Digital Quorum Sensing
for Self-Organizing Malware

by
Mark E. Fioravanti II

A thesis submitted to
Florida Institute of Technology
in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Computer Science

Melbourne, Florida
December 2016
We the undersigned committee hereby approve the attached thesis

**Digital Quorum Sensing**
**for Self-Organizing Malware**

by
Mark E. Fioravanti II

________________________________________________________
Richard Ford, Ph.D.
Professor, School of Computing
Committee Chair, Thesis Advisor

________________________________________________________
Keith Gallagher, Ph.D.
Associate Professor, School of Computing
Thesis Committee

________________________________________________________
Heather Crawford, Ph.D.
Assistant Professor, School of Computing
Thesis Committee

________________________________________________________
Richard Turner, Ph.D.
Professor, Biological Sciences
Thesis Committee

________________________________________________________
Ronaldo Menezes, Ph.D.
Professor, School of Computing
Program Chair, Computer Sciences and Software Engineering
Abstract

Title: Digital Quorum Sensing for Self-Organizing Malware
Author: Mark E. Fioravanti II
Committee Chair: Richard Ford, Ph.D.

INTRODUCTION: Malware authors present an interesting problem for the security community as they evolve and adapt to overcome network and host defenses. The determined adversary is a special class of malware author who may attempt to disrupt national interests. These adversaries may seek potentially novel Command and Control (C2) channels to coordinate their activities. Isolated and air-gapped networks pose an interesting challenge that these adversaries must adapt to in order to maintain persistence on these networks. In this work we propose that a determined adversary may seek to implement a digital quorum sensing system inspired by the quorum sensing systems used by some bacteria to coordinate their social behaviors.

OBJECTIVES: The primary objective of this research was to characterize a potential digital quorum sensing C2 channel that relies on subtly modifying the global packet distribution on a network.

METHODS: A proof of concept was developed and studied to determine if a C2 channel based on quorum sensing is feasible. Based on the results of the proof of
concept, a prototype was implemented and studied in a number of different networking environments in order to more fully characterize the signal. The strength of the quorum sensing signal (the independent variable) was adjusted and through a series of statistical tests the statistical significance of the impact on the global packet distribution was determined.

RESULTS: Network packet captures were analyzed from several different networks with Friedman tests. When the probability of a delaying packets was approximately in the range of (0.25,0.1) the delay was statistically significant with alpha=0.05 for the global packet distribution but not for the packet counts observed from the individual hosts. Wilcoxon rank-sum tests were used to determine which portions of the data sets contained statistically significant deviations, at a significance level of 95% (alpha=0.05).

CONCLUSION: Digital quorum sensing could be used as a novel C2 channel providing a determined adversary a unique method of coordinating activities on a network without allowing the network defender to identify the infected hosts. During the experiment it was observed that this signal is easy to disrupt by altering the time synchronization between the hosts on the network.
# Contents

1 Introduction .......................................................... 1
   1.1 Contributions ................................................. 2
   1.2 Terminology .................................................. 4
   1.3 Organization .................................................. 9

2 Background ........................................................... 10
   2.1 Malware ....................................................... 12
      2.1.1 From Theory to Practice ................................ 13
      2.1.2 Command and Control ................................... 15
      2.1.3 Targeted Malware ....................................... 24
   2.2 Biologically Inspired Computing ............................ 27
   2.3 Bacterial Quorum Sensing .................................. 31
   2.4 Determinism vs Non-Determinism ............................ 38
   2.5 Covert Channels ............................................. 41
   2.6 Digital Quorum Sensing ..................................... 53
   2.7 Ethics of Security Vulnerability Research ............... 55
      2.7.1 Effects of Choosing to Disclose a Vulnerability .... 59
      2.7.2 Process of Vulnerability Disclosure ................. 63
3 Architecture

3.1 Cocktail Party Example ........................................... 67
3.2 Proposed Digital Quorum Sensing Architecture ............... 71
3.3 Proposed Applications of Digital Quorum Sensing ............ 78

4 Proof of Concept Study

4.1 Design ................................................................. 89
4.2 Results ................................................................. 91
   4.2.1 Normal Web Browsing Traffic (Control) ................. 91
   4.2.2 Flat Virtualized Network .................................... 92
   4.2.3 Local Wireless Network ..................................... 93
   4.2.4 Internet-based Hosts ......................................... 95
4.3 Discussion ............................................................. 97

5 Prototype Study

5.1 Design ................................................................. 101
   5.1.1 Design Revisions ............................................. 101
   5.1.2 Environment Setup ........................................... 119
5.2 Results ................................................................. 122
   5.2.1 Quorum Sensing Signal with $P_{\text{notch}}=1.0$ and NO NTP .... 123
   5.2.2 Quorum Sensing Signal with $P_{\text{notch}}=1.0$ ............. 128
   5.2.3 Quorum Sensing Signal with $P_{\text{notch}}=0.5$ ............ 131
   5.2.4 Quorum Sensing Signal with $P_{\text{notch}}=0.25$ ............ 136
   5.2.5 No Quorum Sensing Signal ................................. 141
5.3 Discussion ............................................................. 144
   5.3.1 Experimental Outcomes .................................... 144
5.3.2 Lessons Learned .............................................. 146

6 Emulated Network Study ........................................... 148
  6.1 Design .......................................................... 149
    6.1.1 Design Revisions ......................................... 150
    6.1.2 Environment Setup ....................................... 151
  6.2 Results .......................................................... 159
    6.2.1 FloridaTech Virtual Environment ......................... 159
    6.2.2 Oregon AWS Virtual Environment ......................... 174
    6.2.3 Northern Virginia AWS Virtual Environment ............ 189
    6.2.4 Ireland AWS Virtual Environment ......................... 201
    6.2.5 Tokyo AWS Virtual Environment ......................... 210
  6.3 Discussion ...................................................... 221

7 Potential Applications and Defenses ............................ 224
  7.1 Biologically Inspired Defenses ............................... 224
  7.2 Exploiting Weaknesses in the Architecture .................... 226
  7.3 Applications for Digital Quorum Sensing ..................... 228
  7.4 Applications based on Information Asymmetry ................ 229
    7.4.1 The Magic Memory Management Unit ....................... 230
    7.4.2 Uncertainty in Digital Quorum Sensing ................... 232

8 Conclusions ....................................................... 234

9 List of Publications ............................................. 239
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Handler/Agent C2 Network Information Flow</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Client/Server C2 Network Information Flow</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Peer-to-Peer C2 Network Information Flow</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>HTTP Request/Response with DNS resolution</td>
<td>20</td>
</tr>
<tr>
<td>2.5</td>
<td>HTTP Request/Response with DNS resolution provided by FFSN</td>
<td>21</td>
</tr>
<tr>
<td>2.6</td>
<td>Shannon/Weaver Communication Channel</td>
<td>44</td>
</tr>
<tr>
<td>2.7</td>
<td>Shannon/Weaver Communication Channel with the inclusion of a Covert Channel</td>
<td>46</td>
</tr>
<tr>
<td>3.1</td>
<td>DQS Emitter</td>
<td>74</td>
</tr>
<tr>
<td>3.2</td>
<td>DQS Receptor</td>
<td>75</td>
</tr>
<tr>
<td>3.3</td>
<td>Detailed workflow of the Proposed DQS Receptor</td>
<td>76</td>
</tr>
<tr>
<td>4.1</td>
<td>Design and Analysis Workflow for the Proof of Concept Study</td>
<td>88</td>
</tr>
<tr>
<td>5.1</td>
<td>Design and Analysis Workflow for the Prototype Study</td>
<td>99</td>
</tr>
<tr>
<td>5.2</td>
<td>Normal Transmission Packet Flow through the Linux Kernel</td>
<td>104</td>
</tr>
<tr>
<td>5.3</td>
<td>Implementation of DQS Emitter as a Linux Kernel module</td>
<td>106</td>
</tr>
<tr>
<td>5.4</td>
<td>Implementation of DQS Emitter as a Network Proxy</td>
<td>107</td>
</tr>
<tr>
<td>5.5</td>
<td>Implementation of DQS Emitter as a new Queueing Discipline</td>
<td>109</td>
</tr>
</tbody>
</table>
5.22 Wilcoxon rank-sum statistically significant periods for the Isolated receptor with No quorum signal .......................... 143

