There are many different research methods to investigate problems or implementations (Wieringa, 2014). DSR author Wieringa (2014) describes the following research methods: survey, single case mechanism, technical action research, and statistical difference-making experiments. Instances of an implementation or of a problem is surveyed to gathered statistics for evaluation and problem investigation. A single case mechanism experiment applies stimuli and explains the response. Technical Action Research (TAR) is the evaluation of an artifact in a real-world problem. Statistical difference-making experiments compare the average outcome to samples.
A survey was not appropriate as a research method for this dissertation since the focus of this research is solely technical. This study did not perform a statistical difference-making experiment since only one Bot Scanner dataset is experimented with. In TAR, single cases are experimented with, but the experimentation is not performed to answer a question. TAR is done in the field under real-world conditions. TAR is not an appropriate research method for this study since this study answers three research questions concerning the evaluation of the Bot scanner dataset with the Mirai Bot Scanner Summation Prototype. A single case mechanism best describes the research method for this study. Usually, single case experiments are performed in a laboratory to test a prototype. A prototype serves as inspiration for those that come later (Prototype, n.d.). A case study is performed evaluating the Bot Scanner Summation Prototype with a captured Bot scanner network dataset. The next section discusses the rationale for the research approach.

Rationale for Research Approach

Current research has performed experimentation to evaluate network traffic when the Mirai malware is uploaded onto the new Bot Victim (Frank et al., 2017; Kolias et al., 2017). Additional research has been focused on evaluating the network traffic from a Mirai DDoS attack (Kambourakis et al., 2017; Margolis, Oh, Jadhav, Kim, & Kim, 2017; Sinanović & Mrdovic, 2017). Additional research has focused on the Mirai malware code (J. A. Jerkins, 2017; Roses, 2016; Sinanović & Mrdovic, 2017). While current research has been focused on various aspects of the Mirai botnet, research is far from being exhausted covering the various communications and operations of Mirai.

Anna-Senpai, the creator of Mirai, boasts about the spreading of Mirai (Infosecchazzy, 2019). The Mirai spreading mechanism consists of a Bot scanning for a potential new Bot Victim and then brute-forcing the Bot Victim with a factory default user-id and password. The literature review of this study did not reveal research focused on evaluating network traffic from the Mirai spreading mechanism. Specifically, new research can be conducted to evaluate network traffic when a Bot is scanning for a new Bot Victim. Following the DSR research phases described by researchers Cleven et al. (2009), the research approach for the Mirai Bot Scanner Summation Prototype solution included in this section presents (a)
problem identification; (b) requirements specification; (c) design; (d) evaluation; (e) communication. The next sub-section provides a discussion for problem identification.

**Problem identification.** Review of network traffic analysis tools demonstrated that the requirements of this study could not be met for summation of a Bot scanner dataset with current network traffic analysis tools, such as packet sniffers and NIDS. Packet sniffers and NIDS do not contain the capabilities for summatting the Bot scanner dataset or the functionality for creating reports and line graphs based upon a date-range of the dataset. Packet sniffers and NIDS do not contain a persistent database for storing the Bot scanning summation for each PCAP file of the dataset. NIDS send alerts to a SIEM for additional analysis typically including reporting and dashboards. The Bot Scanner Summation Prototype was developed as a practical solution to meet the requirements of being self-contained to evaluate a Bot scanner dataset to summate Bots, potential new Bot Victims, and network packet totals. The next sub-section presents the requirements for the Bot Scanner Summation Prototype.

**Requirements specification.** The requirements for this study are based upon functional requirements, nonfunctional properties, and additional tool requirements for performing experimentation with the Mirai Bot Scanner Summation Prototype. The functional requirements define the required functions for the Mirai Bot Scanner Summation Prototype. Below, Table 1 provides a description of the functional requirements including the functions and their descriptions.

Table 1

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCAP</td>
<td>The captured network traffic must be in PCAP format.</td>
</tr>
<tr>
<td>SYN</td>
<td>The PCAP file must only contain Bot Scanner SYN network packets for summation.</td>
</tr>
<tr>
<td>Bot Summation</td>
<td>Bots and potential new Bot Victims summation from PCAP.</td>
</tr>
<tr>
<td>Network Packet Summation</td>
<td>TCP SYN and retransmission packets summation from PCAP.</td>
</tr>
</tbody>
</table>
Table 1 (continued)

*Mirai Bot Scanner Summation Prototype Functional Requirements*

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistent Database</td>
<td>The NoSQL MongoDB v1.14.6 provides a database for persistence of the summation. The MongoDB will contain the summation results from the Python scripts.</td>
</tr>
<tr>
<td>Python Programming Modules</td>
<td>All of the artificial scripts are written in Python v2.7. The required modules are:</td>
</tr>
<tr>
<td></td>
<td>- The Scapy v2.4 module is required for reading a PCAP file during summation.</td>
</tr>
<tr>
<td></td>
<td>- The Pandas DataFrame v0.23.0 module is required for providing a high-performance data structure for summation.</td>
</tr>
<tr>
<td></td>
<td>- The Bokeh 1.0.0 module provides the graphical capabilities.</td>
</tr>
<tr>
<td>Date Range</td>
<td>All Reports and Graphs require a date range. The date range allows for the monitoring of the Bot Scanner mechanism over time with the Reports and Graphs.</td>
</tr>
<tr>
<td>Reports</td>
<td>The reports will be produced from reading the summation stored in a table of the persistent database. Three required reports:</td>
</tr>
<tr>
<td></td>
<td>1. A report is required for summation of Bots and potential new Bot Victims.</td>
</tr>
<tr>
<td></td>
<td>2. A report is required for summation of network packets.</td>
</tr>
<tr>
<td></td>
<td>3. A report is required for the run-time of summation.</td>
</tr>
<tr>
<td>Graphs</td>
<td>The graphs will be produced from reading the summation stored in a table of the persistent database. Three required graphs:</td>
</tr>
<tr>
<td></td>
<td>1. A line graph is required for summated Bots.</td>
</tr>
<tr>
<td></td>
<td>2. A line graph is required for summated new potential Bot Victims.</td>
</tr>
<tr>
<td></td>
<td>3. A line graph will compare summated packet totals.</td>
</tr>
</tbody>
</table>

Shown above in Table 1, functional requirements are defined for the Mirai Bot Scanner Summation Prototype. The dataset is required to contain Bot scanning traffic.
recorded in *PCAP* file format. The dataset is limited to *SYN* network packets, representing Bots attempting to connect to potential new Bot Victims. The Mirai Bot Scanner Summation Prototype is required to summate Bots as well potential new Bot Victims. An additional requirement is the summation of network packets. The MongoDB v1.14.6 is required for providing a persistent layer of summation that is powerful and scalable (Chodorow, 2013).

Several *Python programming modules* are required. The Scapy v2.4 module is required for reading a PCAP file during summation. Scapy provides the functions for manipulating network traffic contained in a PCAP file (Biondi, 2011). The Pandas DataFrame v0.23.0 module is required for providing a high-performance data structure for summation (McKinney, 2010). The Bokeh 1.0.0 module provides the graphical capabilities (Barnard & Mertik, 2015).

A *date range* is required for reports and graphs. The date range provides the functionality for monitoring the Bot scanning mechanism by being able to assess the summation during a specified date period. Three *reports* are required for evaluating the Bot scanning mechanism. The required reports include a report summation of Bots and potential new Bot Victims, a separate report for summation of network packets, and another report for the run-time of summation. Three *graphs* are required for providing visual analysis of the Bot scanner dataset. The graphs include a line graph for summated Bots, a distinct line graph for summated new potential Bot Victims, and another line graph for comparing summated packet totals. Shown below, Table 2 contains the nonfunctional properties for the Bot Scanner Summation Prototype.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>A Dell PC running the Windows 10 Operating System (OS) with 12 Gigabytes (GB) of memory, 500 GB hard drive, and an Intel Core i5 2.20 Gigahertz (GHz) processor.</td>
</tr>
<tr>
<td>Development Costs</td>
<td>The development of the Mirai Bot Scanner Summation Prototype solution must be done with free and open tools. There is no budget for development.</td>
</tr>
</tbody>
</table>
Table 2 (continued)

*Bot Scanner Summation Prototype Nonfunctional Properties*

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational Costs</td>
<td>There are no operational costs. The Mirai Bot Scanner Summation Prototype is free to use and incurs no operational costs.</td>
</tr>
<tr>
<td>Self-containment</td>
<td>The Mirai Bot Scanner Summation Prototype does not interface with any other systems. It is self-contained. All of the components of the Mirai Bot Scanner Summation Prototype must reside on the same PC.</td>
</tr>
<tr>
<td>Real time analysis</td>
<td>The reports and line graphs must be generated real time.</td>
</tr>
</tbody>
</table>

Shown above for Table 2, the nonfunctional properties define the operation of the Mirai Bot Scanner Summation Prototype. The *hardware* requirements specify the computing environment for this study. Experimentation was performed on a Dell PC running the Windows 10 Operating System (OS) with 12 Gigabytes (GB) of memory, 500GB hard drive, and an Intel Core i5 2.20 GigaHertz (GHz) processor. The *hardware* requirement represents the suggested minimum hardware configure for the Mirai Bot Scanner Summation Prototype solution. Strict requirements state no *development* or *operational costs*. The Mirai Bot Scanner Summation Prototype is *self-contained* and does not require interfacing with any other systems. Experimentation for this study was performed with all the components of the Mirai Bot Scanner Summation Prototype residing on the same PC, including all of the Python scripts, the MongoDB database, and the Bot scanner dataset. *Real time assessment* is a requirement for reporting and the graphing. Besides Mirai Bot Scanner Summation Prototype requirements and nonfunctional properties, two additional tools are required for performing experimentation. Shown below in Table 3, the requirements for additional tools are described.
Table 3

Additional Tool Requirements

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Traffic Analysis</td>
<td>The Wireshark v2.6 packet sniffer is required for analyzing a packet of the Bot Scanner dataset (Wireshark, 2019). Experimentation performed in this study verifies the Bot scanner dataset by using Wireshark to sample a PCAP file to study the contents of the network packets.</td>
</tr>
<tr>
<td>Text Editor</td>
<td>The Sublime v3.0 text editor is required for performing the Bot scanner and Mirai Bot Scanner Summation Prototype code review. Sublime allows for scrolling thru the code as well as contains a search function that allows reviewing targeted lines of code based upon the search value (Sublime, 2019).</td>
</tr>
</tbody>
</table>

A requirement for the network traffic analysis is the Wireshark v2.6 pack sniffer. The Wireshark packet sniffer contains a GUI for analyzing the contents of the packets of the Bot scanner dataset (Asrodia & Patel, 2012). Sampling a PCAP file from the dataset, analysis performed with Wireshark verified the Bot scanner dataset for only containing network traffic for TCP SYN packets and retransmission packets. The Sublime text editor is required for the code review. Sublime contains features, such as scrolling thru the lines of code and searching for specific task, that make it a good choice for the text editor (Kinder, 2013). The requirements and nonfunctional properties have been defined. The next sub-section discusses the Mirai Bot Scanner Summation Prototype design work.

**Design work.** The design work for the Mira Bot Scanner Summation Prototype is based upon an architectural structure. An architectural structure consists of components that interact with each other. Typically, programs contain components that communicate with each other for a desired result (Wieringa, 2014). Shown below, Figure 4 depicts the architectural components of the Mirai Bot Scanner Summation Prototype.
Shown above in Figure 4, the **summation** component, or process, consists of reading the PCAP files of the dataset. The summation process tabulates each PCAP file to summate the Bots, potential new Bot Victims, and SYN and retransmission packets. The summation process stores the tabulated results from each PCAP file in a *persistent database table*. The **assessment process** queries the database for the summated data to produce reports and line graphs. The assessment of the Bot scanner dataset can only be performed from summated PCAP files. The summation process is independent of the assessment process. The summation component can be in the process of tabulating a PCAP file while the assessment component is querying the database. Both components, summation and assessment, are dependent upon the persistent database. The following sub-section discusses the summation component in detail.

**Summation.** In DSR a method is described as a type of artifact (Offermann et al., 2009). A method can describe an algorithm. Shown below for Figure 5, the summation method, or algorithm:

```plaintext
//Initialization
Total_Bots = 0, Total_Potential_New_Bot_Victims = 0, Total_SYN = 0
Total_Retransmission = 0, Total_Packets = 0, Starting_Time = 0, Ending_Time = 0
```
Packet_date = 0, L = [], S= [], SUBNETS = []
Starting_Time = now
Packet_date = date_from_filename(PCAP)

// Read the network packets of the PCAP file
Insert into list L the source and destination IP of each network packet

// Go thru each element of L
For i in L
  // Summate total packets
  Total_Packets = Total_Packets + 1

  // Determine subnet of destination IP
  Add Subnet(L[i].destination_IP) to SUBNETS

  // Unique source IP represents a Bot
  If the count(L[i].source_IP in L) == 1
    Total_Bots = Total_Bots + 1

  // Unique SYN packet
  If the count (L[i] in L) == 1
    Insert L[i].destination_IP into S
    Total_SYN = Total_SYN + 1

  // Retransmission packet
  If the count (L[i]) > 1
    Total_Retransmission = Total_Retransmission + 1

// Go thru each destination IP in S
For j in S
  // a unique destination IP in S represents a potential New Bot Victim
If the count(L[j] in S) == 1

Total_Potential_New_Bot_Victims =

Total_Potential_New_Bot_Victims + 1

Ending_Time = now

//Insert summation results into the database
Insert Total_Bots, Total_Potential_New_Bot_Victims, SUBNETS
Total_SYN, Total_Retransmission, Total_Packets,
Starting_Time, Ending_Time, Packet_date
Into Persistent Storage

Figure 5. Summation algorithm.

Shown above in Figure 5, the summation algorithm starts with initialization. The network packets are read, and the source and destination IP of each packet is inserted into list \( L \). Searching through the items in \( L \), Total_Packets is summed, Total_Bots is summed from the unique source_IP within \( L \), and the \( SUBNETS \) list is constructed from the destination_IP. The \( S \) list is built to contain the destination_IPs from unique SYN packets, Total_SYN and Total_Retransmission is tabulated. Once the search is complete through \( L \), \( S \) is searched to summate Total_Potential_New_Bots based upon unique destination_IP within \( S \). The summation algorithm analyzes network traffic of a Bot scanner dataset to identify a Bot scanning for a potential new Bot Victim. The summation algorithm is able to analyze Bot scanning network traffic to tabulate Bots, potential New Bot Victims, and Bot scanner network packets. The starting and ending time of the summation is tracked. The summation results are stored in the persistent database.

Instantiation describes implemented and prototype systems (Offermann et al., 2009). The Mirai Bot Scanner Summation Prototype is written entirely in Python and relies heavily upon the Pandas DataFrame for summatring the network packets based upon DataFrame query parameters. The Mirai botnet researcher will supply the PCAP file names for summation. For each PCAP file, the summation component must identify the Bots and potential new Bot
Victims. The Scapy `rdpcap` function allows the summation process to read all of the SYN packets contained in the PCAP file (Biondi, 2018). For each SYN packet, the source and destination IP are inserted in a Python list. The source and destination IP lists are put in a Pandas DataFrame for querying ("Pandas: powerful Python data analysis toolkit," 2018). Shown below, Table 4 contains the queries of the DataFrame for summation.

Table 4

*Pandas DataFrame Summation Results*

<table>
<thead>
<tr>
<th>Query Description</th>
<th>Summation Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query the number of unique source IP addresses</td>
<td>Summation of Bots</td>
</tr>
<tr>
<td>Query number of unique destination IP addresses contained in unique SYN packets</td>
<td>Summation of potential new Bot Victims</td>
</tr>
<tr>
<td>Query the number of unique SYN packets</td>
<td>Summation of SYN packets</td>
</tr>
</tbody>
</table>

Shown above in Table 4, The DataFrame is queried to determine the number of unique source IP addresses. Since the source IP represents the Bot, this query summates the Bots. The number of unique destination IP addresses contained in unique SYN packets is queried from the DataFrame. A unique SYN packet contains a unique source and destination IP address combination. Since the destination IP represents the potential new Bot Victim, the potential new Bot Victims are summated from the unique SYN packets. The prototype identifies a Bot scanning for a potential new Bot Victim for the tabulation of Bots and potential new Bot Victims. Unique SYN packets represent SYN packets without a retransmission. The number of unique SYN packets are summated. Non-unique SYN packets represent retransmission packets. The amount of retransmission packets is calculated by subtracting the number of unique SYN packets from the total packet count of the PCAP file. Also, the runtime information, starting and ending date and time, is recorded for each PCAP summation. The summations and runtime information are stored in a persistent database. The next sub-section presents the details of the persistent database.

**Persistent database.** The summation totals for each PCAP file are stored in a MongoDB (Chodorow & Dirolf, 2010). MongoDB provides an interface to programming
languages, such as Python, for the persistent storage and retrieval of data. Research has shown MongoDB to be a powerful and scalable database for websites (Chodorow, 2013). Shown below, Table 5 presents the fields of the summation table within MongoDB.