6.1 FloridaTech Receptor observed packet counts, with $P_{\text{notch}}=1.0$ ........ 161
6.2 Wilcoxon rank-sum statistically significant periods at FloridaTech with $P_{\text{notch}}=1.0$ ........................................ 162
6.3 FloridaTech Receptor observed packet counts, with $P_{\text{notch}}=0.5$ (Data Set 1) 164
6.4 Wilcoxon rank-sum statistically significant periods at FloridaTech with $P_{\text{notch}}=0.5$ (Data Set 1) ............................. 165
6.5 FloridaTech Receptor observed packet counts, with $P_{\text{notch}}=0.25$ (Data Set 1) .................................................... 167
6.6 Wilcoxon rank-sum statistically significant periods at FloridaTech with $P_{\text{notch}}=0.25$ (Data Set 1) ............................. 169
6.7 FloridaTech Receptor observed packet counts, with $P_{\text{notch}}=0.10$ (Data Set 1) .................................................... 170
6.8 Wilcoxon rank-sum statistically significant periods at FloridaTech with $P_{\text{notch}}=0.10$ (Data Set 1) ............................. 173
6.9 Wilcoxon rank-sum statistically significant periods at FloridaTech with $P_{\text{notch}}=0.10$ (Data Set 2) ............................. 174
6.10 FloridaTech Receptor observed packet counts, with No quorum signal (Data Set 1) ................................................ 175
6.11 Wilcoxon rank-sum statistically significant periods at FloridaTech with No quorum signal (Data Set 1) ............................. 177
6.12 AWS Oregon Receptor observed packet counts, with $P_{\text{notch}}=1.0$ (Data Set 1) ................................................ 179
6.13 Wilcoxon rank-sum statistically significant periods at AWS Oregon with $P_{notch}=1.0$ (Data Set 1) .................................................. 181
6.14 AWS Oregon Receptor observed packet counts, with $P_{notch}=0.5$ (Data Set 1) ............................................................... 182
6.15 Wilcoxon rank-sum statistically significant periods at AWS Oregon with $P_{notch}=0.5$ (Data Set 1) .................................................. 183
6.16 AWS Oregon Receptor observed packet counts, with $P_{notch}=0.25$ (Data Set 1) ............................................................... 185
6.17 Wilcoxon rank-sum statistically significant periods at AWS Oregon with $P_{notch}=0.25$ (Data Set 1) .................................................. 186
6.18 AWS Oregon Receptor observed packet counts, with $P_{notch}=0.10$ (Data Set 1) ............................................................... 188
6.19 AWS Oregon Receptor observed packet counts, with $P_{notch}=0.10$ (Data Set 3) ............................................................... 190
6.20 Wilcoxon rank-sum statistically significant periods at AWS Oregon with $P_{notch}=0.10$ (Data Set 3) .................................................. 191
6.21 AWS Northern Virginia Receptor observed packet counts, with $P_{notch}=1.0$ (Data Set 1) .................................................. 193
6.22 AWS Northern Virginia Receptor observed packet counts, with $P_{notch}=0.25$ (Data Set 1) .................................................. 195
6.23 Wilcoxon rank-sum statistically significant periods at AWS Northern Virginia with $P_{notch}=0.25$ (Data Set 1) .................................................. 197
6.24 AWS Northern Virginia Receptor observed packet counts, with $P_{notch}=0.1$ (Data Set 1) .................................................. 198
6.25 Wilcoxon rank-sum statistically significant periods at AWS Northern Virginia with $P_{notch}=0.10$ (Data Set 1) ................................. 200
6.26 AWS Ireland Receptor observed packet counts, with $P_{notch}=1.0$ (Data Set 1) ................................................................. 203
6.27 AWS Ireland Receptor observed packet counts, with $P_{notch}=0.25$ (Data Set 1) ................................................................. 205
6.28 Wilcoxon rank-sum statistically significant periods at AWS Ireland with $P_{notch}=0.25$ (Data Set 1) ................................................. 206
6.29 AWS Ireland Receptor observed packet counts, with $P_{notch}=0.1$ (Data Set 1) ................................................................. 208
6.30 Wilcoxon rank-sum statistically significant periods at AWS Ireland with $P_{notch}=0.10$ (Data Set 4) ................................................. 209
6.31 AWS Tokyo Receptor observed packet counts, with $P_{notch}=1.0$ (Data Set 1) ................................................................. 212
6.32 AWS Tokyo Receptor observed packet counts, with $P_{notch}=0.25$ (Data Set 1) ................................................................. 213
6.33 Wilcoxon rank-sum statistically significant periods at AWS Tokyo with $P_{notch}=0.25$ (Data Set 1) ................................................. 215
6.34 AWS Tokyo Receptor observed packet counts, with $P_{notch}=0.1$ (Data Set 1) ................................................................. 216
6.35 AWS Tokyo Receptor observed packet counts, with $P_{notch}=0.1$ (Data Set 3) ................................................................. 217
6.36 Wilcoxon rank-sum statistically significant periods at AWS Tokyo with $P_{notch}=0.10$ (Data Set 1) ................................................. 219
6.37 Wilcoxon rank-sum statistically significant periods at AWS Tokyo with 
\[ P_{\text{notch}} = 0.10 \text{ (Data Set 3)} \]
List of Tables
4.1

Comparison of correlation coefficient statistics between an expected
quorum signal and normal network traffic in a WiFi network . . . . . .

4.2

Comparison of multiple quorum signal emitters on a WiFi network
emitting the same quorum sensing signals . . . . . . . . . . . . . . . .

4.3

94

Comparison of multiple quorum signal emitters emitting the same
quorum sensing signal located in a different geographic regions . . . . .

5.1

94

Comparison of multiple emitters with multiple signals on a WiFi
network emitting different quorum sensing signals . . . . . . . . . . . .

4.4

92

96

Friedman ANOVA p-Values results for the Isolated receptor with
Pnotch =1.0 without NTP synchronization . . . . . . . . . . . . . . . . . 125

5.2

Friedman ANOVA p-Values results for the Isolated receptor with Pnotch =1.0128

5.3

Friedman ANOVA p-Values results for the Isolated receptor with Pnotch =0.5133

5.4

Friedman ANOVA p-Values results for the Isolated receptor with
Pnotch =0.25 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 140

5.5

Friedman ANOVA p-Values results for the Isolated receptor with No
quorum signal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 141

6.1

GENI observed time difference histogram . . . . . . . . . . . . . . . . 153

xiv


6.2 Large Network Test Sequence .............................................................. 159
6.3 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=1.0$ ... 160
6.4 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.5$ (Data
Set 1) ........................................................................................................ 163
6.5 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.5$ (Data
Set 2) ........................................................................................................ 163
6.6 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.25$
(Data Set 1) .................................................................................................. 166
6.7 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.25$
(Data Set 2) .................................................................................................. 166
6.8 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.25$
(Data Set 3) .................................................................................................. 168
6.9 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.25$
(Data Set 4) .................................................................................................. 168
6.10 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$
(Data Set 1) .................................................................................................. 171
6.11 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$
(Data Set 2) .................................................................................................. 171
6.12 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$
(Data Set 3) .................................................................................................. 172
6.13 Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$
(Data Set 4) .................................................................................................. 172
6.14 Friedman ANOVA p-Values results at FloridaTech with No quorum
signal (Data Set 1) ....................................................................................... 176
6.15 Friedman ANOVA p-Values results at FloridaTech with No quorum signal (Data Set 2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 176
6.16 Friedman ANOVA p-Values results at AWS Oregon with $P_{\text{notch}}=0.25$ . . 184
6.17 Friedman ANOVA p-Values results at AWS Oregon with $P_{\text{notch}}=0.10$ . . 187
6.18 Friedman ANOVA p-Values results at AWS Northern Virginia with $P_{\text{notch}}=0.25$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 196
6.19 Friedman ANOVA p-Values results at AWS Northern Virginia with $P_{\text{notch}}=0.10$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 199
6.20 Friedman ANOVA p-Values results at AWS Ireland with $P_{\text{notch}}=0.25$ . . 204
6.21 Friedman ANOVA p-Values results at AWS Ireland with $P_{\text{notch}}=0.10$ . . 207
6.22 Friedman ANOVA p-Values results at AWS Tokyo with $P_{\text{notch}}=0.25$ . . 214
6.23 Friedman ANOVA p-Values results at AWS Tokyo with $P_{\text{notch}}=0.10$ . . 218
I would like to acknowledge the many people who have made this dissertation possible. First I would like to sincerely thank my wife, Karen, without whom I would never have been able to complete this work. She has helped and supported me in this endeavour over the years and unfortunately I will never find the right words to be able to convey how grateful I am for her support. I would also like to thank my son, Aurin, who was a constant reminder that there is life beyond research, and sought to remind me of this with his desire to trade Pokémon.

I would also extend my sincere gratitude to my mentor and advisor, Dr. Richard Ford. His advice and guidance throughout this experience have been invaluable to seeing this work to its successful conclusion.

I want to express my thanks to my committee members: Dr. Heather Crawford, Dr. Keith Gallagher and Dr. Richard Turner. Their feedback and comments throughout the entire process helped me to understand the different perspectives required in conducting a project of this size.

I would like to thank Dr. Marco Carvalho, Evan Stoner, and Troy Toggweiler from the Harris Institute for Assured Information, at the Florida Institute of Technology use of network resources while conducting experiments and funding support for the experiments hosted in Amazon’s AWS EC2 service. I would also like to thank
Keith Johnson, also from the School of Computer Sciences, at the Florida Institute of Technology for providing access to the network resources for hosting the prototype emitters.

Finally, I owe a debt of gratitude to the School of Computer Science and the Harris Institute for Assured Information by supporting my studies and funding my trips to several conferences.
Dedication

To Karen
Declaration

I declare that the work in this thesis is solely my own except where attributed and cited to another author. Most of the material in this thesis has been previously published by the author. For a complete list of publications, please refer to Appendix 9 at the end of this thesis.
Chapter 1

Introduction

Software is written for a wide variety of purposes; unfortunately, some software is developed specifically supporting malicious intentions. Malicious software (malware) is not a minor problem, as an entire segment of the software industry has developed for the purpose of combating malware software. As malware authors are constantly seeking to gain access to computational resources and data, there is an escalatory arms race between the malware authors and the anti-malware industry. This conflict has resulted in a dynamic industry as the software ecosystem must adapt to changing malware strategies. The days when malware only displaying a message to the console that the system is infected have long passed. Modern malware can have disastrous impacts on a network such as but not limited to bringing down a network, exfiltrating trade secrets or stealing money.

Early malware was written to demonstrate that a proposed idea was not entirely theoretical [123]. Hobbyists produced early malware to demonstrate their programming skills. Unfortunately, hobbyists are no longer the primary driving force behind malware. Malware is used by cybercriminals for financial gain. More recently another producer
of malware has started to gain notoriety in the news: nation-states are developing and releasing targeted malware.

The threat environment and even computing itself have changed; computers are no longer isolated or in small communities of completely trusted and technically savvy users. The Internet is now a large global community with billions of interconnected systems. User skill levels from technically savvy users to those who require assistance configuring and maintaining their systems are all using the Internet.

The Internet is an environment in which billions of computing devices routinely connect and each of these nodes is a potential resource waiting to be harvested. Despite the target-rich environment the Internet provides, some of the most attractive targets are deliberately isolated from the Internet. Some examples include those maintained by corporations and nations to protect their most valuable secrets or conduct operational missions. A specific type of malware author, the determined adversary [107] is motivated by mission requirements, potentially at the national level, to attack and disrupt these networks. These determined adversaries are often well resourced and motivated to find new and potentially undisclosed techniques to exploit and maintain a presence on these networks [107].

1.1 Contributions

This work explores a novel communication channel which a determined adversary may use to coordinate targeted malware on these high value networks. This work is concerned with a communication channel in which the determined adversary has deemed it acceptable to tolerate a greater level of uncertainty in the targeted malware’s Command and Control (C2) channel. In doing so, the determined adversary is able
to ensure that network defenders are unable to gain actionable intelligence using the C2 channel. Actionable intelligence is necessary for the network defender to properly remediate and recover from a compromise. Instead of incrementally modifying their current approaches, a determined adversary could leverage a new communication scheme by exploiting non-determinism due to the different information requirements of the attacker and defender. Furthermore, the determined adversary could seek communication channels used by natural systems which make use of limited information exchanges and use those channels as a source of inspiration for the development of targeted malware.

There are a number of biologically inspired computing technologies which were developed to allow the defender to enhance its security posture. This work highlights that an attacker using biologically-inspired techniques could leverage paradigms from natural systems to advance their attacks.

The information requirements are different for the attacker and defender. The defender usually requires more information to be successful while the attacker can accomplish their objectives in the presence of uncertainty about the target system’s state. This work will illustrate this by demonstrating that an adversary can exploit the difference in these requirements to its advantage. The determined adversary could make use of the techniques described in this work to allow targeted malware to operate in an environment while preventing the network defender from effectively enumerating compromised hosts and responding to the incident. This work will leverage a self-organizing behavior such as the process utilized by bacteria to coordinate their social behaviors as a proof of concept to a non-deterministic communication channel and explore its effectiveness in different laboratory and emulated network environments.
The goal of this work is not to promote “insecurity” but to enhance security; as such this work attempts to highlight that there is a novel C2 scheme which could be exploited exclusively to the attacker’s benefit. Thus, this work was performed and the results are being released to the security community so that defenses against this attack can be explored by the security community. During the course of conducting this research a weakness was identified in the C2 channel which network defenders could exploit and prevent adversaries from successfully using this C2 channel for coordinating malware.

1.2 Terminology

This work draws upon a number of distinct and seemingly dissimilar fields therefore making definitions necessary. This section will discuss concepts and definitions related to advanced attacks and malware. These definitions will provide the basis for which this work is built upon.