Table 5

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packet_date</td>
<td>Date of the packet capture</td>
</tr>
<tr>
<td>dest_subnet</td>
<td>List of destination subnets</td>
</tr>
<tr>
<td>total_packets</td>
<td>Packet summation</td>
</tr>
<tr>
<td>total_syn_packets</td>
<td>Unique SYN packet summation</td>
</tr>
<tr>
<td>total_retransmission_packets</td>
<td>Retransmission packet summation</td>
</tr>
<tr>
<td>total_bots</td>
<td>Bot summation</td>
</tr>
<tr>
<td>total_potential_new_bot_victims</td>
<td>Potential new Bot Victim summation</td>
</tr>
</tbody>
</table>

Shown above in table 5, the `packet_date` field holds the date determined from the name of the PCAP file of the Bot scanner dataset (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). The `dest_subnet` contains a list of destination subnets from the dataset. `Total_packets` represents the total amount of network packets contain in the PCAP file. `Total_syn_packets` contains the summated unique SYN packets. `Total_retransmission_packets` contains the summated retransmission packets. `Total_bots` represents the Bot summation. The field `total_potential_new_bot_victims` contains the summation for the potential new Bot Victims. The persistent database provides a data source for assessing the summation with reports and graphs. Besides the summation table, another table stores the runtime for each PCAP summation. The runtime table contains the PCAP date along with the start and ending date and time for the summatiting the PCAP file. This information helps to access summation performance. The following sub-section provides details for the assessment component.

**Assessment.** The assessment component provides reports. The reports require a date range for searching the database. Detail and summary reports are produced from records contained in the database from the summation process. The detail and summary reports are
similar in functionality. Based upon the report specified, dataset or runtime, the detail report will report on each record from the database that resides within the date range. Concerning summary reports, the database is queried in the same manner as detailed reports with the difference being that each database record is summated. Figure 6 presents the reporting algorithm:

```plaintext
// Initialization
L = []
S = []

// Search the database
// Add to list L the records of the database that fall within the date range
L ← Select records from Summation database that fall within a date range.

// Sum up the L list
If the report is a summary report
   For i in L
      S = S + i

// Report on the details from L
If the report is a details report
   For i in L
      Print i

// Report on summary from S
If the report is a summary report
   Print S
```

*Figure 6. Assessment reporting algorithm.*

Shown above in Figure 6, the assessment reporting algorithm starts by initializing the L and S list. The L list represents the detail records from the summation database that fall
within the date range. The $S$ list represents the summation of the details. Based upon a date range parameter, the detail records are loaded into $L$. If a summary report is required, each item in $L$ is summed together to form $S$. For a detail report, the contents of $L$ is reported on. Concerning a summary report, the contents of $S$ is reported on.

The botnet researcher selects the desired report and provides a date range for the assessment. The assessment component searches the MongoDB database for the summated records that fall within the date range. The reports generated from the assessment component are:

- PCAP Runtime Details;
- PCAP Runtime Summary;
- Dataset Details;
- Dataset Summary.

There are two PCAP reports and two Dataset reports. The PCAP Runtime Details report provides the detailed runtime information for each PCAP file from the persistent runtime table. The detailed runtime information includes the starting and ending date and time for each PCAP file. The PCAP Runtime Summary report summarizes the runtime information for the PCAP files within the date range. The summary information includes the number of PCAP files within the date range and the average processing time. The Dataset Details report provides the PCAP details from the summation table. It provides the number of packets, the number of Bots, and the amount of potential new Bot Victims per PCAP. The Dataset Summary report summarizes the Bot scanning mechanism from the PCAP files within the date range.

Besides detail and summary reports, graphs are provided for visual assessment. Similar to reports, the graphs require a date range for searching the database. Graphs are produced from records contained in the database from the summation process. Similar functionality is required for generating the various graphs provided by the assessment component. Figure 7 presents the graphing algorithm:
// Initialization
L = []

// Search the database
// Add to list L the records of the database that fall within the date range.
L ← Select records from Summation database that fall within a date range.

// Draw line_graph from list L
For i in L
    draw_line_graph(i)

*Figure 7. Assessment graphing algorithm.*

Shown above in Figure 7, the assessment graphing algorithm initializes the L a list. The L list is constructed from the database search for the records that fall within the date range. A line graph is drawn for each item, or field, in L.

The researcher selects the desired line graph and provides a date range for the assessment. The assessment component searches the MongoDB database for the summated records that fall within the date range. Several line graphs are available for assessment:

- Bot Totals;
- Potential new Bot Victim Totals;
- Bot Averages;
- Potential new Bot Victim Averages;
- Network Packet Totals.

Based upon the date range provided by the botnet researcher, the summation table is queried. The Bokeh Python library contains a function for producing a line graph (Team, 2014). The Mirai Bot Scanner Summation Prototype utilizes the Bokeh line function for plotting the line graph. A distinct line graph is produced for Bot Totals, Potential new Bot Victim Totals, Bot Averages, Potential new Bot Victim Averages, and Network Packet
Totals. The reports and line graphs provide the necessary information for the researcher to access, or monitor, the Bot scanning mechanism over time. The next sub-section presents the artifacts.

**Artifacts.** Artifacts can be methods, techniques, notations, and algorithms in software (Wieringa, 2014). The Mirai Bot Scanner Summation Prototype consists of Python programs that contain the methods, or functions, for implementing the algorithms for the summation and assessment process. The prototype contains two components that interact with the persistent MongoDB database. The summation component summates the Bot scanner dataset while the assessment component provides reports and graphs. Both components are written in the Python scripting language. Shown below in Table 6, the Mirai Bot Scanner Summation Prototype contains summation and assessment Python scripts.

<table>
<thead>
<tr>
<th>Component</th>
<th>Python Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>Summation</td>
<td>Analyze_PCAP_Files.py</td>
</tr>
<tr>
<td>Summation</td>
<td>BotScanner.py</td>
</tr>
<tr>
<td>Assessment</td>
<td>BotScannerResults.py</td>
</tr>
</tbody>
</table>

Shown above in Table 6, the summation component consists of two Python scripts. The *Analyze_PCAP_Files.py* script enumerates the list of PCAP file names contained in the Bot scanner dataset. Each PCAP filename is summated with the *BotScanner.py*. The *BotScanner.py* script contains the functions for summation and database interface for persistent storage of the summation. The assessment component consists of the *BotScannerResults.py* script. The *BotScannerResults.py* script contains the functions for interacting with the database for generating the reports and graphs needed for assessment. To perform summation or assessment, the botnet researcher needs to execute the required functions within the Python scripts. In Chapter 5 Experimentation, a code review is performed for revealing the functionality of the Mirai Bot Scanner Summation Prototype Python scripts. The next sub-section discusses the evaluation step of the DSR process.
Evaluation. Often, prototypes are tested in a single case experiment (Wieringa, 2014). The artifacts, the Python scripts, are evaluated to determine the effectiveness of the Mirai Bot Scanner Summation Prototype. Experimentation is performed with the Mirai Bot Scanner Summation Prototype for summat ing and assessing a Bot scanner dataset. The summation component is evaluated for its ability to summate Bots, potential new Bot Victims, and network packets per PCAP file of the dataset. The database will be evaluated for containing the summation results. The assessment component is evaluated for its effectiveness for generating reports and graphs. Evaluation of the Mirai Bot Scanner Summation Prototype provides deeper insight into monitoring the Bot scanning mechanism over time. The next sub-section discusses communication.

Communication. The single case experiment was performed with the Mirai Bot Scanner Summation Prototype. The Python scripts that constitute the Mira Bot Scanner Summation Prototype are provided in the Appendices and are publicly available on Github. The evaluation and experimentation of the Mirai Bot Scanner Summation Prototype results were presented in this dissertation. The experimentation was presented in a technical manner so that researchers can gain a deeper understanding of the Mirai Bot scanner. The next subsection presents the summary.

Rationale for research approach summary. The phases for DSR artifact development commonly covers (a) problem identification; (b) requirements specification; (c) design; (d) evaluation; (e) communication. (Cleven et al., 2009). The Statement of Problem subsection for this study has identified the problem and determined the artifact type and focus. The requirements for performing the experimentation of this study was provided. The design and evaluation of the Mirai Bot Scanner Summation Prototype was presented for summat ing and accessing the Bot scanner dataset. The research performed in this study is meant to add to the common body of knowledge of the Mirai botnet. The Mirai Bot Scanner Summation Prototype was developed in hopes of inspiring researchers to further investigate Mirai. The next section discusses the research data sources.

Research Data Sources

The previous section described the DSR research process. This section discusses the research data sources and consists of the following sub-sections (a) an introduction to the
required research data sources; (b) a discussion of the Mirai code; (c) a discussion of the Bot scanning network dataset; (e) a discussion for the summary of the research data sources.

**Required research data sources.** Two data sources are utilized for this research. The first data source is the publicly available Mirai code (Gamblin, 2016). The second data source is from a captured Mirai Bot scanning network dataset gathered by the University of Southern California (*Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870*, 2016). A single case experiment is performed with the Mirai Bot Scanner Summation Prototype for evaluating the captured Mirai Bot scanning network dataset. The first data source, the Mirai code, is required for a Bot scanner code review. The next sub-section discusses the Mirai code.

**Mirai program code.** The Mirai code was exposed publicly on the hacking community web forum Hackforums (J. A. Jerkins, 2017). According to Kambourakis et al. (2017), since the release of Mirai’s source code Mirai variants are created daily. Security researcher (Gamblin, 2016) has created a repository for the original Mirai code exposed on Hackforums. Recent research has performed a code review for various components of Mirai (J. A. Jerkins, 2017; Kambourakis et al., 2017; Kolias et al., 2017). Simon Roses, a cybersecurity researcher, performed Mirai static code analysis. *Tintorera*, a static analysis tool, determined Mirai is a small project and not too complicated to review (Roses, 2016). Roses (2016) view point of Mirai code being a small project and not too complicated to review is supported by the current peer-reviewed research performed for analyzing Mirai and its various components (J. A. Jerkins, 2017; Kambourakis et al., 2017; Kolias et al., 2017).

Researchers Kolias et al. (2017) study Mirai operation and communication. Kolias et al. (2017) explain the Bot which is responsible for searching for and finding new Bot Victims as well as executing DDoS attacks. Discussed by Kolias et al. (2017), Mirai avoids propagating to certain IP addresses in order to avoid detection from the US government. Besides black listing IP addresses, Mirai does not contain any other limitations for generating random IP addresses. The Bot scanner module uses telnet and a random generated public IP address to search for a potential new Bot Victim (Kolias et al., 2017). A code review performed by security researcher Conley (2017) concentrates on a Bot connecting to a potential new Bot Victim. Conley (2017) supports the research performed by (Šemić &
Semic and Mrdovic (2017) discovered that after the Bot establishes a connection, a telnet handshake occurs.

Inspired by current peer-reviewed research (J. A. Jerkins, 2017; Kambourakis et al., 2017; Kolias et al., 2017), this study performs a code review for a Bot scanning for a potential new Bot Victim. The original Mirai code exposed publicly on Hackforums is reviewed (Gamblin, 2016). The code review performed in this study studies and consolidates the modules of the Bot functionality for scanning for a new potential Bot Victim. The code review studies the modules and network mechanisms required for a Bot to connect to a potential new Bot Victim. Experimentation is performed from a captured Mirai Bot scanning network dataset. The next sub-section discusses the Bot scanning network dataset.

**Bot scanning network dataset.** Mirai Bot Scanner Summation Prototype experimentation is performed with a Mirai network dataset gathered by the University of Southern California ([Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016](https://www.ripe.net/ripe/ripe-mirai-b-scanning-20160601/rev5870)). The dataset is limited to Bot scanning activity collection starting on June 1, 2016 and ending on March 20, 2017. The dataset is described as,

> The dataset represents scanning traffic from Mirai observed at B-Root. It includes only Mirai-identified TCP SYNs sent to ports 23 and 2323; other traffic is removed. Through 2016-11-20, we observe only traffic to one IP address (130.152.184.2). From 2016-11-20 onward it adds traffic to 192.228.79.0/24. ([Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016](https://www.ripe.net/ripe/ripe-mirai-b-scanning-20160601/rev5870)).

The University of Southern California provides root DNS services (USC, n.d.). The dataset only contains TCP SYN packets sent to telnet ports 23 or 2323. The TCP SYN packet is part of the TCP handshake, used by a Bot to connect to a potential new Bot Victim (Arlitt & Williamson, 2005). This dataset contains captured network packets of Bots attempting to connect to potential new Bot Victims. The next sub-section provides a summary of the research data sources.

**Research data sources summary.** Two data sources are required for this study. The first data source is the publicly available Mirai code (Gamblin, 2016). The Mirai code is
required for a code review to determine the Bot scanning mechanism. The second data source is from a captured Bot scanning network dataset gathered by the University of Southern California (*Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016*). Experimentation with the Mirai Bot Scanner Summation Prototype is performed with the Bot scanning dataset. The next section describes the data analysis methods.

**Data Analysis Methods**

The previous section discussed the two data sources that are required for this study. This section discusses the data analysis methods and consists of the following sub-sections (a) a discussion of the Bot malware code review focused on the Bot scanner; (b) a discussion of the Bot scanner network dataset; (c) a discussion of analyzing a PCAP file from the Bot scanning dataset; (e) a discussion of the Mirai Bot Scanner Summation Prototype for performing experimentation with the Bot scanning dataset; (f) a summary of the data analysis methods. The following sub-section describes the code review.

**Code review.** The first data source required is the publicly available Mirai code (Gamblin, 2016). A code review requires manually reviewing the code. The code review for this study focuses on the modules for the Bot scanning for a potential new Bot Victim. The Mirai code was downloaded onto a PC for performing the code review.

The Sublime text editor contains features required for the code review (Kinder, 2013; Sublime, 2019). Sublime allows for scrolling thru the code as well as contains search functionality which allows for targeted searches which is beneficial for following the flow of the Bot scanner code. The Bot scanner code is reviewed to study the functionality associated with a Bot connecting to a potential new Bot Victim. An example of the Bot scanner code review performed with the Sublime text editor is shown in Figure 8.
Figure 8. Sublime text editor.

Shown above for Figure 8, the Sublime editor provides the line number. Key words are color coded. The tail end of the Bot scanner code is shown below in Figure 9.

Figure 9. Tail end of Bot scanner code.

Shown above in Figure 9, the Bot scanner code contains 991 lines of C programming code. With the use of the Sublime editor, 991 lines of C code is reviewed for the functionality...
of a Bot scanning for a potential new Bot Victim. Besides performing the code review with the Sublime editor, experimentation is performed with a captured Mirai Bot scanning network dataset. The next sub-section discusses the captured Bot scanning network dataset.

**Bot scanning network dataset.** The second data source is from a captured Mirai Bot scanning network dataset gathered by the University of Southern California (*Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016*). The dataset contains Bot scanning activity collection starting on June 1, 2016 and ending on March 31, 2017. A single case experiment with Mirai Bot Scanner Summation is performed with the captured Bot scanning dataset.

The dataset was downloaded on to a PC. The dataset contains two folders. A folder for the year 2016. Another folder for the year 2017. Inside each folder there is a folder of packet capture files associated for each day. An example of the contents of a daily folder is shown below in Figure 10.

![Figure 10. Example of a daily packet capture.](image)

Shown above in Figure 10 is an example for the contents of each packet capture folder. The 2016 daily 2016-06-01.pcap folder contains the Packet Capture (PCAP) file. A PCAP file is in a standard format for captured network traffic that can be read by network monitors or network pack sniffers, such as Wireshark (*Buczak et al., 2016*). Each daily folder contains a packet capture, representing the Bots scanning for potential new Bot Victims for that day. Each 2016 and 2017 daily folder contain one PCAP file. The 2016 daily folders contain 214 PCAP files with a total of 1.66 GB in size. The 2017 daily folders contain 90 PCAP files with a total of 2.22 GB in size. A total of 304 PCAP files comprise the captured
network data with a total size of 3.88 GB. The next sub-section describes the sample network packet analysis.

**Sample network packet analysis.** This study analyzes a sample of the network packets with Wireshark. Wireshark is a tool for analyzing network traffic (Jillepalli, Leon, & Sheldon, 2018). An example of analyzing a PCAP file with Wireshark is shown below in Figure 11.

![Figure 11. Analyzing a PCAP file with Wireshark.](image)

Shown above in Figure 11, the 2016-12-20.pcap file is analyzed with Wireshark. All of the network packets are loaded into Wireshark for manual analysis. Wireshark provides the packet number, Time, Source, Destination, Protocol, Length, and Info for each network pack. Concerning the experimentation and answering the research questions, fields of interest are (a) the Source which represents the IP address of the Bot; (b) the Destination which represents the IP address of a potential new Bot Victim; (c) the Info which contains the source port as well as the destination port of 23 or 2323. The manual analysis performed with Wireshark will sample the PCAP file to evaluate the Bot scanner traffic. To analyze all the PCAP files and perform experimentation to answer the research questions, a utility is needed. Analyzing the PCAP file within Wireshark, shows SYN packets that require retransmission. Shown below in Figure 12 is an example of a retransmission SYN packet.
Figure 12. Analysis of retransmission packet with Wireshark.

Shown in Figure 12, Wireshark highlights the retransmission packets in black. The information for the packet indicates a TCP Retransmission. The TCP retransmission packet occurs when the target IP has not acknowledged the SYN packet as part of the TCP handshake (Go, Kune, Woo, Park, & Kim, 2013). There are many reasons why a destination does not acknowledge a SYN packet. Some of the reasons a destination may not acknowledge the SYN packets are (a) IP address is not assigned to an IoT device; (b) destination IP not listening on the destination port; (c) a firewall has blocked network traffic. For the evaluation and experimentation of this study, a retransmission packet represents a Bot trying to connect to an IP address that is not vulnerable to Mirai infection or that does not have a device assigned to that IP address. The Bot is unable to connect to a potential new Bot Victim. The next sub-section describes performing the experimentation.

Performing experimentation. The Mirai Bot Scanner Summation Prototype components consist of Python programs to perform the experimentation. The summation component contains Python modules for Bot tabulation. The assessment component contains Python modules for producing reports and graphing. Both components, summation and assessment, interact with the database. The summation component stores the PCAP
summation results on the database. The assessment module searches the database for producing the reports and graphs.