The term Advanced Persistent Threat (APT) is commonly associated with sophisticated malware attacks directed towards specific organizations or even individuals. The term APT originated within the US Department of Defense as a result of dealing with sophisticated intrusions sponsored by nation-states; more recently the term APT has been associated with any attack that targets a specific organization [23]. There are several different definitions for APT. Mandiant [94, 96] defines APT as “a group of sophisticated, determined and coordinated attackers that have been systematically compromising U.S. government and commercial networks.” Damballa has another more comprehensive definition of APT which states that three requirements must be satisfied to qualify as an APT [38]:

4
• **Advanced:** It must have access to a variety of advanced computer intrusion techniques and typically develop novel tools and attacks as needed.

• **Persistent:** It focused on accomplishing a specific task as opposed to seeking financial gain. It may be focused on corporate and political targets and are not simply focused on short term immediate financial gain.

• **Threat:** It exhibits a level of human planning and coordination in the attack. It is not simply an automated script which scan the Internet looking for vulnerable systems.

An important aspect of Damballa’s definition is that these adversaries are not automated scanners that simply attack any and all vulnerable computer networks, they are teams of motivated and resourced individuals.

A determined adversary as defined by Microsoft [107] is an adversary that is not deterred by failure or obstacles and will continue to attack its target until its objectives have been satisfied. Another key feature highlighted in the Microsoft definition is that the adversary is typically guided by a well-funded or resourced organization which is actively guiding mission objectives. The combination of these features has the effect of allowing the adversaries to innovate and apply new intrusion tools and techniques as needed. As a result of the broadening usage of the term APT and to avoid potential confusion about the meaning of APT, this document will use the convention that Microsoft has advocated and use the label determined adversary to describe the organization which is conducting the attacks.

Targeted attacks are attacks created and used for the purpose of compromising a specific target by determined adversaries [107]. When targeted attacks are used, they are executed for the purpose of satisfying a specific mission or objective. These attacks
are typically conducted to gather information by exfiltrating information or disrupting mission operations.

Targeted Malware is malware written specifically to support a targeted attack conducted by a determined adversary. This malware is used in support of missions against specific targets, although the adversary may deploy previously created malware to satisfy the current mission. The malware associated with a targeted attack can be sophisticated such as in the case of the Stuxnet \([87, 50, 31, 51]\), Duqu \([19, 18, 17]\) or Flame \([19]\) malware, but this level of sophistication is not always required. The malware deployed against the target is only as sophisticated as needed to accomplish the mission, indicating that the determined adversary is allocating resources appropriate to the level of difficulty when compromising a target.

Indicators of Compromises (IOCs) are a specific set of observable artifacts which provide evidence that a system has been compromised \([95]\). IOCs also relate to ongoing intrusions to a system. When a system is compromised, the state of the system is changed, and these changes represent artifacts which can be used by network defenders to assess the state of the system. These artifacts are not solely related to the initial compromise. New IOCs will be generated as an attacker maintains a presence on a system. Sample IOCs include the presence of files with specific names or files that match certain cryptographic hashes. They can be the presence of a new service or set of open ports on a system. For example, OpenIOC states that the presence of a file with the name ‘mrxnet.sys’ is an indicator that the system has been compromised by Stuxnet. IOCs provide actionable intelligence to a network defender. They provide evidence that either a compromise has occurred or a compromise is ongoing. This evidence allows the network defender to respond to the threat and declare an incident has occurred.
Covert channels are often used as a method for exchanging information in the presence of a security monitor which is attempting to enforce a security policy. Lampson [86] described and defined a covert channel as an information leakage on a system which occurs “through channels intended for other uses onto which the information is encoded.” This definition is used when a system is being analyzed to determine if a subject is able to pass sensitive information which would violate a security model (e.g. the Bell-La Padula model [15]).

Simmons [124] created a thought experiment to illustrate and define a Subliminal Channel. In Simmons’s thought experiment, two prisoners are confined and are attempting to coordinate their escape. A warden is present and willing to allow the prisoners to communicate either to catch them during their escape or to deceive the prisoners into supplying additional information. Both sides are willing to accept the arrangements in order to communicate. The prisoners need to find a way to deceive the warden and exchange information with seemingly innocuous messages. An important aspect of the subliminal channel is that it includes authentication in addition to a message being exchanged as the warden may maliciously modify the messages as they are being relayed.

The Lampson definition of a covert channel recognizes that two entities are legitimately exchanging messages through a pre-arranged protocol, but additional information is being encoded to the messages. This leaked information is not enhancing or reducing the effectiveness of the initial communication. Lampson’s definition is similar to the one often used in operating system security when systems are subjected to formal verification as what was seen in the Trusted Computer Security Evaluation Criteria (TCSEC) [106] or the Orange Book to provide evidence that the system is secure. TCSEC defines a covert channel as “any communication channel that can be
exploited by a process to transfer information in a manner that violates the systems security policy” [106]. The information being leaked is assumed to be in violation of the local security policy. Simmons’s definition is different in that it explicitly assumes that the monitor or warden may be watching and inspecting the messages that are being exchanged [124].

Gray III[66] highlighted that, although there are a few definitions of a covert channel, these definitions do not offer a distinction which can be used to differentiate between a covert channel and a legitimate communication channel. Gray proposes that there are two characteristics that distinguish a covert channel from a legitimate channel. The first characteristic is that the covert channel is attempting to hide from the network defenders. The second difference is that the “covert channel is capable of violating the intended security policy” [66]. The second distinguishing characteristic, which is unique from other definitions, highlights that a covert channel may only need to be capable of violating the security policy; it is not required that the covert channel act always to violate the security policy.

A communication scheme is being proposed as part of this work, but, as the communication scheme being proposed is not leaking information in violation of a security policy, the TCSEC definition of a covert channel is not the primary definition that this work will rely upon. Within this work a potential covert channel will be discussed based on a self-organizing behavior of bacterial cells. The message being embedded is not being authenticated and provides minimal information about individual senders because the embedded message does not specify a receiver. This work will leverage Lampson’s definition of a covert channel. The proposed covert channel diverges from Gray III’s characteristics of a covert channel in that it is desirable for the channel to remain hidden but it is not necessary for it to remain hidden for it to
remain effective. Also, the proposed covert channel is not attempting to violate the security policy by exchanging sensitive information.

1.3 Organization

This work is organized into the following sections. The background provides an overview of the history of malware, a survey of biologically inspired computing with respect to security (including artificial immune systems) and description of the quorum sensing process that bacteria use to coordinate their social behaviors. The next section describes an architecture that was developed and based on the bacterial quorum sensing process, i.e. the digital quorum sensing process. Following the architecture are the chapters that describe development a proof of concept and a prototype which was used as part of a series of proposed studies to determine the effectiveness and possible limits on the effectiveness of this communication channel. The next chapter describe a re-implementations of the quorum sensing process which allows the communication channel to make use of existing network traffic. The revised design was used in a set of experiments in an isolated network. Another set of experiments were performed with the quorum sensing process in which the components were geographically distributed. A chapter was devoted to examining countermeasures and potential mitigation strategies which could be used to disrupt digital quorum sensing. Finally a conclusion is included which summarizes the goals and potential contributions of this work.
Chapter 2

Background

The research addressed in this work leverages a number of distinct fields of study. This section provides background and literature review of the various fields utilized as part of this work. There are six different major fields that are discussed in this section: malware, biologically inspired computing, bacterial quorum sensing, determinism vs non-determinism, covert channels, digital quorum sensing, and the ethics associated with security research.