During experimentation, following the completion of the summation process, the database will be verified to contain the Bot scanning summation. The results from the assessment reports and line graphs will be verified from the PCAP summated records in the database. The reports and line graphs will be experimented over a date range of the Bot scanning dataset. The reports and line graphs will be assessed for their effectiveness for providing the summation results from the Bot scanner dataset. The next sub-section provides a summary.

Data analysis methods summary. This section presents the research data sources. A code review for the Bot scanner was discussed. Details of the captured Bot scanner network dataset was provided. Sample network packet analysis was performed for SYN and retransmission packets. Single case experimentation with the Mirai Bot Scanner Summation Prototype components interacting with the recorded Bot scanning dataset was discussed. The next section discusses the limitations and delimitations of the research methodology for this study.

Limitations and Delimitations

This selected research methodology filled a void for studying the Bot scanner. Although current peer-reviewed research has experimented with analyzing network traffic, the focus of those studies has not been on a Bot scanning for a potential new Bot Victim. It is possible that experimentation exists for the Bot scanner, but there was a lack of references in peer-reviewed journals during the literature review.

Delimitations. Mirai variants are created daily (Kolias et al., 2017). The code review for this study does not consider variants. The code review is performed on the original Mirai code that was exposed publicly on the hacking community web forum Hackforums (J. A. Jerkins, 2017). Security researcher (Gamblin, 2016) has created a repository for the original Mirai code exposed on Hackforums. The repository created by Gamblin (2016) is used for the code review. Focusing on the experimentation of this study, which focuses on a Bot connecting to a potential new Bot Victim, the code review focuses on the mechanism of the Bot scanner for connecting to a potential new Bot Victim. The code review will not cover
other steps of the Bot scanner, such as (a) remotely accessing the new Bot Victim with a factory default user-id and password; (b) sending that logon information back the C&C for the upload of the malware onto the new Bot Victim (Kolias et al., 2017).

**Limitations.** Sample packet analysis and experimentation is performed with the dataset gathered by the University of Southern California (*Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016*). The dataset contains Bot scanning activity collection starting on June 1, 2016 and ending on March 31, 2017. Through 2016-11-20, Bot scanning traffic is sent to one IP address [130.152.184.2]. From 2016-11-20 onward Bot scanning traffic is added to an additional subnet [192.228.79.0/24]. The dataset is limited to SYN packets sent from the Bot to the potential new Bot Victim. The TCP SYN is the first step of the TCP handshake for a Bot establishing a connection to a potential New Bot Victim. Additional research questions could have been formulated and answered if the dataset contained the network packets for the complete tasks of the Bot scanner. The Bot scanner tasks include (a) the complete TCP handshake between the Bot and potential Bot Victim; (b) the Bot remotely accessing the new Bot Victim thru telnet with a factory default user-id and password; (c) the Bot communicating back to C&C with the logon information for malware upload onto the new Bot Victim. With the dataset containing SYN packets from the Bot scanner, the experimentation with the Mirai Bot Scanner Summation Prototype is focused on a Bot attempting to connect to a potential new Bot Victim. The next section provides a summary of the research methodology.

**Chapter Summary**

Chapter 3 discusses the (a) research method justification; (b) rationale for the research approach; (c) research data sources; (d) data analysis methods; (e) limitations and delimitations. The research method justified is a single case mechanism for the Mirai Bot Scanning Summation Prototype. The DSR research process is discussed for the required artificial artifacts including the algorithms and Python programs. The data analysis methods include (a) code review; (b) sample packet analysis; (c) experimentation with the recorded Bot scanner dataset. This study is delimited by the original Mirai code and experimentation is limited to the recorded Bot scanning dataset. The next chapter presents the experimentation for this study.
CHAPTER 4

EXPERIMENTATION

Chapter 3 presented the methodology for this study which included (a) the research method justification; (b) the rationale for the research approach; (c) research data sources; (d) data analysis methods; (e) limitations and delimitations. The research justification is for a single case mechanism for experimentation with the Mirai Bot Scanner Summation Prototype. The rational for the research approach details the DSR methodology. The research data sources for this study presented the Bot scanner code and a recorded Bot scanner network dataset. The data analysis methods included (a) a code review of the Bot scanner code; (b) sample packet analysis of the Bot scanner network dataset; (c) the Mirai Bot Scanner Summation Prototype experimentation with the recorded Bot scanner dataset. The Bot scanner code review focuses on the modules for the Bot scanning for a potential new Bot Victim. A sample PCAP file from the Bot scanning dataset is analyzed to characterize the network packets. The Mirai Bot Scanner Summation Prototype solution performs the experimentation with the Bot scanning dataset.

Experimentation is performed to help answer the research questions. The research questions are answered based upon the findings from experimentation performed from the (a) Bot scan code review; (b) sample packet analysis; (c) Mirai Bot Scanner Summation Prototype. The Bot scan code review focuses on the Bot scanner modules to delineate the functionality associated with a Bot scanning for a potential new Bot Victim. A sample PCAP file from the Bot scanning dataset is analyzed to characterize network packets for a Bot attempting to remotely connect to a new Potential Victim. The Mirai Bot Scanner Summation Prototype contains a summation module that tabulates the Bot scanning mechanism and stores the summations in a persistent database. An assessment module provides reports and line graphs from the persistent database. The reports and graphs, based upon a supplied date range of the capture Bot scanning dataset, assess the Bot scanning mechanism over time.

Chapter 4 contains the following sections (a) a code review of the Bot scanner; (b) network packet analysis from a sample PCAP file of the Bot scanning dataset; (c) Mirai Bot
Bot Scanner Code Review

Described as the first step in communication and operation, the Bot brute forces a potential new Bot Victim (Kolias et al., 2017). The Bot brute forcing consists of (a) a Bot connecting to a potential new Bot Victim; (b) the Bot remotely accessing the new Bot Victim with a factory default user-id and password. After the Bot establishes a connection, a telnet handshake occurs (Sinanović & Mrdovic, 2017). A list of 62 factory default user-id and password combinations are used to gain remote access to the new Bot Victim with telnet (Sinanović & Mrdovic, 2017). Since the focus of this study is a Bot connecting to a potential new Bot Victim, the code review will focus on the Bot scanner code. The modules providing the functionality for a Bot connecting to a potential new Bot Victim are reviewed. This section contains the following sub-sections for the code review: (a) a review of the Bot scanner initialization module which contains the functionality for creating the IP and TCP headers for connecting to a new potential Bot Victim; (b) a review of the module for creating a random IP address that the Bot scanner utilizes for the destination IP address for the potential new Bot Victim; (c) a summary provides the important findings from the above mentioned sections for the Bot scanner code review. The next sub-section provides a code review for the Bot scanner initialization.

**Scanner initialization.** Focusing on a Bot connecting to a potential new Bot Victim, the code review concentrates on the mechanism of the Bot scanner for connecting to a potential new Bot Victim. The Bot scanner.c program contains the functionality to brute force a potential new Bot Victim (Gamblin, 2016). The Bot scanner initialization module is the main driver for the Bot scanner and contains the functionality for a Bot connecting to a potential new Bot Victim. Figure 13 reviews the functionality for assigning a random IP address for connecting to a potential new Bot Victim.
Figure 13. Assign random IP address to IP header.

Shown above in Figure 13, is the main while-loop. Beginning at line 211, The IP header [iph] is assigned a random id as well as a random IP address for the destination address [daddr]. The destination address represents the potential new Bot Victim IP address. Beginning at line 205, concerning the for-loop, SCANNER_RAW_PPS is defined to be 160, thus the scanner will generate 160 IP headers with random generated IP addresses. At line 201, fake_time ensures the synchronization of generating 160 IP headers and brute forcing potential new Bot Victims. Fake_time ensures time has progressed before generating the IP header. Fake_time and its calculation are out of scope and not reviewed for this study. Once the IP header has been assigned the random destination address, the TCP header destination is assigned. Figure 14 reviews the assignment of the port destination to the TCP header.
Figure 14. Assign TCP header destination.

Shown above in Figure 14, the TCP header [tcph] is initialized. Beginning at line 217, the TCP destination is assigned either 23 or 2323. Every tenth iteration of a packet header, the destination is assigned 2323 else the destination is assigned 23. Therefore, 9 out of 10 destinations will be assigned 23 and one destination will be assigned 2323. Telnet ports are associated with 23 and 2323 (Kolias et al., 2017). At line 225, the TCP header sequence is assigned the IP header destination address [tcph->seq = iph->daddr]. Starting on line 229, the socket address is initialized. The socket address [paddr] is set. The socket is a network interface that allows the Bot to communicate with TCP/IP over the internet (Koutsoubelias & Lalis, 2013). AF_INET refers specifically to IPv4 addresses (Li, Jin, Shao, & Liao, 2009). On line 230, the destination address of the packet header is assigned to the socket. Line 231 assigns the TCP header destination, which is the telnet port 23 or 2323. At line 233, the scanner raw packet is sent thru the [paddr] socket to remotely connect to the potential new Bot Victim on the telnet port (Donahoo & Calvert, 2009).

The TCP handshake exchanges SYN and ACK packets to establish connection between hosts (Arlitt & Williamson, 2005). Once the SYN packet is sent to connect to a potential new Bot Victim, the Bot waits for an ACK packet response from the potential new Bot victim. Figure 15 depicts the functionality for reading packets from the raw socket and verifying IP header.
Shown above in Figure 15, an infinite while loop receives the network packets to get the SYN and ACK responses from the potential new Bot Victim. At Line 248 the variable \( n \) is assigned the return code from receiving the data from the socket. In the following line 249, a break out of the while loop occurs if no data is received from the socket \( [n <= 0] \), timeout expired before data was received \( [errno == EAGAIN] \), or the receive operation would block communication on the socket \( [errno == EWOULDBLOCK] \) (Donahoo & Calvert, 2009). At line 252, the size of the data \( [dgram] \) received from the socket \( [n] \) is verified to at least contain the size of the IP header \( [iphdr] \) and the size of the TCP header \( [tcphdr] \). If the data received from the socket is not at least the size of the IP header plus the TCP header, then a continue to the beginning of the while loop is issued to receive more data from the socket.

Starting at line 254, IP header \( [iph] \) checks are performed. On line 254, if the IP header destination is not equal to the Bot address \( [iph->daddr != LOCAL_ADDR] \) then the code will continue to the beginning of the while-loop to receive another \( dgram \) from the socket. On line 256, if the IP protocol is not equal to TCP \( [iph->protocol != IPPROTO_TCP] \) the code will continue to the beginning of the while-loop to receive another \( dgram \) packet from the socket. Once the IP header has been verified, then the TCP header is verified. Figure 16 shows the functionality for verifying the TCP header.
Figure 16. Verify TCP header.

Shown above in Figure 16, various checks verify the TCP header \([tcph]\). The verification starts with line 258 verifying that the source is from telnet port 23 or 2323. If the source is not from a telnet port \([tcph->source != htons(23) && tcph->source != htons(2323)]\), then the code will continue to the beginning of the while-loop to read another dgram packet from the socket. On line 260, if the TCP header destination is not equal to the source port then the code will continue to the beginning of the while-loop to read another dgram packet from the socket. Line 260 represents a check for verifying that the telnet port, 23 or 2323, matches between the TCP header and the Bot source for network communication. The code will continue to the beginning of the while-loop to read another dgram packet from the socket.

Starting at line 262, if the \(syn\) flag is not set \([!tcph->syn]\) and the \(ack\) flag is not set \([!tcph->ack]\) in the TCP header, the code will continue to the beginning of the while-loop to read another dgram packet from the socket. The \(syn\) and \(ack\) TCP header flags verify a connection was made with the TCP handshake (Arlitt & Williamson, 2005). The socket is establishing the network connection between the Bot and a potential new Bot Victim with the SYN and ACK packets. On line 266, if the TCP header restart flag is encountered \([tcph->rst]\) the code will continue to the beginning of the while-loop to read another dgram packet from the socket. A TCP restart flag is sent when the target is not able to process the SYN packet (Arlitt & Williamson, 2005). A TCP restart flag indicates that the Bot is unable to connect to the potential new Bot Victim. At line 266, if the TCP header \(fin\) flag is set \([tcph->fin]\) the code will continue to the beginning of the while loop to read another dgram packet from the
socket. The fin flag indicates the target closing the TCP connection (Arlitt & Williamson, 2005). The potential new Bot Victim sets the fin flag to close the socket connection with the Bot.

Line 270 performs a check for the TCP header ACK Sequence \[ntohl(tcph->ack_seq) - 1) != iph->saddr\]. The check at line 270 ensures the correct sequence for exchanging SYN and ACK TCP packets (Arlitt & Williamson, 2005). The check at line 270 verifies a SYN packet was sent by the Bot and an ACK network packet was received from a potential new Bot Victim. If the TCP header ACK sequence minus one is not equal to the IP header source address, then the code will continue to the beginning of the while-loop to read another dgram packet from the socket. Once the checks have been performed for the IP header and TCP header, the TCP handshake has verified it is possible for the Bot to connect to a potential new Bot Victim. (Arlitt & Williamson, 2005). Figure 17 presents the code for establishing the connection table parameters and calling the setup_connection function with the connection parameters.

![Code snippet](image)

**Figure 17.** Connection table parameters.

Shown above in Figure 17, the connection parameters are established. At line 278, the connection is added to the connection table \[conn = &conn_table[n]\]. At line 288, the connection destination address is assigned from the IP header \[conn->dst_addr = iph->saddr\]; the IP address of a potential new Bot Victim is assigned in the connection table. At
line 289, the destination port is assigned to the connection table from the TCP header source \[conn->dst_port = tcph->source\]. Once the connection table parameters are set, at line 290, the \textit{setup_connection} function is called providing the connection as a parameter \[setup_connection(conn)\]. Figure 18 reviews the \textit{setup_connection} function.

```c
646 static void setup_connection(struct scanner_connection *conn) {
647   struct sockaddr_in addr = {{0}};
648   if (conn->fd != -1)
649     close(conn->fd);
650   if ((conn->fd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
651     {
652       #ifdef DEBUG
653         printf("[scanner] Failed to call socket()\n");
654       #endif
655       return;
656     }
657     conn->rdbuf_pos = 0;
658     util_zero(conn->rdbuf, sizeof(conn->rdbuf));
659    fcntl(conn->fd, F_SETFL, O_NONBLOCK | fcntl(conn->fd, F_GETFL, 0));
660     addr.sin_family = AF_INET;
661     addr.sin_addr.s_addr = conn->dst_addr;
662     addr.sin_port = conn->dst_port;
663     conn->last_recv = fake_time;
664     conn->state = SC_CONNECTING;
665     connect(conn->fd, (struct sockaddr *)&addr, sizeof(struct sockaddr_in));
666 }
```

\textit{Figure 18.} Setup connection module.

Shown above in Figure 18, the Bot establishes a connection to a potential new Bot Victim. At line 652, a handle is assigned to the socket. Starting at line 665, the \textit{socketaddr} variable \textit{addr} is initialized. At line 665, the socket family is set to IPv4 \[addr.sin_family = AF_INET\]. At line 666, the source address is set to the destination IP of the potential new Bot Victim \[addr.sin_addr.s_addr = conn->dst_addr\]. At line 667, the telnet port from the TCP handshake between the Bot and the potential new Bot Victim is assigned \[addr.sin_port = conn->dst_port\]. At line 669, the last time the Bot connected to the potential new Bot Victim is set \[conn->last_recv = fake_time\]. At line 670, the connection state is set to connecting \[conn->state = SC_CONNECTING\]. At line 671, The Bot makes a connection thru the socket to the potential new Bot Victim.

The focus of this study is a Bot connecting to a potential new Bot Victim. Therefore, the remaining Bot code for remotely accessing a potential new Bot Victim with a factory
default user-id and password is not reviewed. Shown below in Figure 19, a random IP address is generated for the potential new Bot Victim. The next sub-section reviews the code for generating the random IP address for identifying a potential new Bot Victim.

**Random IP address.** Shown in Figure 19, a random IP address is generated for identifying a potential new Bot Victim. Discussed by Kolias et al. (2017), Mirai avoids propagating to certain IP addresses in order to avoid detection. The black-list contains Loopback, Invalid address space, General Electric Company, Hewlett Packard Company, US Postal Service, Internal network, Internet Assigned Numbers Authority (IANA) Network Address Translation (NAT) reserved, IANA special use, Multicast, and US Department of Defense. Figure 19 presents the `get_random_ip` function which generates the random IP address.

```c
static ipv4_t get_random_ip(void)
{
    uint32_t tmp;
    uint8_t o1, o2, o3, o4;
    do
    {
        tmp = rand_next();
        o1 = tmp & 0xff;
        o2 = (tmp >> 8) & 0xff;
        o3 = (tmp >> 16) & 0xff;
        o4 = (tmp >> 24) & 0xff;
    }
    while (o1 == 127 ||
           (o1 == 0) ||
           (o1 == 3) ||
           (o1 == 15) ||
           (o1 == 56) ||
           (o1 == 10) ||
           (o1 == 192 & o2 == 168) ||
           (o1 == 172 & o2 >= 16 & o2 < 32) ||
           (o1 == 198 & o2 >= 18 & o2 < 20) ||
           (o1 > 224) ||
           (o1 == 6 || o1 == 7 || o1 == 11 || o1 == 21
            || o1 == 22 || o1 == 26 || o1 == 28 ||
            || o1 == 29 || o1 == 30 || o1 == 33 || o1 == 55
            || o1 == 214 || o1 == 215)) // Department of Defense
    );

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

**Figure 19.** Get random IP address.

Shown above in Figure 19, the `get_random_ip` generates the random IP address for identifying and connecting to a potential new Bot Victim. At line 678, a `do-while` loop is implemented. Inside the `do-while` loop, beginning at line 680 and ending at line 685, the IP
address octets are randomly generated. Starting at line 687, based upon the randomly generated octets the while condition black-lists several IP subnets. Corporation subnets are black-listed, such as General Electric and Hewlett-Packard, as well as US government subnets, such as the US Postal Service and the Department of Defense. The next sub-section provides a summary of the code review performed in this study.

**Bot scanner code review summary.** Described as the first step in communication and operation, the Bot brute forces a potential new Bot Victim (Kolias et al., 2017). The Bot brute forcing consists of (a) a Bot connecting to a potential new Bot Victim; (b) the Bot remotely accessing the new Bot Victim with a factory default user-id and password. After the Bot establishes a connection, a telnet handshake occurs (Sinanović & Mrdovic, 2017). The focus of this study is a Bot connecting to a potential new Bot Victim. Focusing on a Bot connecting to a potential new Bot Victim, the code review concentrates on the mechanism of the Bot scanner for connecting to a potential new Bot Victim.

The Bot `scanner.c` program contains the functionality to brute force a potential new Bot Victim (Gamblin, 2016). The Bot scanner initialization module is the main driver for the Bot scanner and contains the functionality for a Bot connecting to a potential new Bot Victim. The initialization module generates a random IP address for the potential new Bot Victim. Corporations, IANA, and US government entities are black-listed from the random IP address. A socket connection is attempted with the random IP address and the telnet port. A connection between the Bot and the potential new Bot Victim is attempted with the TCP handshake. When the TCP handshake has been established between the Bot and the potential new Bot Victim, then the Bot connects to the potential new Bot Victim. The connection is made thru the socket to the random generated IP address of potential new Bot Victim on the telnet port. The next section provides packet analysis of the Mirai dataset that only contains network traffic representing Bots initiating the TCP handshake with potential new Bot Victims (*Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016*).

**Bot Scanning Sample Network Packet Analysis**

The previous section provided a code review of the Bot scanner. The code review concentrates on the mechanism of the Bot scanner for connecting to a potential new Bot
Victim. The Bot scanner initialization module generates a random IP address for the potential new Bot Victim with corporations, IANA, and US government entities being black-listed. A socket connection is attempted with the random IP address and the telnet port. The socket connection is established with the TCP handshake being performed between the Bot and the potential new Bot Victim. When the TCP handshake has been established between the Bot and the potential new Bot Victim, then the Bot connects to the potential new Bot Victim back thru the socket. The connection is to the random generated IP address of potential new Bot Victim on the telnet port. The code review supports the experimentation performed with the Bot scanning dataset.

This section analyzes a sample PCAP file from the Bot scanning dataset. The sample is indicative of the PCAP files gathered for the dataset. The Bot scanning dataset consists of daily PCAP files gathered starting on June 6, 2016 and ending March 31, 2017. The dataset includes TCP SYNs sent on ports 23 and 2323. (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). This next section discusses the Mirai dataset network packets captured consisting of (a) a discussion of the sample PCAP file; (b) a discussion of SYN packets for when the Bot is attempting to establish a TCP handshake with a potential new Bot Victim; (c) a discussion of retransmission packets indicating the potential new Bot Victim not willing to perform the TCP handshake with the Bot; (d) a summary of the packet analysis.

**Sample PCAP file.** The PCAP file analyzed for this section is for the Bot scanning traffic recorded on 2016-11-20. The PCAP file contains SYN and retransmission packets. Figure 20 displays the PCAP file analyzed with Wireshark.
Figure 20. 2016-11-20.pcap file analysis.

Shown above in Figure 20, the sample PCAP file for 2016-11-20, analyzed in Wireshark, shows SYN packets and retransmission packets. A total of 93463 packets are captured in this PCAP file. There are 61,343 SYN packets. Wireshark has identified 32029 retransmission packets. The IP source varies in the network packets. The IP destination of the packets vary; 130.152.184.2 is shown as the destination but other packets in the PCAP file contain different IP addresses. Only the TCP protocol is captured. The length of the packets is either 60 or 66. The info column shows either telnet port 23 or 2323 is utilized for the SYN or retransmission packets. This information provided supports the code review performed by this study for packet size and telnet ports. The code review identifies the (a) SYN packets being sent from the Bot to the potential new Bot Victim; (b) the Bot may receive a packet with the TCP header restart flag set which would cause a retransmission packet; (c) telnet ports 23 and 2323 are used when the Bot attempts to establish the TCP handshake with the potential new Bot Victim. The next sub-section analyzes the SYN packet.

**SYN packet.** The SYN packet initiates the TCP handshake for network connections (Arlitt & Williamson, 2005). The code review showed that the Bot scanner creates a socket that connects to a potential new Bot Victim. The socket connection initiates a SYN packet sent from the Bot to the potential new Bot Victim. Figure 21 shows the details of the SYN packet.
Shown above in Figure 21, the details of the (a) Internet Protocol Version 4 and (b) Transmission Control Protocol are presented for the SYN packet. The IP packet header contains the Source [89.212.62.62] and Destination [130.152.184.2] IP addresses. The Bot IP is the Source IP and the potential new Bot Victim is the Destination IP. The IP header protocol is TCP. The TCP header contains the Source Port [58920], the telnet Destination Port [23], and the SYN Flag [0x002]. The Source Port on the Bot is available for the Bot to use for initiating the SYN packet. The Bot is trying to connect to the telnet Destination Port to determine if the potential new Bot Victim has that telnet port open. The values of the variable and flags of the IP header and TCP header support the analysis performed by the code review. The next sub-section analyzes the retransmission packet.

Retransmission packet. When a Bot is attempting to establish a connection to a potential new Bot Victim with the TCP handshake, the potential new Bot Victim may not respond to the SYN with an ACK packet or may send back a restart flag in the TCP header (Arlitt & Williamson, 2005). If the Bot does not receive an ACK back from the potential new Bot Victim or the potential new Bot Victim sends back the TCP header restart flag, the Bot sends another SYN packet. Figure 22 shows the details of the retransmitted SYN packet.
Shown above in Figure 22, the TCP header details are presented for a retransmission packet. The Source Port, Destination Port, Sequence Number, and Flags are the same as the original SYN packet. Wireshark suspects this packet as a retransmission since the Retransmission Time-Out (RTO) for this segment was approximately 68 seconds. The retransmission packet indicates a potential new Bot Victim not responding to the Bot request for establishing a TCP handshake. The next sub-section provides a summary for the sample packet analysis.

**Bot scanning network packet analysis summary.** The PCAP file analyzed for this section is for the Bot scanning traffic recorded on 2016-11-20. The PCAP file contains SYN and retransmission packets. A total of 93463 packets are captured in this PCAP file. There are 61,343 SYN packets and 32029 retransmission packets. The IP source varies. The IP destination of the packets vary. Only the TCP protocol is captured. Either telnet port 23 or 2323 is utilized for the SYN or retransmission packets. The IP packet header contains the Source and Destination IP addresses. The Bot IP is the Source IP and the potential new Bot Victim is the Destination IP. The TCP header contains the Source Port, the telnet Destination Port, and the SYN Flag. The Source Port on the Bot is available for the Bot to use for
initiating the SYN packet. The Bot is trying to connect to the telnet Destination Port to determine if the potential new Bot Victim has that telnet port open. If the Bot does not receive an ACK back from the potential new Bot Victim or the potential new Bot Victim sends back the TCP header restart flag, the Bot retransmits the SYN packet in an attempt to establish the TCP handshake. The Source Port, Destination Port, Sequence Number, and Flags for the TCP header in the retransmission packet are the same as the original SYN packet. The values of the variables and flags of the IP header and TCP header support the analysis performed by the code review. The next sub-section presents a code review of the Mirai Bot Scanner Summation Prototype.

**Mirai Bot Scanner Summation Prototype Code Review**

The previous section analyzed a sample PCAP file from the Bot scanning traffic recorded on 2016-11-20. The PCAP file contains SYN and retransmission packets. A total of 93463 packets are captured in this PCAP file consisting of 61,343 SYN packets and 32029 retransmission packets. The SYN packet represents a Bot attempting to establish a connection to the potential new Bot Victim with a TCP handshake over a telnet port. As a reaction to the potential new Bot Victim not responding back with an ACK packet, the retransmission packet signifies a Bot re-attempting a connection to the potential new Bot Victim by sending another SYN packet. With the Bot scanning dataset consisting of daily PCAP files gathered starting on June 6, 2016 and ending March 31, 2017, this study presents the Mirai Bot Scanner Summation Prototype for experimentation with the Bot scanner dataset consisting of TCP SYN and retransmission packets.

This section presents the code review for the Mirai Bot Scanner Summation Prototype. The *summation* and *assessment* component Python artifacts will be reviewed. The *summation* process is responsible for tabulating the Bot scanning mechanism for each PCAP file of the Bot scanning dataset. The *assessment* component provides reports and graphs that allow for the monitoring of the Bot scanning mechanism over time. The *summation* and *assessment* component interfaces with the persistent database. The *summation* process stores the tabulated results within the database. The *assessment* process retrieves the records from the database to build the reports and line graphs. This section provides (a) a code review for
the summation; (b) a code review for the assessment; (c) a summary of the Mirai Bot Scanner Summation Prototype. This next section presents the code review for the summation process.

Summation. The Bot scanning dataset consists of 304 daily PCAP files gathered starting on June 6, 2016 and ending March 31, 2017. Each PCAP file contains only Bot scanning data consisting of TCP SYNs and TCP retransmission packets sent on telnet ports 23 and 2323. (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). The summation component a Python script that analyzes and tabulates each PCAP (Infosecchazzy, 2019). The Python script, BotScanner.py, contains the functions necessary to summate the Bot scanning mechanism contained in each PCAP file of the Bot scanning dataset. Each PCAP file is read and are tabulated and saved in a MongoDB database. The BotScanner.py function analyze_pcap_file reads a PCAP file and saves the summation into the MongoDB database. Figure 23 shows the functionality of analyze_pcap_file for reading the PCAP file and gathering all the source and destination IP addresses.

```python
## Read the PCAP file
packets = rdpcap(pcap_file)

## Go thru each packet
for each_packet in packets:
    ## increment total packets read
    total_packets = total_packets + 1

    ## Build the needed lists
    ip_src_list.append(each_packet["IP"].src)
    ip_dst_list.append(each_packet["IP"].dst)

    ## assign destination ip
    dest_ip = each_packet["IP"].dst

    ## Get the destination subnets in the PCAP file
    dst_subnet = dst_subnet(dest_ip)

    ## put subnet in list
    if dst_subnet not in subnet_list:
        subnet_list.append(dst_subnet)
```

*Figure 23. Gathering the packet source and destination IPs.*
Shown above in Figure 23, the function `analyze_pcap_file` function reads the PCAP file and stores the source and destination IP in a list. Also, the destination subnet is stored in a list. At line 110, the Scapy `rdpcap` routine is called to read the PCAP file (Montante, 2018). At line 113, a `for-loop` is initiated to read each packet `[each_packet in packets]`. At line 116, `total_packets` is incremented as a counter for calculating the total number of packets in the PCAP file. Line 119 adds the source IP to the `source IP list [ip_src_list]`. At line 120, the destination is added to the `destination IP list [ip_dst_list]`. Lines 123 to 130 represent the functionality for putting the `destination subnet into the subnet list [subnet_list]`. The source and destination IP address of the network packet represents the identification of a Bot scanning for a potential new Bot Victim. Once all the packets have been read the `for-loop` ends. All the source and destination IP addresses have been discovered as well as the destination subnets. With the source and destination IP addresses discovered, the number of Bots and potential new Bot Victims can be calculated along with the number TCP SYN and retransmission packets. Shown below, Figure 24 contains the logic for determining the number of Bots, potential new Bot Victims, SYN packets, and retransmission packets.

```
packets_dict = {'Source':ip_src_list, 'Destination':ip_dst_list}

packets_df = pd.DataFrame(packets_dict)

total_bots = len(packets_df['Source'].unique())

criteria = packets_df1.loc[packets_df1['Freq'] == 1]

total_syn_packets = criteria['Freq'].sum()

total_syn_packets = 0

df_pot_new_bot_victims = criteria['Destination'].unique()

total_potential_new_bot_victims = len(df_pot_new_bot_victims)

total_retransmission_packets = total_packets - total_syn_packets

packet_date = pcap_file_date(pcap_file)
```

*Figure 24. Determining the Bots and potential new Bot Victims.*
Shown above in Figure 24, the source and destination lists are analyzed to determine the Bots, potential new Bot Victims, SYN packets, and retransmission packets. At line 137, a dictionary `packets_dict` is constructed containing the source and destination for each packet. On line 140, a Pandas `DataFrame` `packets_df` is created from the dictionary of source and destination IP addresses. The DataFrame provides methods for adding data, deleting data, indexing data, and querying data ("Pandas: powerful Python data analysis toolkit," 2018). Line 143 calculates the total number of bots `[total_bots]` from the unique source IPs contained in the DataFrame. On line 146, the frequency for the source and destination IP is calculated `[packets_df1]`. Line 149 gathers all the source and destination IP combos that only occur once `[Freq == 1]`. Lines 151 to 155 determine the number of SYN packets `[total_syn_packets]` based upon the number of packets with a source and destination IP with a frequency of one. Line 158 and 159 calculate the number of potential new Bot Victims based upon the number of unique destination IP from the SYN packets `[total_potential_new_bot_victims]`. At line 162, the total number of retransmission packets is calculated by subtracting the total SYN packets from the total packets contained in the PCAP file `[total_retransmission_packets]`. The total number of packets will be comprised of SYN packets and retransmission packets. At line 165, the packet date `[packet_date]` is determined by calling the function `pcap_file_date` with the `pcap_file` parameter. The number of Bots, potential new Bot Victims, SYN packets, and retransmission packets along with the packet date have been determined for the PCAP file. The totals for the PCAP file are stored in a MongoDB database. Figure 25 contains shows the PCAP file totals being inserted into the MongoDB database.
Shown above in Figure 25, the totals are inserted into the *Daily_PCAP* table. The *Daily_PCAP* table will contain the totals for all the Bot Scanning PCAP files. Once the PCAP totals have been inserted into the *Daily_PCAP* table, run-time processing information is inserted into another table. Figure 26 shows the run-time information inserted into the MongoDB table.

*Figure 26. PCAP processing time.*
Shown above in Figure 26, the starting and ending date for processing the PCAP will be inserted into the Daily_PCAP_Runtime table. The starting and ending date will contain the date along with the hour, minutes, and second. The information in the Daily_PCAP_Runtime table is used to analyze the processing time of the packets for the PCAP file. To process all the PCAP files, the utility Analyze_PCAP_Files.py is included (Infosecchazzy, 2019). Shown in Figure 27, the analyze_pcaps function enumerates the PCAP files provides each PCAP file on as a parameter to the summation function bs.analyze_pcap_file.

```python
54  def analyze_pcaps(pcap_dir):
55      # Directory separator
56      Slash = '/'
57
58      # Analyze the PCAP files
59      for root, dirs, files in os.walk(pcap_dir):
60          # for each file in the directory
61              for filename in files:
62                  # calculate the pcap file including its path
63                  pcap_file = pcap_dir + Slash + filename + Slash + filename
64                  # analyze the pcap file
65                  bs.analyze_pcap_file(pcap_file)
66
67      # Main
68      #
69      if __name__ == "__main__":
70          # Analyze the 2016 PCAP files
71          analyze_pcaps(PCAP_DIR_2016)
72          # Analyze the 2017 PCAP files
73          analyze_pcaps(PCAP_DIR_2017)
```

Figure 27. Process PCAP files.

Shown above in Figure 27, starting at line 79 the directories containing the 2016 and 2017 PCAPS are supplied as parameters to analyze_pcaps. The analyze_pcaps function determines the PCAP filenames in the directory and then calls bs.analyze_pcap_file to summate each PCAP within the directory. Executing Analyze_Pcap_Files.py will sequentially analyze the PCAP files from the Bot scanning dataset. Once all the PCAP files have been analyzed with the summation being stored in the Daily_PCAP table, then the assessment module can assess the summation by interfacing with the Daily_PCAP table. The following sub-section presents the code review for the assessment process.
Assessment. The previous sub-section presented the summation process. This sub-
section provides a code review for the assessment component. The BotScannerResults.py
Python program contains the reporting and graphical functions for assessing the summation
contained within the MongoDB database (Infosecchazzy, 2019). The details and summaries
are provided for the PCAP files in formatted text. Shown below, Table 7 provides a listing of
the functions providing the details and summaries.

Table 7
Detail and Summary Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pcapruntime_details</td>
<td>This function prints out the details for the PCAP runtime</td>
</tr>
<tr>
<td>pcapruntime_summary</td>
<td>This function summarizes the PCAP runtime</td>
</tr>
<tr>
<td>dataset_details</td>
<td>This function prints out the details for the dataset</td>
</tr>
<tr>
<td>dataset_summary</td>
<td>This function summarizes the dataset</td>
</tr>
</tbody>
</table>

Shown above in Table 7, there are four functions which provide formatted text output.
Pcapruntime_details provides the details for the PCAP runtime. Pcapruntime_summary
summarizes the PCAP runtime. Dataset_details provides the details for the PCAP files
contained in the Bot scanning dataset. Dataset_summary summarizes the PCAP files. Each
function contains two parameters, start_date and end_date. The start_date and end_date
parameters allow for the flexibility to include a range of PCAP files for assessment. Figure
28 presents the dataset_details function.
Shown above in Figure 28, the details are provided for the PCAP file. At line 312, the \textit{Daily\_PCAP} table is searched for finding all records between the \textit{start date} and \textit{end date}. All PCAP files between the start and end date will be selected. At line 324, for each PCAP file, the totals will be displayed on the screen \texttt{[for p\_file in all\_pfiles]}. Each field from the \textit{Daily\_PCAP} table is presented. The code for the remaining functions mentioned in Table 1. Detail and summary functions will not be reviewed. The code for the detail and summary Python functions are similar in functionality (Infosecchazzy, 2019). Besides providing detail and summary reporting functions, three graphical functions are provided. The Bokeh library is utilized for providing the graphing capabilities for the solution \texttt{("Welcome to Bokeh," 2015). Table 8 provides a listing of the functions providing graphical analysis.
Table 8

*Graphical Functions*

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bot_totals_graph</td>
<td>This function produces two line graphs</td>
</tr>
<tr>
<td></td>
<td>Graph 1: Number of Bots</td>
</tr>
<tr>
<td></td>
<td>Graph 2: Number of potential new Bot Victims</td>
</tr>
<tr>
<td>bot_average_graph</td>
<td>This function produces two line graphs</td>
</tr>
<tr>
<td></td>
<td>Graph 1: Average Number of Bots</td>
</tr>
<tr>
<td></td>
<td>Graph 2: Average Number of potential new Bot Victims</td>
</tr>
<tr>
<td>packet_total_graph</td>
<td>This function produces a three-line graph</td>
</tr>
<tr>
<td></td>
<td>Line 1: Total Packets</td>
</tr>
<tr>
<td></td>
<td>Line 2: Total Syn Packets</td>
</tr>
<tr>
<td></td>
<td>Line 3: Total Retransmission Packets</td>
</tr>
</tbody>
</table>

Shown above in Table 8, there are three functions which provide graphical analysis. *Bot_totals_graph* provides two line graphs. The first line graph contains the *Bot totals*. The second graph contains the *potential new Bot Victim totals*. *Bot_average_graph* provides two line graphs. The first graph provides the *average number of Bots scanning per second*. The second graph provides the *average number of potential new Bot Victims per minute*. The *packet_total_graph* function provides a three-line graph for *total packets*, *total SYN packets*, and *total retransmission packets*. Each function contains two parameters, *start_date* and *end_date*. The *start_date* and *end_date* allow for the flexibility to include a range of PCAP files. Figure 29 presents the *bot_totals_graph* function.
Shown above in Figure 29, the `bot_totals_graph` function computes PCAP totals from a starting date and ending date. At line 361, the daily PCAP information is found for PCAPS in the search range of the starting date and end date `[all_pfiles]`. Starting at line 367, for each PCAP file included in the search (a) the packet date is appended to the `pcaps` list `[pcaps]`; (b) the total number of bots is appended to the `bots` list `[bots]`; (c) the total number of potential new bot victims are appended to the potential new bot victims list `[potential_new_bot_victims]`. When all the PCAP information is gathered and the lists are built, then the graphs are created. Figure 30 presents the code for creating the line graph for the Bot Totals.