The first topic discussed within the background section is malware. It includes a discussion on malware with respect to the ideas of self-replicating software, malicious mobile code (MMC) and malware Command and Control (C2). Following a discussion of C2 channels, a discussion on the use of targeted malware, or malware written by nation-states to accomplish tactical and strategic objectives, is provided. The author refers the reader to Szor’s The Art of Virus Research and Defense for a more complete history and discussion of malware in general [131].

Another topic included within this section, is biologically inspired computing. The discussion in this section is restricted to biologically inspired computing as it applies
to security. It emphasizes how biologically-inspired computing is focused on computer and network defense and highlights that determined adversaries could leverage this as a fruitful area of research.

After discussing biologically-inspired computing, a discussion of bacterial quorum sensing is presented. Bacterial quorum sensing is one of the primary areas that this work uses as a source of inspiration to demonstrate that determined adversaries could leverage to gain a potentially disruptive C2 channel for targeted malware. Bacterial quorum sensing is interesting in that pathogenic bacteria can coordinate the expression of social behaviors to attack and overwhelm a host’s immune system. A novel aspect of this communication is that the coordination relies entirely on a non-deterministic information population count. As such, a discussion of determinism vs. non-determinism is included following the discussion of biologically inspired computing.

The next topic covered in this section is a literature review of covert channels. With computers and networks that handle sensitive information, a security policy is established to allow and restrict access to information. Not all users who have access to information are interested in following these polices. Malicious insiders may seek to abuse existing mechanisms and exchange information which violates the implemented security policy and leak information to entities which should not have access to this information. A number of different studies have been performed since Lampson’s original work [86] to understand, characterize and defend against covert channels.

Using bacterial quorum sensing as a mechanism for communication and coordination for potentially malicious purposes is neither new nor novel, and a discussion of Digital Quorum Sensing is included that summarizes the work that has previously been performed in this area. For example, Vogt et al.[137]; proposed that bacterial quorum sensing could be used as an algorithm to control the propagation of
a self-stopping worm; this section discusses Vogt’s work in relation to this proposal as well as other work which is relevant to this research.

Finally a discussion of the ethics involved in security vulnerability research is presented. At first glance this work could be interpreted as substantially aiding a determined adversary. This section addresses the possible perception that this work could be construed to entirely support determined adversaries and provides arguments to counter this perception and demonstrate that the work benefits the entire security community and not just the attacker. It is shown that this work conforms to best practices for ethical research in information security.

2.1 Malware

The idea of self-replicating software has existed for a long time. The scientist von Neumann proposed the idea that artificial entities similar to organic life could be constructed such as to allow for self-replication [138]. When von Neumann proposed self-replicating automata his discussion was concerned with physical machines that read a tape which contains instructions that allow the automata to replicate. The automata was based on a machine which had multiple states and the instructions on the tape caused the automata to transition between the states. The instructions on the tape allowed for self-replication, as the complete process concluded with the result of producing another instance of the automata. The new instance of the automata was complete in that it was also capable of self-replication.

Although von Neumann’s self-replicator did not mention software, Thomas used von Neumann’s self-replication ideas and applied them to software which resulted in the creation of the Creeper worm [123]. The Creeper worm was a benign form of self-
replicating software. Despite Creeper being benign in nature, another worm, named Reaper, was written to remove Creeper from infected systems.

Not all malicious software was designed to replicate. Trojan horses are a class of malware in which software is provided that has a stated purpose but includes functionality which is malicious in nature. If inclusion of this functionality is deliberate, it results in the application being classified as a Trojan horse [133]. Furthermore, Thompson demonstrated that even though the source code of a program contained no malicious functionality, the compiler could be modified such that malicious functionality was included at compile-time.

The term “computer virus” was not widely used until it was proposed by Cohen as part of his PhD dissertation [35]. Richard Dawkins’s experiments on artificial life in a simulated environment [39] were used as a source of inspiration for Cohen. Cohen’s work provides a theoretical framework which demonstrated how malware could be written. Beyond providing a formalized framework to describe malware, Cohen’s work discussed different methods that can be used to identify malware and highlighted that the malware detection is not a trivial problem.

2.1.1 From Theory to Practice

Malware is not simply a theoretical problem. Two significant pieces of early malware are the Morris worm and the Brain virus. The Morris worm is an example of malicious mobile code (MMC) and was released into the Internet in 1988 [108]. The Morris worm was a self-replicating piece of software which infected 5% of the computers connected to the Internet. The worm exploited a system and then attempted to spread itself to other systems. The Brain virus is cited as being the first virus which infected the IBM
personal-computer (PC) platform in 1986 [131], which has become one of the dominant computing platforms.

Malware authors began to leverage other methods for expanding to new hosts based on the documents that were being shared between users. The macro viruses Concept and ILOVEYOU used Microsoft Visual Basic for Applications (VBA) scripts to infect and spread between Microsoft Word and Excel documents [131]. The Concept virus was released in 1996 and attempted to infect the document template on a host to ensure that all documents created or opened were similarly infected. In 2000, the ILOVEYOU Visual Basic Script (VBS) macro virus infected hosts by enticing the user to open a “Love Letter” which in reality was a script. The virus did not carry a destructive payload but rather emailed copies of itself to each entry in the users personal address book.

Between 2001 and 2003, malware returned to following the pattern of the Morris Worm and spread over the Internet by exploiting host vulnerabilities. The Sadmind worm was a self-replicating piece of software that affected vulnerabilities in both Solaris and Microsoft Windows. It was discovered on the Internet (i.e. in the “wild”) during May of 2001. Sadmind was novel in that it contained exploits for operating on two different platforms within the same worm. The worm contained two exploits; one exploited a vulnerability in the Solaris Operating System, while the second exploited a vulnerability in Microsoft Internet Information Services (IIS). After a Solaris system was infected, the worm would scan the local network to look for vulnerable Microsoft Windows hosts. If a vulnerable IIS server was discovered, the worm would exploit the server and change the content hosted.

Later in 2001, the Code Red worm was released. Code Red exploited Microsoft Windows hosts which were running a vulnerable version of the Index Service associated with Microsoft IIS [149, 101, 129]. The worm was able to successfully exploit a buffer
overflow vulnerability in the Indexing Service, despite a patch released by Microsoft over a month before. Code Red II was released about a month after Code Red and relied upon the same vulnerability, but a different malicious payload was included [131], [129]. While Code Red relied upon a single exploit, another worm, Nimda, released in September of 2001, relied upon five different infection vectors. Nimda’s infection vectors consisted of spreading from infected hosts via email with infected attachments, copying to network shares, adding drive-by exploit code to web pages, exploiting directory traversal vulnerabilities in Microsoft IIS and even making use of backdoors created by Sadmind and Code Red II [129].

In 2003, Microsoft SQL Server was exploited by the SQL Slammer worm [85, 113]. SQL Slammer was one of the fastest spreading worms on the Internet. Later in 2004, the Witty worm was released [120]. These worms and other security vulnerabilities caused Microsoft to form the Trustworthy Computing Initiative [90] [103]. Code Red, Nimda, SQL Slammer, and Witty rapidly spread over the Internet and found a large number of infected hosts. As the malware spread, their impact on Internet traffic was discernible and their impact on the Internet could be detected [85] [32]. In 2007 and 2009, worms such as Storm [35] and Conficker [110, 121, 122] demonstrate that malware is still a significant threat to the security of the Internet.