```
## find the records in the db from start_date until the end_date
all_pfiles = db.Daily_PCAP.find("packet_date": {"$gte": start_date, "$lte": end_date})

## Close client
client.close()

## loop thru all of the records
for p_file in all_pfiles:
    ## PCAP files
    pcaps.append( dt.datetime.strptime( str(p_file['packet_date'] ), "%Y-%m-%d") )

    ## Bots
    bots.append(int(p_file['total_bots']))

    ## potential new Bot victims
    potential_new_bot_victims.append(int(p_file['total_potential_new_bot_victims']))

## output to static HTML file
output_file("bot_totals_graph.html")

## define the title and x-axis
p1 = figure(title="Bot Totals", x_axis_type='datetime')

## Bots Line
p1.line(pcaps, bots, legend="Bots", line_width=5, line_color="red")

## show the line
show(p1)
```

**Figure 29.** Bot totals graph function.

**Figure 30.** Creating the line graph for the number of Bots.
Shown above in Figure 30 is the code that produces a line graph for the *Bot Totals* for the date range from the *start_date* and *end_date* parameters. Line 379 defines the Hyper Text Markup Language (HTML) file that will open in a web-browser [output_file]. At line 382, the title and x-axis type are set for the graph [p1]. At line 385, the Bokeh *line* function is called providing the *pcaps* as the x-axis values and the *bots* as the y-axis values. A legend is provided as well as the line width of five and the line color as red. At Line 388, the *show* function displays the graph in the browser containing the red Bots line over the time period provided in the *pcaps* list. Once the Bots line graph has been shown, then the *potential new Bot Victims* line graph is produced. Figure 31 presents the code for producing the potential new Bot Victim line graph.

```python
390 # Assign HTML file
391 # output to static HTML file
392 output_file("pot_bot_totals_graph.html")
393
394 # define the title and x-axis
395 p2 = figure(title="Potential New Bot Victim Totals", x_axis_type='datetime')
396
397 # potential new Bot Victims Line
398 p2.line(pcaps, pot_new_bot_victims, legend="Pot New Bot Victims", line_width=5, line_color="yellow")
399
400 # show the line
401 show(p2)
```

*Figure 31. Creating the line graph for number of potential new Bot Victims.*

Shown above in Figure 31, the code for creating the line graph for the number of *potential Bot Victims* is similar to the code for creating the line graph for the number of Bots. At line 392, the HTML output file is defined [output_file]. At line 392, the title and x-axis type are defined for the graph [p2]. On line 398, *pcaps* list is provided for the x-axis and the *pot_new_bot_victims* list is provided for the y-axis. Line 401 shows the potential Bot Victims line graph in a different browser session than the Bots line graph. The *bot_totals_graph* function provides a graphical representation for *Bots* and *potential new Bot Victims* over a time-period. The code for the remaining functions mentioned in Table 2. Graphical functions will not be reviewed. The code for all the graphical functions is similar to the code for *bot_totals_graph* (Infosecchazzy, 2019). The next sub-section provides a summary.
Prototype code review summary. The Mirai Bot Scanner Summation Prototype includes the following Python programs (a) BotScanner.py; (b) Analyze_PCAP_Files.py; (c) BotScannerResults.py. The BotScanner.py program contains the functions for summation of a PCAP file from the Bot scanning dataset (Infosecchazzy, 2019). The summation process is able to analyze each network packet of the Bot scanner dataset to identify a Bot scanning for a potential new Bot Victim. The summation for each PCAP file includes the totals for: Bots, potential new Bot Victims, unique SYN packets, and retransmission packets. The summation totals are stored in a MongoDB database. Analyze_pcap_files.py contains the function for listing all the PCAP files of the Bot scanning dataset (Infosecchazzy, 2019). Each PCAP file is supplied as a parameter to the PCAP summation function analyze_pcap_file of BotScanner.py. BotScannerResults.py provides the assessment functions for reporting along with corresponding line graphs (Infosecchazzy, 2019). The next section focuses on single case experimentation performed with the Mirai Bot Scanner Summation Prototype.

Single Case Experimentation

The previous section presented a code review for the Mirai Bot Scanner Summation Prototype. This section performs experimentation with the Mirai Bot Scanner Summation Prototype for answering the research questions. This section presents (a) a discussion of the persistent database; (b) a discussion of assessment automation; (c) a discussion concerning reporting assessment; (d) a discussion concerning line graphing assessment; (e) a summary for the single-case experiment. The next sub-section discusses the persistent database.

Persistent database. The summation process tabulates the Bot scanning mechanism for each PCAP file. The summation results are stored in the MongoDB database. The assessment process searches the MongoDB to produce reports and line graphs. Performing the experimentation with the summation process to summate all of the PCAP files of the Bot scanner dataset, Figure 32 presents the contents of the MongoDB database.
Shown above in Figure 32, the MongoDB Daily_PCAP contains 304 records. Each record corresponds to a summated PCAP file of the Bot scanner dataset. Each record contains the packet_date, total_potential_new_bot_victims, total_retransmission_packets, total_packets, total_syn_packets, dest_subnet, and total_bots. The Daily_PCAP_Runtime table contains the start and ending time for the summation of each PCAP file. The summation was successful storing the summated Bot scanning mechanism in the persistent MongoDB database. The next sub-section discusses assessment automation.

**Assessment automation.** The Python program Answer_Research_Questions.py defines two functions for answering the research questions (Infosecchazzy, 2019). In the main portion of the Answer_Research_Questions.py program, the two functions are called for automating the assessment. Figure 33 presents the two functions for answering the research questions.
Shown above in Figure 33, two functions answer the research questions. Starting at line 35, the function `can_the_bots_and_potential_new_bot_victims_be_identified` is defined. The function `can_the_bots_and_potential_new_bot_victims_be_identified` answers the first two research questions for identifying and summating Bots as well as potential new Bot Victims. Two `BotScanner.py` functions are called. Both functions include the starting and ending date of the Bot scanning dataset as parameters. At line 38, the `bsr.dataset_details` function provides the details from the `Daily_PCAP` table. At line 41, the `bsr.dataset_summary` function provides the summary of the records from the `Daily_PCAP` table.

Starting at line 48, the `is_it_possible_to_monitor_bot_scanning_over_time` function is defined. The `is_it_possible_to_monitor_bot_scanning_over_time` function performs assessment to answers the third research question concerning monitoring the Bot scanning mechanism over time. Three `BotScannerResults.py` functions are called. The starting date, 2016-12-20, and the ending date, 2016-21-24, is supplied as parameters. The Bot scanning assessment is performed starting on 2016-12-20 and ending on 2016-12-24. At line 51, the function `bsr.dataset_summary` provides the summary of the records from the date range. At line 54, the

---

```python
# First and second Research Questions

# def can_the_bots_and_potential_new_bot_victims_be_identified():
#     # details for each PCAP in the dataset
#     bsr.dataset_details("2016-06-01", "2017-03-31")
#     # summary for all of the PCAP files
#     bsr.dataset_summary("2016-06-01", "2017-03-31")

# Third research question

def is_it_possible_to_monitor_bot_scanning_over_time():
    # summary for all of the PCAP files
    bsr.dataset_summary("2016-12-20", "2016-12-24")
    # packet totls graph
    bsr.packet_total_graph("2016-12-20", "2016-12-24")
    # Bot and potential new Bot Victim graph
    bsr.bot_totals_graph("2016-12-20", "2016-12-24")
```
bsr.packet_total_graph function is called to graph the packet totals. At line 58, the
bsr.bot_total_graph function is called to graph the Bot and potential new Bot Victims.

The main part of the Python program Answer_Research_Questions.py calls the
functions for answering the research questions. Figure 34 presents the main driver for calling
the two functions for answering the research questions.

```python
61  ##
62  ## Main
63  ##
64  
65  if __name__ == "__main__":
66    ## Answer research questions one and two
67    can_the_bots_and_potential_new_bot_victims_be_identified()
68  
69  ## Answer research question three
70  is_it_possible_to_monitor_bot_scanning_over_time()
```

*Figure 34. Main driver for answering the research questions.*

Shown above in Figure 34, the main portion of Answer_Research_Questions.py calls
the two functions for answering the research questions. At line 68, the
can_the_bots_and_potential_new_bot_victims_be_identified function is called to answer the
first two research questions concerning identifying and summating Bots and potential new
Bot Victims. At line 71, the is_it_possible_to_monitor_bot_scanning_over_time function is
called to answer the third research question for monitoring the Bot scanning mechanism over
time (Infosecchazzy, 2019). The next sub-section discusses Bot assessment.

**Reporting assessment.** The previous sub-section presented a code review for the
Answer_Research_Questions.py Python program for automating assessment. This sub-
section will evaluate the can_the_bots_and_potential_new_bot_victims_be_identified function for reporting assessment. The function calls the BostScannerResults.py functions for
dataset details and summary. Figure 35 shows an example of the details for a PCAP file.

<table>
<thead>
<tr>
<th>Packet Date: 2016-11-20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Subnets: [u'130.152.184.0/24', u'192.228.79.0/24']</td>
</tr>
</tbody>
</table>
Number of Packets: 93463  
Number of SYN Packets: 54302  
Number of Retransmission Packets: 39161  
Number of Bots: 40540  
Number of Potential New Bot Victims: 257

**Figure 35.** Dataset details example.

Shown above in Figure 35, the dataset details are provided for the PCAP file on 2016-11-20. *ANSWER_RESEARCH_QUESTIONS.PY FORMATTED TEXT OUTPUT* contains the details for each PCAP file of the Bot scanning dataset from the MongoDB database (Infosecchazzy, 2019). In the example above, two *destination subnets* were identified \([130.152.184.0/24, 192.228.79.0/24]\). The following was calculated for the packets (a) 93,463 *network packets*; (b) 54,302 *SYN packets*; (c) 39,161 *retransmission packets*. The PCAP file consists of 40,540 *Bots* and 573 *potential new Bot Victims*. Shown below, Figure 36 presents the summated PCAP record in the MongoDB.

```json
{id: ObjectId("5e4e471393735f2018a383f0")  
packet_date: "2016-11-20"  
total_potential_new_bot_victims: 257  
total_retransmission_packets: 39161  
total_packets: 93463  
total_syn_packets: 54302  
dest_subnet: Array  
0: "130.152.154.0/24"  
1: "192.228.79.0/24"  
total_bots: 40540
```

**Figure 36.** Dataset verification from MongoDB Daily_PCAP table.

Shown above for Figure 36, the *Daily_PCAP* table contains an entry for the PCAP file on 2016-11-20. The summation totals in the *Daily_PCAP* table match the totals from the dataset details report. The report presented accurate totals from the MongoDB database. Evaluating and verifying the assessment from the details report has provided some interesting insight into the Bot scanning mechanism over time.
- From 2016-06-01 to 2016-07-31 there is no Bot scanning activity. The PCAP files during this time period are empty.
- From 2016-08-01 to 2016-11-19 only one Subnet [130.152.184.0/24] is scanned by Bots. Multiple Bots are attempting to connect to only one potential new Bot Victim. There are no retransmission packets. This data seems to indicate targeted scanning. The original Mirai scanner randomly generates IP addresses. Future research could include further analysis of these packets.
- From 2016-11-20 to 2016-11-28 two subnets are scanned [130.152.184.0/24, 192.228.79.0/24]. The details during this time period contains varying Bot scanning summation totals as well as retransmission packets.
- From 2016-11-29 to 2017-03-08 three subnets are scanned [192.228.79.0/24, 130.152.184.0/24, 199.9.14.0/24]. The description for the dataset does not include the 199.9.14.0/24 subnet (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). The description of the dataset only included the subnets [130.152.184.0/24, 192.228.79.0/24]. Future research is needed for clarifying the subnets contained in the dataset. During this time period, summation fields contain fluctuating values.
- From 2017-03-09 to 2017-03-20 514 potential new Bot Victims are recorded while the other summation fields contain fluctuating values.
- From 2017-03-21 to 2017-03-31 four subnets were scanned [192.228.79.0/24, 199.9.14.0/24, 130.152.184.0/24, 190.103.186.0/24]. During this time period, each PCAP contained 515 potential new Bot Victims. The other summation details contain fluctuating values.

The detail report provided insight for monitoring the Bot scanning mechanism. From 2016-06-01 to 2016-07-31 there is no Bot scanning activity. 2016-08-01 to 2016-11-19 recorded targeted scanning one Subnet [130.152.184.0/24]. From 2016-11-20 to 2016-11-28 scanning increased to two subnets [130.152.184.0/24, 192.228.79.0/24]. The time period from 2016-11-29 to 2017-03-08 recorded three subnets being scanned [192.228.79.0/24, 130.152.184.0/24, 199.9.14.0/24]. From 2017-03-09 to 2017-03-20 514 potential new Bot Victims were reported. From 2017-03-21 to 2017-03-31 the scanning increased to four...
subnets [192.228.79.0/24, 199.9.14.0/24, 130.152.184.0/24, 190.103.186.0/24] and 515 potential new Bot Victims were reported. Over time, the recorded Bot scanning dataset shows a slight increase in the subnets being scanned, from one to four. From 2017-03-09 to 2017-03-31 there is a slight increase in potential new Bot Victims from 514 to 515. Future research could include additional statistical analysis of the dataset details.

The summary report provides dataset totals and averages. Figure 37 contains the summary for the Bot scanner dataset.

```
Summary for Bot Scanning Dataset
Start Date:  2016-06-01
End Date:  2017-03-31
Destination Subnets:  ['130.152.184.0/24', '192.228.79.0/24', '199.9.14.0/24', '190.103.186.0/24']

-------------------------------------------------------------------------
Total number of packets: 45422522
Total number of successful SYN packets: 24946360
Total number of re-transmission packets: 20476162

-------------------------------------------------------------------------
Average number of Bots scanning (per PCAP): 41325.53
Average number of potential new Bot Victims (per PCAP): 215.74

-------------------------------------------------------------------------
Average Number of Packets (per minute): 103.76
Average Number of Bots Scanning (per minute): 28.70
Average Potential New Bot Victims (per minute): 0.15
Average Potential New Bot Victims (per hour): 8.99

Figure 37. Dataset summary for all PCAP files.
```

Shown above in figure 37, the Bot scanner dataset is summarized spanning all the PCAP files of the dataset. The dataset contained four different destination subnets
[130.152.184.0/24, 192.228.79.0/24, 199.9.14.0/24, 190.103.186.0/24]. Totals calculated for the packets included: 45,422,522 network packets, 24,946,360 SYN packets, and 20,476,162 retransmission packets. The dataset contains a total of 12,562,962 unique instances of Bots scanning and 65,584 instances of potential new Bot Victims. Averages per PCAP include: 41,325.53 Bots and 215.74 potential new Bot Victims. Average per minute include: 103.76 packets, 28.70 Bots, and 0.15 potential new Bot Victims. Results indicate 8.99 potential new Bot Victims per hour.

The summary report provides the Bot and potential new Bot Victim totals along with packet totals. Also, averages per PCAP are calculated. The dataset summary provides averages for Bots, potential new Bot Victims, and packets. The detail report provides the contents of each summated PCAP file from the MongoDB database. Both reports, detail and summary, were generated in real time with no noticeable delay. Assessment with the detail report allows monitoring of the Bot scanning mechanism over time. The next sub-section discusses graphical assessment.

**Graphical Assessment.** This sub-section performs graphical assessment of the Bot scanner dataset. This sub-section will detail the experimentation with the

`is_it_possible_to_monitor_bot_scanning_over_time` function. This function calls the

`BotScannerResults.py dataset_summary` function in the date range from 2016-12-20 to 2016-12-24. Figure 38 presents the dataset summary.

```
Summary for Bot Scanning Dataset
Start Date:  2016-12-20
End Date:  2016-12-24
Destination Subnets:  ['199.9.14.0/24', '192.228.79.0/24', '130.152.184.0/24']
-------------------------------------------------------------------------
Total number of packets: 2702090
Total number of successful SYN packets: 1242849
Total number of re-transmission packets: 1459241
-------------------------------------------------------------------------
Average number of Bots scanning (per PCAP): 118556.40
```
Average number of potential new Bot Victims (per PCAP): 513.60

Average Number of Packets (per minute): 375.29
Average Number of Bots Scanning (per minute): 82.33
Average Potential New Bot Victims (per minute): 0.36
Average Potential New Bot Victims (per hour): 21.40

Figure 38. Dataset summary for a certain date range.

Shown above in Figure 38, the dataset summary spans the PCAP dates starting 2016-12-20 and ending on 2016-12-24. Three destination subnets are determined [199.9.14.0/24, 192.228.79.0/24, 130.152.184.0/24]. The packet totals include: 2,702,090 network packets, 1,242,849 SYN packets, and 1,459,241 retransmission packets. Averages for the PCAP include: 118,556.40 Bots and 513.60 potential new Bot Victims. Average per minute include: 375.29 packets, 82.33 Bots, and 0.36 potential new Bot Victims. Results indicate 21.40 potential new Bot Victims per hour.

The `packet_total_graph` function provides a line graph for the packet totals from the dataset summary. Figure 39 shows the line graph containing the various packet totals.

Figure 39. Packet totals line graph.
Shown above in Figure 39, the packet totals are represented in a line graph containing three lines. Starting on 12/20 and ending on 12/24, an individual line is graphed for the following totals: total packets, SYN packets, and retransmission packets. This graph monitors the packet totals over a time period starting on 12/20 and ending on 12/24. This time period shows a decline in total packets. There is a decline in SYN packets while retransmission packets seem consistent. After 12/21, retransmission packets outgain SYN packets. The Bots and potential new Bots Victims are graphed by calling the `bot_totals_graph` function. Figure 40 presents the line graph for the Bots.

![Bot Totals Graph](image)

*Figure 40. Bots line graph.*

Shown above in Figure 40, the total number of Bots per day is represented by a line graph. This graph monitors the Bot totals over a time period starting on 12/20 and ending on 12/24. Between 12/20 and 12/22 there is a steady decline for Bot totals. Start on 12/23, the Bot totals rise. Then on 12/24 the Bot totals decline. Figure 41 presents the potential new Bot victims.
Figure 41. Potential new Bot Victims line graph.

Shown above in Figure 41, the total number of potential new Bot Victims per day is represented by a line graph. This graph monitors the potential new Bot Victim totals over a time period starting on 12/20 and ending on 12/24. Between 12/20 and 12/22 total number of potential new Bot Victims remains constant at 514. Starting on 12/23, the number of potential new Bot Victims drops by one to 513.

The line graphs were produced in real time with no delay. The BotScannerResults.py line graph functions, for packet totals and the Bot totals, provide a visual representation of the Bot scanning mechanism. Experimentation with line graphing provided the assessment to monitor the Bot scanning mechanism over time. The next sub-section provides a summary.

**Single case experimentation summary.** This section presented validation of the MongoDB database for *summation* and *assessment*. A Python program, Answer_Research_Questions.py, was evaluated for assessment automation. Reporting assessment was evaluated for *dataset details* and *dataset summary*. Line graphing was assessed for Bot, potential new Bot Victim, and network packet totals. The report for the *dataset details* provided the PCAP summation totals, from the MongoDB database, that allowed for monitoring of the Bot scanning mechanism. Also, the line graphs were able to monitor the Bot scanning mechanism with visual representation. The prototype assessment component provided investigation of the Bot scanning dataset for identifying a Bot scanning
for a potential new Bot Victim. Experimentation with the prototype demonstrated *it is possible to develop a solution that can analyze network traffic to identify a Bot scanning for a potential new Bot Victim*. The next section provides a discussion of a solution comparison between the Mirai Bot Scanner Summation Prototype and NIDS.

**Prototype Solution Comparison**

The previous section discussed a single case experiment with the Mirai Bot Scanner Summation Prototype. This section compares a NIDS solution to the prototype. This section presents (a) a discussion of the method for the NIDS evaluation; (b) a discussion of the experimentation performed with the NIDS solution for generating reports and graphs; (c) a discussion for a comparison between the NIDS and the Mirai Bot Scanner Summation Prototype; (d) a summary for the prototype solution comparison. The next sub-section discusses the NIDS evaluation.

**NIDS evaluation.** A signature-based NIDS solution produces an alert log from signatures applied to a captured PCAP file for identifying malicious network activity. Typically, the NIDS alert log is forwarded to a SIEM system for further analysis and reporting (Nagaraja & Kumar, 2018). Recent research has focused on the Suricata NIDS for applying signatures to PCAP files to alert on malicious network traffic (Day & Burns, 2011). Splunk is a popular SIEM for assessing NIDS alert logs (Shah, 2015). This study performs NIDS experimentation on the same PC as in the single case experiment with the Mirai Bot Scanner Summation Prototype. The PC requirements are identical for the NIDS and prototype experimentation. The PC is a Dell computer running the Windows 10 OS with 12 GB of memory, 500GB hard drive, and an Intel Core i5 2.20 GHz processor. Suricata v4.0 and Splunk v7.2 reside on the same PC as the dataset. The Bot scanning dataset is alerted on by Suricata. Once every PCAP file of the dataset has been alerted on by Suricata, Splunk will index the alert log. Indexing the alert log allows Splunk to search thru the indexed alert log to produce reports and charts of summated Bot scanning activity. Similar to the Mirai Bot Scanner Summation Prototype, Splunk performs the assessment for monitoring the Bot scanning mechanism over time. Shown below, Table 9 contains the method for the NIDS evaluation.
Table 9

**Method for NIDS Evaluation**

<table>
<thead>
<tr>
<th>Step</th>
<th>Solution</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Python script</td>
<td>Convert PCAP Next Generation (PCAPNG) files of Bot scanner dataset to PCAP format.</td>
</tr>
<tr>
<td>2.</td>
<td>Suricata</td>
<td>Develop signature for alerting on all packets of the dataset.</td>
</tr>
<tr>
<td>4.</td>
<td>Suricata</td>
<td>Generate alerts for all of the PCAP files of the dataset.</td>
</tr>
<tr>
<td>5.</td>
<td>Splunk</td>
<td>Index alert log with Splunk.</td>
</tr>
<tr>
<td>6.</td>
<td>Splunk</td>
<td>Produce reports and charts to assess Bot scanner dataset.</td>
</tr>
</tbody>
</table>

Shown above in Table 9, the NIDS evaluation method requires six major steps. The *Step, Solution, and Description* is provided in Table 9. *Step 1* requires a Python script for converting the PCAPNG files of the dataset to PCAP files. *Step 2* describes a signature created in Suricata for detecting the SYN packets of the dataset. *Step 3* represents the configuration of the Suricata alert log. For *Step 4*, Suricata generates alerts for all of the converted PCAP files of the dataset. In *Step 5*, Splunk indexes the alert log. For *Step 6*, Splunk searches the indexed alert log to produce reports and charts to assess the Bot scanner dataset. The next sub-section discusses the PCAP conversion.

**PCAP conversion.** The dataset consists of PCAPNG files. PCAPNG is a format for storing network traffic (Velea, Ciobanu, Gurzau, & Patriciu, 2017). Suricata is unable to read the format of PCAPNG files. Suricata can process PCAP files. A Python utility was written to convert the Bot scanning dataset PCAPNG files to PCAP files (Infosecchazzy, 2019). Shown below, Figure 42 presents the `convert_pcaps` function for converting a PCAPNG file to PCAP.
Shown above in Figure 42, the convert_pcaps function is called to convert all of the PCAPNG files of the dataset to PCAP format. At line 119, tiger_shark is assigned the Tshark, the Wireshark Command Line Interface (CLI), command with the -F option for pcap format, -r for reading the pcapng_file, and -w for writing to the pcap_file. The next subsection discusses the Bot scanning signature.

**Bot scanning signature.** Suricata is a signature-based NIDS. A signature is needed to detect the Bot scanning packets from the dataset. A signature file was created for the Bot scanning (Infosecchazzy, 2019). The signature is shown below:

```python
99:     def convert_pcaps(pcapng_dir):
100:         # Directory separator
101:         Slash = '/'
102:         # Analyze the PCAP files
103:         for root, dirs, files in os.walk(pcapng_dir):  
104:             # for each file in the directory
105:                 for filename in files:
106:                     # calculate the pcap file including its path
107:                     pcapng_file = pcapng_dir + Slash + filename + Slash + filename
108:                     pcap_file = PCAP_CONV_DIR + Slash + filename
109:                     tiger_shark = "C:/Program Files/Wireshark/tshark -F pcap -r " + pcapng_file + " -w " + pcap_file
110:                     print "Tiger Shark Command: ", tiger_shark
111:                     os.system(tiger_shark)
112:     return 0

Figure 42. PCAP conversion function.
```

Shown above in Figure 42, the convert_pcaps function is called to convert all of the PCAPNG files of the dataset to PCAP format. At line 119, tiger_shark is assigned the Tshark, the Wireshark Command Line Interface (CLI), command with the -F option for pcap format, -r for reading the pcapng_file, and -w for writing to the pcap_file. The next subsection discusses the Bot scanning signature.

**Bot scanning signature.** Suricata is a signature-based NIDS. A signature is needed to detect the Bot scanning packets from the dataset. A signature file was created for the Bot scanning (Infosecchazzy, 2019). The signature is shown below:

```python
alert tcp any -> any $MIRAI_SCANNING_PORTS (flags: S; msg: "MIRAI";)
```

*Figure 43. Bot scanning signature.*

Shown above in Figure 43, based upon the Bot scanner code review performed in this study, the Bot scanning signature alerts on TCP packets from *any* source IP or port. The destination can be *any* IP. $MIRAI_SCANNING_PORTS$ is configured to contain telnet ports 23 and 2323. $MIRAI_SCANNING_PORTS$ is defined in the suricata.yaml configuration file (Infosecchazzy, 2019). *Flags: S* specifies alerting only on SYN packets. The Bot scanner dataset only contains SYN network packets. *Msg: "MIRAI"* represents a message to include
with each alert generated from the signature. The signature will alert on all of the network packets contained in the Bot scanner dataset. The signature analyzes the Bot scanner network packets to alert on a Bot scanning for a potential new Bot Victim. The next sub-section discusses the configuration of the Suricata alert log.

**Alert log configuration.** For this evaluation, the Suricata alert log is configured for JavaScript Object Notation (JSON) format. The Splunk SIEM can parse JSON formatted alert logs for assessment and charting (Shah, 2015). Shown below, Figure 44 shows the configuration for the Suricata alert log.

```
# Extensible Event Format (nicknamed EVE) event log in JSON format
- eve-log:
  enabled: yes
  filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
  filename: eve.json
```

*Figure 44. Suricata alert log configuration.*

Shown above in Figure 44, the Suricata alert log is configured for JSON format in the suricata.yaml configuration file (Infosecchazzy, 2019). The alert log is enabled, the filetype is set to regular, and the alert filename is eve.json. The next sub-section discusses generating alerts from the dataset.

**Generating alerts from dataset.** For generating alerts from the dataset, a batch script was written that calls Suricata to alert on each PCAP file of the dataset (Infosecchazzy, 2019). Shown below in Figure 45 is an example a line in the batch script.

```
suricata -c suricata.yaml -s mirai-bot-scanning.rules -r C:/MiraiCONVPCAPS/2016-08-03.pcap
```

*Figure 45. Suricata command line options.*

Shown above in Figure 45, Suricata is alerting on a PCAP file. The -c option specifies the suricata.yaml configuration file, -s specifies the mirai-bot-scanning.rules signature file, and the -r option specifies reading the C:/MiraiCONVPCAPS/2016-08-03.pcap file. Once the
batch script has completed, the *mirai-bot-scanning.rules signature* has been applied to all of
the PCAP files of the dataset. A 32.6 GB eve JSON formatted log file was created. The next
section presents indexing the alert log.

**Indexing alert log.** The Suricata eve alert log is indexed with Splunk. The index
allows Splunk to search the alert log. Figure 46 displays information from the indexed alert
log.

<table>
<thead>
<tr>
<th>Name</th>
<th>Actions</th>
<th>Type</th>
<th>App</th>
<th>Current Size</th>
<th>Max Size</th>
<th>Event Count</th>
<th>Earliest Event</th>
<th>Latest Event</th>
<th>Home Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>bot_scanning</td>
<td>Edit</td>
<td>search</td>
<td>Events</td>
<td>30.59 GB</td>
<td>500 GB</td>
<td>74.8M</td>
<td>2 years ago</td>
<td>7 days ago</td>
<td>$SPLUNK_DB/bot_scanning/db</td>
</tr>
</tbody>
</table>

*Figure 46. Splunk alert log index.*

Shown above in Figure 46, the alert log bot_scanning index is 30.59 GB. The index is
contained in $SPLUNK_DB/bot_scanning/db. Splunk references the bot_scanning index to
perform searches for generating reports and charts. The next sub-section discusses generating
Splunk reports and charts.

**NIDS reports and charts.** For monitoring the Bot scanning mechanism, Splunk
performs searches, utilizing the bot_scanning index, to produce reports and charts. This sub-
section presents (a) a search for Bot totals from the alert log; (b) a search for network packet
totals from the alert log; (c) a search for potential new Bot Victim subnets from the alert log.
The next sub-section discusses the search for Bot totals.

**Searching for bot totals.** Splunk provides a web interface for searching indexed data.
Figure 47 presents the Splunk search statement for summatting Bots and potential new Bot
Victims per PCAP file of the dataset.

```bash
source="C:\\aaaeeve\\eve.json" host="DESKTOP-D52LDB1" index="bot_scanning" | eval packet_month=case(date_month=="january","01", date_month=="february","02",
date_month=="march","03", date_month=="april","04", date_month=="may","05",
date_month=="june","06", date_month=="july","07", date_month=="august","08",
date_month=="september","09", date_month=="october","10",
```
Shown above in Figure 47, the Splunk search statement specifies the alert log source as `source="C:\aaaeve\eve.json"`. The index is specified as `index="bot_scanning"`. A chart is produced for Bots and potential new Bot Victims grouped by the PCAP date. Upon search completion, a report and chart is available for assessment. The report contains three columns representing the PCAP date, the number of Bots, and number of potential new Bot Victims (Infosecchazzy, 2019). Table 10 contains a sample from the Splunk Bot totals report.

<table>
<thead>
<tr>
<th>Pcap_Date</th>
<th>Bots</th>
<th>Pot_Bot_Victims</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/1/16</td>
<td>108</td>
<td>1</td>
</tr>
<tr>
<td>8/2/16</td>
<td>235</td>
<td>1</td>
</tr>
<tr>
<td>8/3/16</td>
<td>332</td>
<td>1</td>
</tr>
</tbody>
</table>

Shown above in Table 10, is a sampling from the Bot totals report. The Bot totals report contains a row for each PCAP of the dataset that contained network traffic. The first row is for the PCAP recorded on 8/1/16 and the last row of the contains the PCAP date of 3/31/17. For each row, `Pcap_Date` represents the PCAP date, `Bots` represents the summation of Bots on the Pcap_date, and `Pot_Bot_Victims` represents the summated potential new Bot Victims. Based upon the search results, Splunk creates a chart. Shown below in Figure 48, the Splunk Bot totals line graph is presented.
Figure 48. Splunk Bot totals line graph.

Shown above in Figure 48, Splunk produces a line graph for the summated Bots and potential new Bot Victims spanning the entire dataset of PCAP files. The line graph shows the fluctuation of Bots per PCAP while the potential new Bot Victims summation remains constant. Splunk provides the functionality for zooming in on a part of the line graph. Figure 48 displays a column chart zoomed in from the line chart.

Figure 49. Splunk Bot totals column chart.
Shown above in Figure 49, a column chart is presented starting on 2016-12-09 to 2016-12-18. On 2016-12-14, 218,803 Bots were summated. The potential new Bot Victims column, Pot_Bot_Victims, does not appear in the chart since the value [513] is much lower than the Bots total. Splunk provides the searching capabilities as well as the reporting and charting functionality to monitor the Bots and potential new Bot Victims over time. The next sub-section discusses searching for network packet totals.

**Searching for network packet totals.** To determine the network packet totals, a Splunk search is executed for summating the packet total, the unique SYN packets, and the retransmission packets. Figure 50 presents the search command for summatng the network packets.