2.1.2 Command and Control

A key difference between modern malware and early self-replicating malware is that the malware is no longer attempting to simply deface web sites. It is more common for modern malware to attempt to establish resources which can be used to form a network of infected clients which is often referred to as a “BotNet”. Cybercriminals are
interested in leveraging compromised resources for financial gain, and, if a computer is compromised and it is taken offline, it is no longer a usable resource. Compromised computers are collected into networks are referred to as BotNets. BotNet command and control (C2) allows the BotNet herder to command the BotNet to perform various malicious actions such as harvesting credentials or performing a Distributed Denial of Service (DDoS) against a third party.

Multiple C2 schemes have been developed by malware authors, as a result BotNet herders do not directly communicate with the infected hosts. In doing so, it allows a network defender to trivially identify the herder. BotNets have adapted in an attempt to evade detection and provide resilient C2, a number of different types of C2 channels have developed. The methods used to control a BotNet can be divided into four different categories: handler/agents relationships [45], centralized C2 such as a client/server architecture [45], distributed resilient Peer-to-Peer (P2P) networks [45] and hybrid Fast Flux networks [78], [140]. These four different C2 channels will be discussed in more detail in the remainder of this section.

**Handler/Agent Networks**

Handler/Agent networks are simple networks of small numbers of infected hosts [45]. As illustrated in Figure 2.1, each of the infected hosts has a list of handlers which provide a source of updated instructions. An infected host often connects to three or more handers to provide resistance to handler takedowns. By listening for commands from multiple handlers, the BotNet can continue to operate while handlers are identified and removed from the network by network defenders. Although the network has some robustness, it is limited in that if all of the handlers are removed, the agents can no longer receive any commands.
Client/Server Networks

Another important C2 channel for BotNets is the client/server architecture, which is illustrated in Figure 2.2. The BotNet Herder relies on a number of hosts to act as servers to which all of the infected clients connect. This architecture is different from the Handler/Agent architecture in that this communication typically relies on an established Internet service, such as Internet Relay Chat (IRC). IRC is commonly used as there are a number of libraries that can be used to trivially integrate this C2 channel into the Bot. IRC services make available a number of servers to which clients can connect and discuss any number of topics. Clients connect to a server and post a message. The message is transmitted to the IRC server, which exchanges the message with other IRC servers. These remote IRC servers in turn pass the message to all of the clients, who are subscribed to the same topic. An IRC topic is a string which is used to segregate IRC traffic into manageable volumes by ensuring that messages are only passed to a particular subset of IRC clients.
Malware can make use of this or a similar infrastructure by having infected clients connect to a topic where they can wait for a command. Fortunately, IRC server managers look for signs that IRC topics are being used to manage BotNets and terminate the channels once the malicious activity is identified [45].

Peer-to-Peer Networks

P2P networks are commonly used for C2 of highly sophisticated and resilient malware networks. In the case of Handler/Agent networks or Client/Server networks, if the handlers or servers are taken down by network defenders, the Bot herder is unable to task their BotNet. P2P solves this vulnerability by providing multiple paths through which the handler or agent tasks the BotNet. In P2P C2 networks, there is no central node (or a small set of nodes) through which all of the C2 traffic passes. Agobot/Phatbot,
Storm, and Nugache are examples of malware that rely on a P2P network for C2 [45], [46].

Once a host is infected, the malware begins scanning for other hosts which are also infected. When another host is identified, this new host is considered to be a peer. When the infected host receives a command from a Bot Herder, the message is passed along to all peers. Each time the message is passed, a Time-To-Live (TTL) counter is decremented; and when the TTL reaches zero, the message is no longer passed along. This ensures that the message is passed to all peers, but it also ensures that the message is not repeatedly passed through the entire network as some peers could be indirectly connected to each other (e.g., there are loops in the connection graph). The messages are encrypted and signed to ensure the confidentiality and integrity of the messages are not compromised and originate from the BotHerder. This type of C2 is illustrated in Figure 2.3.
Fast Flux Service Networks

In an effort to enhance the robustness of BotNet C2 functionality further, Fast Flux service networks (FFSNs) were created. A FFSN contains features from both P2P networks and the Content Distribution Networks (CDNs) such as those provided by Akamai [78]. CDNs work by allowing a single Domain Name Service (DNS) host name to resolve to multiple IP addresses. These addresses can be located so they are geographically close to the host or allow the host to request information from a secondary address in the case that the primary address does not respond with a low enough network latency.

When a user submits a request via their web browser, the URL must be translated to an IP address which relies upon DNS. Figure 2.4 illustrates the normal process for a Hypertext Transfer Protocol (HTTP) request. In Step (1), the client looks up the domain name by submitting a request to a DNS server (the process in the illustration is simplified as DNS recursion and caching is excluded). The DNS server returns a response that contains an IP address where the www.example.com domain can be found, as part of
Step (2). The client is then able to submit a HTTP GET request to the IP address as part of step (3). In step (4), The HTTP Server processes the request and generates a HTTP response which includes the requested content.

Figure 2.5 provides an illustration of a FFSN providing content to a client. When compared the process associated with a normal HTTP GET request/response, additional nodes have been introduced and the process becomes more complex. Individual steps in the process of fulfilling the HTTP GET request are as follows;

**Step 1:** The user or client submits a HTTP request. The client system begins with a DNS lookup on the associated Fully Qualified Domain Name (FQDN).

**Step 2:** The client submits a DNS query for flux.botnet.com to a DNS server. Similar to the process illustrated in Figure 2.5, the entire process of resolving a DNS request has been simplified for the purposes of illustration.

**Step 3:** The DNS server is not the authority for the botnet.com domain, so the DNS response consists of a referral to the ns.botnet.com domain.
Step 4: A request for flux.botnet.com is submitted to the DNS server ns.botnet.com. The DNS server for ns.botnet.com an infected node which is part of the BotNet, and not a traditional DNS server.

Step 5: The ns.botnet.com infected node does not cache the DNS records, so it submits a request for DNS lookup for the address to redirect the client to a mothership.

Step 6: The mothership determines which other infected node in part of the BotNet will act as the HTTP server for the client (for example 10.10.10.10). A response containing the IP address of the infected node is returned to ns.botnet.com.

Step 7: The ns.botnet.com infected node returns a DNS response to the client informing the client that the flux.botnet.com server can be found at 10.10.10.10.

Step 8: The client submits a HTTP GET request to the 10.10.10.10 address for the flux.botnet.com host.

Step 9: The content “server” at 10.10.10.10 is another infected node within the BotNet and does not contain the content. In this situation, the content server is acting a blind proxy relay. A content request is submitted to the BotNet content provider which is hosting the desired content.

Step 10: The content provider returns the requested content to 10.10.10.10.

Step 11: The HTTP response to the HTTP GET request is returned to the client.

One of the main purposes of DNS is to translate domain names into IP addresses. A feature of this translation is the DNS server allows the Domain record to specify a TTL. The TTL included within the DNS response specifies how long before the client should request a new record as it may change in response to an address update. It would
be reasonable to expect that a DNS record would be good for between 1 and 3 days as it is expensive to change hosts or hosting providers, but with a FFSN the shorting of TTLs periods is what happens. TTLs for DNS records associated with FFSN have been observed with times less than 15 minutes [104] and some on the order of 3 minutes [112]. In a FFSN, when the next DNS request is submitted the IP addresses associated with the FQDN have changed, so the Bot is communicating with a new set of servers. The servers are actually blind proxy relays which handle the URL request. In essence, these blind proxy relays are intermediaries which relay content from another server that is the content provider.