```source="C:\\aaaeve\\eve.json" host="DESKTOP-D52LDB1" index="bot_scanning" | eval S_Packets=src_ip + dest_ip | eval packet_month=case(date_month=="january","01", date_month=="february","02", date_month=="march","03", date_month=="april","04", date_month=="may","05", date_month=="june","06", date_month=="july","07", date_month=="august","08", date_month=="september","09", date_month=="october","10", date_month=="november","11", date_month=="december","12") | eval packet_day=printf("%02d",date_mday) | eval Pcap_Date = (date_year + "-" + packet_month + "-" + packet_day) | where date_year in (2016, 2017) AND event_type == "alert" | chart sum(linecount) AS Total_Packets, distinct_count(S_Packets) AS SYN_Packets by Pcap_Date | eval Retran_Packets = Total_Packets - SYN_Packets```

*Figure 50. Splunk search for packet totals.*

Shown above in Figure 50, the packet totals search statement specifies the alert log as source="C:\\aaaeve\\eve.json". The bot_scanning index is specified. A chart is produced for Total Packets, SYN Packets, and Retransmission Packets grouped by the PCAP date [chart sum(linecount) AS Total_Packets, distinct_count(S_Packets) AS SYN_Packets by Pcap_Date | eval Retran_Packets = Total_Packets - SYN_Packets]. Upon search completion, a report and chart is available for assessment. The report contains four columns representing the PCAP date, total number of network packets, the amount of unique SYN packets, and the number of...
retransmission packets (Infosecchazzy, 2019). Table 11 contains a sample from the Splunk network packet totals report.

Table 11

<table>
<thead>
<tr>
<th>Pcap_Date</th>
<th>Total_Packets</th>
<th>SYN_Packets</th>
<th>Retran_Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/1/16</td>
<td>109</td>
<td>108</td>
<td>1</td>
</tr>
<tr>
<td>8/2/16</td>
<td>235</td>
<td>235</td>
<td>0</td>
</tr>
<tr>
<td>8/3/16</td>
<td>332</td>
<td>332</td>
<td>0</td>
</tr>
</tbody>
</table>

Shown above in Table 11, is a sampling from the network packet totals report. The report contains a row for each PCAP of the dataset that contained network traffic. The first row is for the PCAP recorded on 8/1/16 and the last row of the report contains the PCAP date of 3/31/17. For each row, Pcap_Date represents the PCAP date, Total_packets represents the summation of network packets on the Pcap_date, SYN_Packets represents the summated unique SYN packets, and Retran_Packets represents the summated retransmission packets. Based upon the search results, Splunk creates a chart. Shown below in Figure 51, the Splunk network packet totals line graph is presented.

Figure 51. Splunk packet totals line graph.
Shown above in Figure 51, Splunk produces a line graph for the summated Total_Packets, SYN_Packets, and Retran_Packets spanning the entire dataset of PCAP files. The line graph shows the fluctuation of the summated packets per PCAP. Splunk provides the functionality for zooming in on a part of the line graph. Figure 52 displays a column chart zoomed in from the line chart.

Figure 52. Splunk packet totals column chart.

Shown above in Figure 52, a column chart is presented starting on 2016-02-04 to 2016-2-23. On 2016-12-14, 1,289,585 network packets were summated. The column chart provides the Total_Packets, SYN_Packets, and Retrans_Packets totals per PCAP file. Experimentation has shown that Splunk is able to monitor the dataset network packet totals over time. The next sub-section discusses searching for potential new Bot Victim subnets.

Searching for potential new Bot Victim subnets. To determine the potential new Bot Victim subnets, a Splunk search is executed for determining the subnets based upon the PCAP date. Figure 53 presents the search command for identifying the victim subnets.
source="C:\aaaeve\eve.json" host="DESKTOP-D52LDB1" index="bot_scanning" | eval temp=split(dest_ip,".") | eval Sub0 = mvindex(temp,0) | eval Sub1 = mvindex(temp,1) | eval Sub2 = mvindex(temp,2) | eval packet_month=case(date_month=="january","01", date_month=="february","02", date_month=="march","03", date_month=="april","04", date_month=="may","05", date_month=="june","06", date_month=="july","07", date_month=="august","08", date_month=="september","09", date_month=="october","10", date_month=="november","11", date_month=="december","12") | eval packet_day=printf("%02d",date_mday) | eval Pcap_Date = (date_year + "-" + packet_month + "-" + packet_day) | where date_year in (2016, 2017) AND event_type == "alert" | dedup Sub0, Sub1, Sub2, Pcap_Date | eval D_Subnet = printf("%d.%d.%d.0/24",Sub0, Sub1, Sub2) | table botscanning_subnets Pcap_Date D_Subnet | sort by Pcap_Date

Shown above in Figure 53, the subnet search statement specifies the alert log as source="C:\aaaeve\eve.json". The "bot_scanning" index is specified. A table [botscanning_subnets] is created containing the PCAP Date [Pcap_Date] and the potential new Bot Victim subnet [D_Subnet] sorted by the PCAP date [table botscanning_subnets Pcap_Date D_Subnet | sort by Pcap_Date]. Upon search completion, a table and report is available for assessment. The table and report contains two columns. One column represents the PCAP date and the other column represents the subnet (Infosecchazzy, 2019). Table 12 contains a sample from the Splunk subnet report.

Table 12

<table>
<thead>
<tr>
<th>PCAP DATE</th>
<th>SUBNET</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/24/16</td>
<td>192.228.79.0/24</td>
</tr>
<tr>
<td>11/24/16</td>
<td>130.152.184.0/24</td>
</tr>
</tbody>
</table>

Shown above in Table 12, is a sampling from the subnet report. The report contains a row for each new potential Bot Victim subnet. In the sample above, the PCAP file dated 11/24/16 contained two new potential Bot Victim subnets [192.228.79.0/24,
The report contains a row for each potential new Bot Victim subnet identified in the PCAP file. The next sub-section provides the prototype comparison with the NIDS solution.

**Prototype comparison with NIDS solution.** This sub-section compares the evaluation of the Mirai Bot Scanner Summation Prototype with the NIDS solution consisting of Suricata integrated with Splunk. This section presents (a) a comparison of the methods; (b) a comparison of performance; (c) a comparison of Bot scanning assessment results. The next sub-section discusses the method comparison between the prototype and NIDS solution.

**Methods comparison.** The Mirai Bot Scanner Summation Prototype contains a component for summating the dataset and a component that provides for assessment consisting of reports and line-graphs. The NIDS solution consists of the Suricata NIDS for producing an alert log and the Snort SIEM for assessing the alert log with reports and charts. Shown below in Table 12 is the methods a Bot researcher follows for the prototype and NIDS solution.

<table>
<thead>
<tr>
<th>Step</th>
<th>Prototype Component</th>
<th>Description</th>
<th>NIDS Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Summation Component</td>
<td>The PCAP files are summated into a MongoDB database via Python functions.</td>
<td>Python script</td>
<td>Convert PCAP Next Generation (PCAPNG) files of Bot scanner dataset to PCAP format.</td>
</tr>
<tr>
<td>2.</td>
<td>Assessment Component</td>
<td>Real-time reports and line-graphs produced from Python functions searching the MongoDB database.</td>
<td>Suricata</td>
<td>Generate alerts, with the Bot scanning signature, for all of the PCAP files of the dataset.</td>
</tr>
<tr>
<td>3.</td>
<td>-</td>
<td>Splunk</td>
<td>Index alert log with Splunk.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>-</td>
<td>Splunk</td>
<td>Search indexed alert log to produce reports and charts to assess Bot scanner dataset.</td>
<td></td>
</tr>
</tbody>
</table>
Shown above in Table 13, the Mirai Bot Scanner Prototype contains two methods, or steps, for evaluating the dataset. The Bot researcher summates the packets contained in the PCAP files of the dataset. Then, the Bot researcher performs assessment thru Python functions. Concerning the NIDS solution, four steps are required. In the first step, the PCAP files must be converted from PCAPNG format to PCAP format. Suricata is unable to read PCAPNG files. The second step requires Suricata to generate the alert log. The alert log is produced by reading the PCAP converted files of the dataset and applying the Bot scanning signature. In the third step, the alert log is indexed by Splunk. Indexing allows Splunk to perform searches of the alert log. In the final step, the Bot researcher executes searches for assessment of the alerted Bot scanning. The NIDS solution requires two more steps than the prototype. The prototype is able to read the PCAP files while a conversion of the PCAP files is necessary for Suricata. Evaluation showed that assessment with Splunk required complex searches whereas the prototype requires a Python function call with two parameters for specifying a date range. The next sub-section provides a performance comparison.

**Performance comparison.** This section provides a performance comparison. The Mirai Bot Scanner Summation Prototype and the NIDS solution was evaluated on the same PC. The performance comparison will consist of comparing the run-time for the summation and assessment process of each solution.

**Table 14**

*Performance Comparison*

<table>
<thead>
<tr>
<th>Step</th>
<th>Prototype Component</th>
<th>Description</th>
<th>Run Time (Hr.)</th>
<th>NIDS Component</th>
<th>Description</th>
<th>Run Time (Hr.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Summation Component</td>
<td>The PCAP files are summated into a database.</td>
<td>19</td>
<td>Python script</td>
<td>Convert PCAPNG files of Bot scanner dataset to PCAP format.</td>
<td>1</td>
</tr>
</tbody>
</table>
Table 14 (continued)

*Performance Comparison*

<table>
<thead>
<tr>
<th>Step</th>
<th>Prototype Component</th>
<th>Description</th>
<th>Run Time (Hr.)</th>
<th>NIDS Component</th>
<th>Description</th>
<th>Run Time (Hr.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Assessment Component</td>
<td>Real-time Reports and line-graphs produced from Python functions.</td>
<td>0</td>
<td>Suricata</td>
<td>Generate alerts, with the Bot scanning signature, for all of the PCAP files of the dataset.</td>
<td>19</td>
</tr>
<tr>
<td>3.</td>
<td>-</td>
<td>Splunk</td>
<td>Index alert log with Splunk.</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>-</td>
<td>Splunk</td>
<td>Produce reports and charts to assess Bot scanner dataset.</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td></td>
<td></td>
<td><strong>19</strong></td>
<td></td>
<td></td>
<td><strong>25</strong></td>
</tr>
</tbody>
</table>

Shown above in Table 14, the Mirai Bot Scanner Summation Prototype required 19 hours for summation. Assessment was performed real time. Concerning the NIDS solution, a total of 25 hours was required for the evaluation. For the first step of the NIDS solution, one hour was needed to perform the PCAP conversion. For the second step, Suricata required 19 hours to produce the alert log. Concerning the third step, Splunk required four hours for indexing the alert log. For step four, Splunk required one hour to search the index for assessment that produced reports and charts. The prototype required seven less hours, or 24% less time, to summate and assess the Bot scanning dataset. The next sub-section presents a comparison for the Bot scanning results.

**Bot scanning results comparison.** During experimentation, the Mirai Bot Scanner Summation Prototype summated the Bot scanning mechanism per PCAP file (Infosecchazzy, 2019). The Bots, potential new Bot Victims, and network packets were summated. For the NIDS solution, a Splunk report was generated for Bot totals per PCAP file. Another report
summated the packet totals. The prototype and Splunk summation totals are compared to determine the accuracy of the assessment of the dataset (Infosecchazzy, 2019). Table 15 presents a solution totals comparison.

Table 15

Solution Totals Comparison

<table>
<thead>
<tr>
<th>Assessment Category</th>
<th>Prototype</th>
<th>Splunk</th>
<th>Splunk Percentage Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bots</td>
<td>12562962</td>
<td>12505157</td>
<td>-0.46%</td>
</tr>
<tr>
<td>Potential New Bot Victims</td>
<td>65584</td>
<td>65587</td>
<td>0.00%</td>
</tr>
<tr>
<td>Network Packets</td>
<td>45422522</td>
<td>45288087</td>
<td>-0.30%</td>
</tr>
<tr>
<td>SYN Packets</td>
<td>24946360</td>
<td>27229399</td>
<td>8.38%</td>
</tr>
<tr>
<td>Retransmission Packets</td>
<td>20476162</td>
<td>18058688</td>
<td>-13.39%</td>
</tr>
</tbody>
</table>

Shown above for Table 15, the total amount of Bots, Potential New Bot Victims, Network Packets, SYN Packets, and Retransmission Packets is compared between the Prototype and Splunk assessment. The Splunk percentage difference is provided. There is an insignificant percentage difference for Bots, Potential New Bot Victims, and Network Packets. There is an 8.38% difference for SYN Packets. Also, there is a 13.39% difference for Retransmission Packets. The difference between the prototype and Splunk totals is expected since the prototype summation algorithm classifies Bots, potential new Bot Victims, and network packets differently than the Splunk searches performed for summation. Concerning the SYN summation, the prototype includes packets with only one instance. The originating SYN packet that initiated the retransmission packets is included with the retransmission summation. The Splunk search does not include the originating SYN packet in the summation for retransmission packets. Compared to the Splunk totals, it is anticipated that the prototype will have less SYN packet totals and more retransmission packet totals.
The `Tshark` CLI for Wireshark can be executed to summate the SYN and retransmission packets per PCAP file. Figure presents summatung the PCAP SYN packets with Tshark.

```
C:"Program Files"\Wireshark\Tshark -r <filename> -T fields -e ip.src -e ip.dst | sort | uniq | wc -l
```

*Figure 54. Tshark SYN packet summation.*

Shown above in Figure 54, Tshark reads a PCAP file `[-r <filename>]` and the ip.src and ip.dst fields are chosen `[-T fields -e ip.src -e ip.dst]`. The list is `sorted`, then `uniq` rows determined, and finally a count of the rows is taken `[wc -l]`. The count of the rows determines the amount of unique SYN packets. Also, Tshark can summate the number of retransmission packets per PCAP file. Figure 55 presents the Tshark command for retransmission packet summation.

```
C:"Program Files"\Wireshark\tshark -r <filename> -Y "tcp.analysis.retransmission" | wc -l
```

*Figure 55. Tshark retransmission packet summation.*

Shown above in Figure 55, Tshark reads a PCAP file `[-r <filename>]` and applies the read filter for retransmission packets `[-Y "tcp.analysis.retransmission"]`. The count of the rows `[wc -l]` determines the amount retransmission packets. Table 16 provides a sampling of the dataset for comparing the Tshark SYN and retransmission packet totals to the Mirai Bot Scanner Summation Prototype and Splunk.

Table 16

<table>
<thead>
<tr>
<th>PCAP Date</th>
<th>Packet Type</th>
<th>Tshark</th>
<th>Prototype</th>
<th>Prototype % Diff. with Tshark</th>
<th>Splunk</th>
<th>Splunk % Diff. with Tshark</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/11/16</td>
<td>SYN</td>
<td>272</td>
<td>271</td>
<td>0.37%</td>
<td>261</td>
<td>4.04%</td>
</tr>
<tr>
<td>8/11/16</td>
<td>Retran.</td>
<td>0</td>
<td>2</td>
<td>0.00%</td>
<td>1</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
Table 16 (continued)

*Tshark Packet Comparison*

<table>
<thead>
<tr>
<th>PCAP Date</th>
<th>Packet Type</th>
<th>Tshark</th>
<th>Prototype</th>
<th>Prototype % Diff. with Tshark</th>
<th>Splunk</th>
<th>Splunk % Diff. with Tshark</th>
</tr>
</thead>
<tbody>
<tr>
<td>9/22/16</td>
<td>SYN</td>
<td>909</td>
<td>869</td>
<td>4.40%</td>
<td>979</td>
<td>-7.70%</td>
</tr>
<tr>
<td>9/22/16</td>
<td>Retran.</td>
<td>217</td>
<td>321</td>
<td>-47.93%</td>
<td>244</td>
<td>-12.44%</td>
</tr>
<tr>
<td>10/27/16</td>
<td>SYN</td>
<td>1009</td>
<td>842</td>
<td>16.55%</td>
<td>1017</td>
<td>-0.79%</td>
</tr>
<tr>
<td>10/27/16</td>
<td>Retran.</td>
<td>1477</td>
<td>1810</td>
<td>-22.55%</td>
<td>1585</td>
<td>-7.31%</td>
</tr>
<tr>
<td>11/5/16</td>
<td>SYN</td>
<td>1817</td>
<td>1633</td>
<td>10.13%</td>
<td>1867</td>
<td>-2.75%</td>
</tr>
<tr>
<td>11/5/16</td>
<td>Retran.</td>
<td>1147</td>
<td>1453</td>
<td>-26.68%</td>
<td>1233</td>
<td>-7.50%</td>
</tr>
<tr>
<td>12/22/16</td>
<td>SYN</td>
<td>158289</td>
<td>129480</td>
<td>18.20%</td>
<td>148421</td>
<td>6.23%</td>
</tr>
<tr>
<td>12/22/16</td>
<td>Retran.</td>
<td>205066</td>
<td>250349</td>
<td>-22.08%</td>
<td>187291</td>
<td>8.67%</td>
</tr>
<tr>
<td>1/11/17</td>
<td>SYN</td>
<td>159842</td>
<td>136810</td>
<td>14.41%</td>
<td>159977</td>
<td>-0.08%</td>
</tr>
<tr>
<td>1/11/17</td>
<td>Retran.</td>
<td>206785</td>
<td>244711</td>
<td>-18.34%</td>
<td>200833</td>
<td>2.88%</td>
</tr>
<tr>
<td>2/15/17</td>
<td>SYN</td>
<td>121382</td>
<td>109332</td>
<td>9.93%</td>
<td>123882</td>
<td>-2.06%</td>
</tr>
<tr>
<td>2/15/17</td>
<td>Retran.</td>
<td>102083</td>
<td>123049</td>
<td>-20.54%</td>
<td>111247</td>
<td>-8.98%</td>
</tr>
<tr>
<td>3/19/17</td>
<td>SYN</td>
<td>194794</td>
<td>186687</td>
<td>4.16%</td>
<td>193105</td>
<td>0.87%</td>
</tr>
<tr>
<td>3/19/17</td>
<td>Retran.</td>
<td>34694</td>
<td>46673</td>
<td>-34.53%</td>
<td>37688</td>
<td>-8.63%</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>1189783</td>
<td>1234292</td>
<td>-3.74%</td>
<td>1169631</td>
<td>1.69%</td>
</tr>
</tbody>
</table>

Shown above in Table 16, the prototype and Splunk SYN and retransmission counts are compared with Tshark. There is a 3.74% total difference for SYN and retransmission packet totals between the prototype and Tshark. A total difference of 1.69% is calculated for the difference between Tshark and Splunk. Splunk percentage differences for SYN and retransmission packets do not contain as wide a range as compared to the prototype. Concerning Splunk, the largest difference was on 9/22/16 with a 12.44% difference for retransmission packets. On 12/12/16, Splunk experienced its highest difference for SYN packets, 6.23%. On 9/22/16, concerning the prototype, there is a 47.93% difference in retransmission packets. On 12/22/16, there is a 18.20% difference in SYN packets for the prototype. The prototype experienced a wide range of difference ranging from -47.93% for retransmission packet totals on 9/22/16 to 18.22% for SYN packet totals on 10/22/16. The differences between Tshark are expected.
Tshark calculates retransmission packets based upon old sequence numbers and retransmission timeout (RTO) (Asrodia & Patel, 2012). The Prototype and Splunk classify SYN and retransmission packets differently. The Splunk and Tshark searches classify unique IP addresses for SYN packet totals. The Splunk retransmission packet total calculation is performed by subtracting the SYN total from the network packet total. The Mirai Bot Scanner Summation Prototype algorithm classifies SYN and retransmission packets differently. Concerning the SYN summation, the prototype includes packets with only one instance. The originating SYN packet that initiated the retransmission packets is included with the retransmission summation. Tshark and Splunk do not include the originating SYN packet in their summation for retransmission packets. Therefore, it is expected that the prototype will have less SYN packet totals and more retransmission packet totals per PCAP. The next sub-section provides a summary.