Although the process of providing content to clients is handled via a separate channel from the C2 of the FFSN BotNet, the functionality is used to provide C2 throughout the BotNet. This functionality is separate from the C2 channel of the BotNet, but it similar mechanisms are used for C2. The C2 serves for a FFSN are referred to as Fast Flux motherships and the commands are relayed to the clients through blind proxies so the clients never actually directly communicate with the Fast Flux mothership. This level of obfuscation provides a defense against takedowns as the C2 messages and content are not passing through the same routes and are constantly changing.

**C2 Network Classification**

Dittrich classified three of these types of networks with regards to their C2 method. Hander/Agent is classified as direct, as there is a direct C2 channel between the entity issuing the commands and the clients acting upon the commands. Client/server and P2P architectures were classified as indirect [45] as the infected nodes never directly interact with the entity issuing the commands. The entity issuing the commands over the C2 channel is not directly connected to the client. In the case of the IRC client/server, client
to client communication passed through at least one intermediary IRC server. Likewise with P2P, the command is not issued directly from the Bot Herder to the clients. The message passes from client to client so it traverses the BotNet. The messages may have passed through a large number of peers before arriving at a specific host. Although Dittrich did not classify FFSNs in their work, it is clear that FFSNs belong to the indirect category.

Although the communication of some C2 channels is indirect and even rapidly changing (in the case of FFSN), the communication is deterministic. Although it may be time consuming or resource intensive, it is computationally feasible to enumerate the infected hosts of the network by monitoring the IP addresses included in the DNS response for a FQDN. By tracking the communication further, a network defender could determine the location (e.g., IP address) of the content provider with enough effort. This determinism is an exploitable weakness that the network defender can leverage against the C2 channels used by malware.

### 2.1.3 Targeted Malware

Not all malware seeks to infect every host available. Some malware such as targeted malware seeks to infect a subset of susceptible hosts; i.e., those hosts which are part of the mission objective. Several different sophisticated attacks have been recorded, analyzed, and reported to the larger security community. Types of malware include GhostNet in 2009 [41], [136], Stuxnet in 2010 [50, 97], [97], Duqu in 2011 [19, 18, 17, 132], and Flame in 2012 [16]. Targeted malware continues to be discovered and disclosed. This section will detail some of the recent reports on targeted malware but is mainly focused on the mission and C2 aspects of the targeted malware.
In 2009, Deibert and Rohozinski released a report on GhostNet [41], an operation that was conducted against the Tibetan community. The nature of the operation was to collect information about the Dalai Lama. Based on information within the reports regarding GhostNet, the malware was the GhostNet Remote Access Trojan (RAT), which used a C2 channel based on the Handler/Agent pattern. Shortly after the report regarding the discovery of the GhostNet was released, the IP addresses associated with the handler nodes were shutdown [136].

Not all targeted malware is focused on collecting intelligence and exfiltrating information. Some targeted malware, such as Stuxnet which was publicly disclosed in 2011, is focused on disrupting operations. Stuxnet attacked a specific model of the Programmable Logic Controllers (PLCs) found within an Industrial Control Systems (ICSs) in such a way that the attack was actively being hidden from the controller [50, 97]. Falliere’s report on Stuxnet documents the sophisticated nature of Stuxnet and the multiple methods that the worm used to mask its presence on the infected system [50]. The worm relied upon several different methods for spreading including exploiting four unpatched Microsoft Windows vulnerabilities as well as relying on two zero-day exploits.

The C2 infrastructure supporting Stuxnet is a hybrid of the Handler/Agent and P2P patterns. Stuxnet contacts external servers for C2 which allows the handler to deliver and execute payloads via HTTP. On an isolated network, it appears that the determined adversary recognized that all of the infected nodes may not be able to contact the Handler, so the Stuxnet worm also relied on a P2P channel to share updates [50].

Duqu was disclosed in 2011 and was similar in capabilities to Stuxnet but operated with a different mission [19, 18, 17, 132]. Bencsasth et al. and Szor highlighted there were a number of similarities between Duqu and Stuxnet which included: the
modular architecture, use of fraudulent digital certificates and similar DLL injection mechanisms for infection and maintaining persistence on the system [19, 18, 17]. Instead of manipulating the logic for a specific set of ISC PLCs, Duqu was written to support intelligence collection and exfiltration of information. Duqu’s C2 is similar to Stuxnet, but is it more advanced and can operate in a distributed manner. The C2 traffic for the Duqu malware occurred over HTTP and HTTPS in a Handler/Agent model, but it can also operate in a P2P model in which the malware receives commands from infected peers. In the case of the Handlers, it appears that the servers were compromised by the attackers supporting Duqu, as the Handlers were configured to act as relays and passed C2 messages to multiple servers [33].

The targeted malware known as Flame or Skywiper was publicly disclosed in 2012; however it is estimated to have been active between 5 and 8 years prior to its discovery [16]. The Flame family represents another series of targeted malware that collects and exfiltrates information, but despite the similar mission profile it is not thought that Flame and Duqu were developed by the same determined adversary [16]. Flame seeks to collect information in a number of different ways from an infected host; it features keylogging, screenshots can be taken periodically, and even microphones (if present) can be activated and used for recording [19]. The C2 channel for Flame relied on techniques that BotNets have used for FFSNs with HTTP requests submitted to one of several different domains. More than 80 domains associated with 15 IP addresses were recorded during the initial analysis of Flame [16], [139].

**Targeted Malware’s Problem**

This concept can be stated as a problem in which the determined adversary must be overcome. The targeted malware may be operating in an environment in which the
network architecture and system defenses are unknown. The determined adversary may have some insight into the available configuration or network resources, even if only limited information is available. The network is subjected to intensive host and network monitoring; however it is assumed that the determined adversary has the resources, tools, and techniques to bypass host-based defenses.

In this work it is assumed that the primary objective for the determined adversary is to disrupt a target’s mission as opposed to exfiltrating information. The targeted malware must communicate to coordinate its activities, but, it is not able to receive external C2 from handlers as the network is either physically or logically isolated from outside networks. The use of traditional C2 is a not an option due to the level of network monitoring as the target network is a high value network supporting critical missions, so the determined adversary will seek novel C2 channels. As the malware is isolated it will need to leverage self-organizing behaviors to determine when it is appropriate to disrupt the mission or capability.

In summary, the targeted malware’s problem is defined as the following: “malware is operating in an unknown potentially hostile environment in which it must communicate to coordinate its activity, but the act of communicating can be used as a signal that something malicious is present to network defenders.” There are many different potential solutions to this problem, but the solution that is of most relevance to this work is when the determined adversary seeks inspiration from natural systems.

### 2.2 Biologically Inspired Computing

Biological systems have been used as a source for inspiration to solve general problems in computing. Greensmith et al. have highlighted that a biological system as a source
of inspiration is potentially desirable as biological systems are seen as having key properties which researchers are interested in leveraging [68] for security; they can continue to function in the presence of errors or conflicting signals, they are error-tolerant, they can adapt and respond to changing environments and stimuli and they are decentralized and can respond independently [68, 127, 28].

The idea of malware behaving like a biological “virus” as termed by Cohen [35] continues the idea that malware can be addressed in terms of immunology. Significant research has been conducted where intrusion detection systems and computer defenses can be based on the human immune system.