**Prototype solution comparison summary.** The Suricata solution was evaluated for creating an alert log from the dataset. Required by Suricata for reading PCAP files, the dataset PCAP files were converted from PCAPNG format to PCAP format. Suricata read each converted PCAP file of the dataset to apply a Mirai Bot scanning signature to alert on all of the Bot scanning network packets. Since Suricata does not contain any functionality for assessing the alert log, the Splunk SIEM indexed the alert log. Once the alert log was indexed, Splunk searched the indexed alert log for assessing Bot totals and network packet totals. Splunk produced reports as well as various charts for assessment. Evaluation of the NIDS solution, Suricata and Splunk, was compared with the Mirai Bot Scanner Summation prototype. The prototype required 50% less steps for summating and assessing the dataset. Analysis of the run-time steps indicates that the prototype is 24% faster for summation and assessment. Lastly, a comparison of summation results with the Tshark packet analyzer shows a difference in SYN and retransmission packets for the prototype as well as Splunk. The difference with Splunk is unexpected since Splunk summates the SYN packets similar to Tshark. The Mirai Bot Scanner Summation Prototype algorithm classifies SYN and retransmission packets differently than Tshark. The prototype classifies unique SYN packets and includes the originating SYN packet with retransmission totals. The difference in classification is expected to cause a difference for the Mirai Bot Scanner Summation
Prototype summation of SYN and retransmission packets. The next section provides a chapter summary.

**Chapter Summary**

Chapter 4 presents (a) a code review of the Bot scanner; (b) network packet analysis from a sample PCAP file of the Bot scanning dataset; (c) Mirai Bot Scanner Summation Prototype code review; (d) single case experimentation performed with the Mirai Bot Scanner Summation Prototype; (e) a comparison of the prototype to a NIDS solution. The Bot scanner code review identified the Bot scanning mechanism of attempting to establish a connection to a potential new Bot Victim with a TCP handshake over the telnet port 23 or 2323. The packet analysis of the Bot scanning dataset identified the SYN and retransmission packets. The Mirai Bot Scanner Summation Prototype code review presented the Python programs for the summation and assessment components. Experimentation with the Mirai Bot Scanner Summation Prototype assessed the reporting and graphing capabilities for monitoring the Bot scanning mechanism over time. Comparison with the Suricata NIDS and Splunk SIEM for the dataset assessment indicates the efficiency and effectiveness of the prototype for summatng and assessing the Bot scanning dataset. Both, the prototype and the NIDS solution, were able to assess the Bot scanning dataset to identify Bots, potential new Bot Victims, and Bot scanning network packets. Both solutions monitored the Bot scanning mechanism over time. Experimentation performed with the prototype and NIDS demonstrated *it is possible to develop a solution that can analyze network traffic to identify a Bot scanning for a potential new Bot Victim.* The next chapter provides the conclusion of this study.
CHAPTER 5

CONCLUSION

The focus of this study is performing single case experimentation with the Mirai Bot Scanner Summation Prototype for summatting a Bot scanning dataset and assessing the Bot scanning mechanism over time. Chapter 4 performed experimentation to answer the research questions presented for this dissertation. The experimentation comprised of a code review of the Bot scanner, network packet analysis from a sample PCAP file of the Bot scanning dataset, a Mirai Bot Scanner Summation Prototype code review, and research questions answered from experimentation performed with the Mirai Bot Scanner Summation Prototype. A comparison to a NIDS solution indicated the usefulness of the prototype for assessing the Bot scanning dataset.

Chapter 5 contains the following sections (a) an overview of this study; (b) contributions from the completed research; (c) findings of the research conducted; (d) limitations of the study; (e) recommendations for future research. The next section provides an overview of this study.

Overview

This research performed experimentation with the Mirai Bot Scanner Summation Prototype to summate and assess a Bot scanning dataset (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). The Bot scanning dataset contains PCAP files starting on June 1, 2016 and ending on March 31, 2017. It includes Bot TCP SYNs sent to ports 23 and 2323. The Mirai Bot Scanner Summation Prototype assesses the Bot scanning dataset to answer the research questions of this dissertation. The Mirai Bot Scanner Summation Prototype (a) summates Bots; (b) summates potential new Bot Victims; (c) summates packet totals; (d) assesses or monitors the Bot scanner mechanism over time.
The Mirai Bot Scanner Summation Prototype consists of a summation and assessment component. The summation and assessment components interface with a persistent database. The assessment component contains the functionality for summation of each PCAP file of the Bot scanner dataset. The assessment component provides reports and line graphs. The summation component stores the tabulated Bot scanning totals, per PCAP file, in the persistent database. The assessment component searches the database for the records needed for reporting and graphing. The experimentation performed with the Mirai Bot Scanner Summation Prototype is automated. Reporting and line-graphing assessment was able to monitor the Bot scanning mechanism over time. Compared to a NIDS solution, the prototype has been shown to be advantageous for summing and assessing the Bot scanning dataset. The next sub-section provides the major findings.

Contributions

The previous sub-section provided an overview of the Bot scanning dataset and the Mirai Bot Scanner Summation Prototype for summing and assessing the dataset. This sub-section presents the contributions of this study. According to Mirai botnet researchers (Kolias et al., 2017), “Surprisingly, IoT botnets have received only sporadic attention from researchers” (p. 81). Current research has performed dynamic analysis of Mirai network traffic, but not concentrated on the Bot scanner (Kolias et al., 2017; Sinanović & Mrdovic, 2017). Considering that Mirai is receiving only sporadic attention, the major contribution of this study is in performing experimentation with an actual Bot scanning dataset.

This study follows the DSR research methodology. The main DSR requirements are rigor and relevance (Offermann et al., 2009). This study performs rigorous experimentation with the Mirai Bot Scanner Summation Prototype for summing and assessing the recorded Bot scanning dataset. The Bot scanner code review reveals the Bot functionality for a Bot connecting to a potential new Bot Victim. Sporadic code reviews have been performed for the Bot scanner (Conley, 2017; Kolias et al., 2017; Sinanović & Mrdovic, 2017). This study presents a comprehensive code review for a Bot trying to make a connection to a potential new Bot Victim. Analysis of a sample PCAP file from the Bot scanning dataset confirms the Bot scanner functionality discovered from the code review. This study presents the rigor and relevance necessary for experimentation with the Bot scanning dataset.
The Mirai Bot Scanner Summation Prototype summates each PCAP file of the Bot scanning dataset. Each PCAP is analyzed and the results are stored in a MongoDB table. Therefore, each PCAP file needs to be analyzed once to determine the number of Bots, potential new Bot Victims, and packet totals. The Bot scanning results functions, which include (a) dataset details; (b) dataset summary; (c) packet totals line graphs; (d) bot totals line graphs, which provide formatted text and graphical line charts. These functions require the start date and end date as parameters. The results of any valid date range of the Bot scanning dataset can be assessed. The prototype for this study provides the ease and flexibility to assess the dataset for the desired date range, which provides for rigor and relevance. The next sub-section presents the findings from the experimentation performed for this study.

Findings

The previous sub-section presented the contributions of this study. Considering that Mirai botnet research is sporadic, a major contribution of this study is performing experimentation with a Bot scanner dataset. A comprehensive code review for a Bot trying to make a connection to a potential new Bot Victim is performed. Analysis of a sample PCAP is reinforced with the Bot scanning functionality discovery from the code review. The Mirai Bot Scanner Prototype provides the components to summate and assess the dataset. This sub-section presents the findings from the (a) code review; (b) sample PCAP analysis; (c) experimentation performed with the prototype; (d) comparison of the prototype with NIDS; (e) discussion of the research questions.

The Bot scanner code review revealed that the Bot randomly generates an IP address. Corporate subnets are blacklisted, such as General Electric and Hewlett-Packard, as well as US government subnets, such as the US Postal Service and the Department of Defense. The Bot attempts to establish a TCP connection, thru a network socket, with the randomly generated IP address via a TCP handshake over telnet ports 23 and 2323. The Bot will progress to brute-forcing when an ACK packet is received from the potential new Bot victim. The code review provided the necessary background for analyzing the Bot scanner network traffic captured in a PCAP file.
Analysis with the Wireshark packet analysis tool shows that the sample PCAP file contains TCP SYN packets and retransmission packets. The SYN packet depicts a Bot attempting to connect to a potential new Bot Victim over telnet ports. The retransmission packet represents a potential new Bot Victim not responding to the SYN packet with an ACK packet. The retransmission packet indicates a failure of the Bot connecting to a potential new Bot Victim. The Bots are contained in the source field of the network packets. The potential new Bot Victims are contained in the destination field of the SYN packets. Analysis of the PCAP file provided necessary knowledge for the solution.

The prototype summates each PCAP file. Based upon the SYN packets contained in the PCAP file, the number of Bots and potential new Bot Victims are calculated and stored in a database table. The summation process of the Mirai Bot Scanner Summation Prototype required approximately 19 hours and 15 minutes on a PC to summate all the PCAP files and store the tabulated results for each individual PCAP file in a table (Infosecchazzy, 2019). The solution was able to dynamically provide reports and line graphs for assessment, based upon a date range. Based upon the date range, the PCAP details database table is queried to gather the total number of Bots, total number of potential new Bot Victims, total number of SYN packets, and total number of retransmission packets. The Mirai Bot Scanner Summation Prototype provides real time response for reports and line graphs.

Comparison with a NIDS solution, Suricata NIDS and Splunk SIEM, highlights the advantages of the prototype. Suricata produces an alert log from applying a Bot scanning signature to all of the PCAP files. Snort indexes the alert log to enable searches that produce reports and charts of the Bot scanning mechanism over time. The NIDS solution requires two solutions interfacing with each other while the prototype is self-contained and does not interface with other solutions. The NIDS solution requires more steps for assessing the dataset and these steps require 24% more time to complete. Comparing the assessment results, the Mirai Bot Scanner Summation Prototype summation algorithm classifies SYN and retransmission network packets differently than Splunk, therefore a comparison of the summation results expected differences in summation totals.

Experimentation demonstrated with the Mirai Bot Scanner Summation Prototype and NIDS solution verified it is possible to develop a solution that can analyze network traffic to
identify a Bot scanning for a potential new Bot Victim. These solutions were able to answer the three research questions presented in this study:

1. Can the Bots be identified for summation?
2. Can the potential new Bot Victims be identified for summation?
3. Is it possible to monitor the Bot scanning mechanism over time?

**Bots identified.** Concerning the first research question, the prototype is able to read the source IP address for each network packet of the Bot scanning dataset. The source IP represents the Bot. The prototype summation component is able to tabulate the Bot totals. The assessment component is able to evaluate Bot totals. With the Suricata NIDS solution, a Bot scanning signature alerts on each network packet of the dataset. The alert log is indexed by the Splunk SIEM to allow for a query to summate and assess the Bots based upon the source IP in the alert log. Experimentation with the prototype and NIDS has shown that the Bots can be identified for summation.

**Potential new Bot Victims identified.** Similar to the process for answering the first research question, the prototype is able to read the destination IP address for each network packet of the Bot scanning dataset. The destination IP represents the potential new Bot Victim. The prototype summation component is able to tabulate the potential new Bot Victim totals. The assessment component is able to evaluate the Bot Victim totals. With the Suricata NIDS solution, a Bot scanning signature alerts on each network packet of the dataset. The alert log is indexed by the Splunk SIEM to allow for a query to summate and assess the potential new Bot Victims based upon the destination IP in the alert log. Experimentation with the prototype and NIDS has shown that the potential new Bot Victims can be identified for summation.

**Bot scanning mechanism monitored.** For the third research question, the Mirai Bot Scanner Summation Prototype and Splunk assess Bot totals and potential new Bot Victims totals over time. The prototype provides Python functions for assessment. The assessment is performed based upon the starting and ending date supplied by the botnet researcher. A Splunk query, constructed by the researcher, assesses the Bot scanning. The prototype and the NIDS solution, consisting of Suricata and Splunk, assess Bots, potential new Bot Victims, and network packet totals over time. Experimentation performed with the prototype and
NIDS solution has demonstrated it is possible to monitor the Bot scanning mechanism over time. The next sub-section presents the limitations of this study.

**Limitations**

The previous sub-section discussed the findings of the study. Considering that Mirai botnet research is sporadic, a major contribution of this study is performing experimentation with a Bot scanner dataset. A comprehensive code review reveals the functionality of a Bot connecting to a potential new Bot Victim. Analysis of a sample PCAP discloses the network packet types, SYN packets and retransmission packets, sent from the Bot to the potential new Bot Victim. The Mirai Bot Scanner Summation Prototype provides the components to summate and assess the dataset. This sub-section presents the limitations of the study.

The Bot scanner dataset contains only Bot scanning packets (*Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016*). The prototype components contain functionality specific to the Bot scanning dataset containing only SYN packets and retransmission packets. The dataset does not contain non-Bot scanning packets and thus the porotype does not summate or assess for network packets besides TCP SYN and retransmission packets. The Mirai Bot Scanner Summation Prototype is a specific solution for only summatint and accessing Bot scanning TCP SYN packets which are contained in the dataset.

The prototype contains specific functionality for summation and assessment. If additional functionality is need, then the Mirai Bot Scanner Summation Prototype needs to be modified. Each individual PCAP file is summated for the number of Bots, potential new Bot Victims, SYN packets, and retransmission packets. If summation requires tabulating different fields for the SYN packet, the summation component will need to be modified. The assessment component is specific to the tabulated information stored in the database from the summation component or process. If the summation is modified to include additional fields from the SYN packet, assessment component will need to be modified to include the additional fields for reporting and graphing. The next sub-section presents future research for summatint and assessing Bot scanning mechanism.
Future Research

The previous sub-section discussed limitations for this study. The Bot scanner dataset contains Bot scanning TCP SYN packets only. The Mirai Bot Scanner Summation Prototype is specific to the Bot scanner dataset. Additional functional requirements will cause modification of the summation and assessment components. This sub-section presents future Bot scanning research.

This research filled a void regarding the Mirai botnet. Experimentation is performed with the Mirai Bot Scanner Summation Prototype to summate and assess a recorded Bot scanner dataset. The prototype summates the dataset to tabulate the number of Bots and potential new Bot Victims. Additional summation could be performed to determine the non-vulnerable IoT devices. The Bot scanner dataset contains retransmission packets. A retransmission packet represents a destination IP address that has not responded to the SYN packet sent by the Bot. To determine the non-vulnerable IoT devices, the retransmission packets could be analyzed to summate the unique destination IP addresses within the retransmission packets.

The Mirai Bot Scanner Summation Prototype components contain functionality that is specific to the Bot scanner dataset (Internet Addresses Census dataset, IMPACT ID: USC-LANDER/Mirai-B-scanning-20160601/rev5870, 2016). Considering that assessment searches for summated records in the database, additional Bot researcher functionality could provide added insight into the Bot scanner dataset. Future research could include the use of Artificial Intelligence (AI) techniques, within the prototype components, for learning from the Bot scanner dataset. Future research could include experimentation with the prototype that includes a different Bot scanning dataset. The dataset could include Bot scanning and non-Bot scanning network packets. The prototype could be modified to account for the different network packets captured in a dataset.

Researchers (Kolias et al., 2017) describe the operation and communication steps of Mirai. The Bot brute-forcing is described as the Bot connecting to a potential new Bot Victim and then the Bot attempting to log into the potential new Bot Victim with a factory default user-id and password. Currently the solution is focused on analyzing the network traffic of a Bot connecting to a potential new Bot Victim. Future research could include identifying a Bot attempting to remotely log into a potential new Bot Victim. Once a Bot brute-forces the
potential new Bot Victim, the logon information of the potential new Bot Victim is sent from the Bot to the report server (Kolias et al., 2017). The prototype could be extended to identify a Bot communicating back to the C&C report server. The other function of the Bot is to respond to C&C for executing DOS attacks (Kolias et al., 2017). The solution could include identifying a Bot via C&C. The solution could be modified to include additional network packet analysis for identifying a Bot based upon command and operational functionality. The command and operational functionality including: A Bot remotely accessing a potential new Bot Victim with a factory default user-id and password, a Bot sending the logon information of the potential new Bot Victim back to the report server, and a Bot responding to a C&C request to execute a DOS attack.
REFERENCES


the Proceedings of the 11th Annual Cyber and Information Security Research Conference, Oak Ridge, TN, USA.