The construction of a Computer Immune System or Artificial Immune System (AIS) has been proposed multiple times as the vertebrate immune system offers a set of desirable characteristics which computer lack. Computers are seen as fragile and unreliable, while the vertebrate immune system is seen as adaptive, resilient, redundant, not reliant on trusted components, and self-healing [28, 77, 76, 127]. Somayaji noted that there are three key assumptions which are relied upon for building a secure computer but in practice these are unobtainable. These are: security policies can be completely and accurately defined, software can be correctly implemented, and systems can be properly configured [127]. Dynamic systems are more difficult to characterize and cannot be realistically evaluated through formal methods. Use of adaptive models such as a Computer Immune System were proposed [127]. Cfengine was an example of a software system which attempted to determine the current system state and alter its own program in such a way to return to a desired state. Cfengine was not written with security in mind but it can have applications to security if the desired security state could be formally defined [28].
The immune system is able to recognize cells which are part of the body from those which are foreign such as viruses and bacteria. In essence, this aspect is the ability to recognize self from nonself. Cells which are part of the body are determined to be self, and foreign cells and particles are determined to be nonself. Forrest et al., proposed that this ability to distinguish between self and nonself could be used as a model for computer security [56, 57, 75] in which self as defined to include authorized users and legitimate processes while nonself was defined to include unauthorized users and illegitimate process (such as those created by malware). Some of the key features of the system include: that multiple sets of different detection signatures and algorithms were used, detection is probabilistic in nature, and the system was designed to be robust. By allowing multiple algorithms to be used, it ensured that if the system was bypassed on one system, it would be detected on another system as different checks were performed. As multiple methods were used to determine if the entity was part of self or nonself, the detection was probabilistic in nature; the same criteria were not used constantly. It was demonstrated that these systems could be used for anomaly detection as part of an Intrusion Detection System (IDS) [57].

Another key concept from the immune system which could be applied to computer security is clonal selection and negative selection [56]. Kim and Bentley extended the work of Forrest et al [56] to apply the negative selection algorithm to a Network Intrusion Detection System (NIDS)[83, 20]. Negative selection is used with self-nonself determinations and works as part of the clonal selection process. The negative selection process works by generating a set of random strings, which are compared with “self” strings. If the correlation between the random strings and the “self” strings is significant, the random string will trigger on “self” processes and is rejected. Those random strings which do not trigger a correlation with the “self” strings are retained and included as
part of the possible detector set of strings. The detector strings are then used to identify potential malicious strings. In theory, it is safe to do so since the “self” string will not trigger a reaction. As the detection set can result in a large number of strings, clonal selection picks up at the end of the process and provides a selection process which is used to refine the detection set. The strings within the detection set are mutated and the mutants are compared against known bad or malicious strings. Those strings which have the highest correlation or affinity to the known bad or malicious strings are kept and used as the next starting point for potential mutations. At the same time the detector set is also compared against “self” strings to ensure that the resulting detection set does not trigger on “self”. This cyclical process of refinement is referred to as clonal selection and expansion.

The human immune system is not limited to the principles of self vs. nonself, or the use of clonal selection and negative selection. Another important aspect of the adaptive immune system is known as danger theory. Danger theory was proposed by Matzinger [98, 99] and is based on detecting threats to the host when cells are damaged or die in ways that are associated with illness or injury; i.e., apoptosis vs. necrosis. Natural cell death is a coordinated and controlled process in which an individual cell destroys itself, referred to as apoptosis. When cells die via this process, a response from the immune system is not triggered. Interestingly, the host has symbiotic relationships with bacterial communities which are recognized as nonself, yet a response from the immune system is not triggered. One of the key differences is how cells die. In necrosis when the cell opens and the contents are released it is thought that a series of danger signals are released in which a response from the immune system is triggered. This process is referred to as Danger Theory [98, 99] and has formed the basis of more recent research in AISs [8, 9, 67].
Despite the research and advances in biologically inspired computing with regards to AISs, there is still disagreement about the utility of biologically inspired security. There are four different points of view: the community has learned everything that should be learned or the community is aware of what it needs, the community has only begin to understand everything that is possible, or the community has leveraged this idea too heavily or it has hampered progress by wasting resources that could be spent elsewhere [128]. Regardless of the position that is taken, most of the research that has been performed in the area of biologically inspired security is focused on defensive technologies. There is nothing to prevent a determined adversary from seeking to leverage the features of a biological system and using these systems as a source of inspiration to further their attacks.

2.3 Bacterial Quorum Sensing

Pathogenic bacteria face significant challenges; these single celled organisms are often attempting to harvest resources from an actively hostile environment. Pathogenic bacteria have demonstrated a number of different strategies to cope with a host’s immune system: One of these strategies is to exhibit multiple different behaviors [105]. When a pathogenic bacterial cell is alone, it can persist in the environment and behave in a way that limits its interaction with the environment. These asocial strategies allow the bacterial cell to survive when it is isolated and alone. The bacterial cell can acquire resources and slowly multiply in order to increase its population. When the population is sufficient, the bacterial cell can switch from asocial to social strategies and attack the host. A single pathogenic cell is not a significant threat to the host’s immune system. The host’s immune system can identify and clear a single pathogenic cell before it
can cause significant harm. A community of pathogenic bacterial cells represents a
different and more significant threat to the host; by acting in concert the community can
overwhelm the host’s immune system.

Pathogenic bacteria can accomplish this transition by utilizing a process which
allows the cell to measure the Local Population Density (LPD) of the bacterial cells
and alter their behaviors accordingly. At Low Cell Densities (LCD) the pathogenic
cells’ population is insufficient to allow the cells to act as a community or adopt their
social strategies, so these cells behave as if there are no other cells around and pursue
their asocial strategies. At High Cell Densities (HCD), the density of pathogenic cells
is sufficient to allow the cells to adopt social strategies which causes the bacteria to
behave as though they are a community. The use of multiple strategies allows the cells
to alter their behavior according to the strategies which provide the most benefit to the
pathogenic cells [105]. An example of a few different strategies including: the initiation
of bioluminescence [105], pathogenesis and production of virulence compounds [105],
sporulation [105], formation of a biofilm and uptake of extracellular DNA via bacterial
competence [105, 40].

Quorum Sensing is the process that bacterial cells use to measure their LPD
and coordinate their transition from LCD to HCD states. There are typically three
components in the bacterial quorum sensing systems; autoinducer synthase, a receptor
and an autoinducer. The Autoinducer synthase is the intracellular machinery which
produces the AI compounds, although the literature may also refer to autoinducers as an
Auto-Inducing Peptide (AIP) when referencing specific quorum sensing systems such as
those associated with *Staphylococcus aureus* [105]. The autoinducer is either produced
and diffuses out of the cell or is produced at the cell membrane and released directly
into the environment. Autoinducers are the signal and these are the compounds that are
measured to determine the LPD. Autoinducers are small light-weight molecules so their production likely does not have a significant impact on the cell’s metabolism. They do not interact with the host in a way that triggers an immune response. The final component is the receptor which responds to the presence of the AI. The receptor measures the concentration of the autoinducer. When the autoinducer’s concentration crosses a critical threshold the receptor triggers gene expression in the cell. The activation of the new genes alters the behavior of the cell such as the initiation of social behaviors (HCD). This causes an autoinduction feedback loop [40, 105] and forces a community-wide transition to the HCD state.

There are some variations on the quorum sensing systems used by bacteria. The process previously described is commonly used by *Vibrio fischeri* when it creates bioluminescence [105]. Another type of quorum sensing system is used by *Vibrio harveyi*, which is not a simple two state LCD and HCD system instead intermediate cell densities are measured by having three different states. Utilization of multiple autoinducers within the same quorum sensing system allows the bacteria to perform a preliminary population estimate before switching on cellular machinery which is more expensive.

Another facet of the autoinducer used in bacterial quorum sensing systems is that the AI contains multiple messages. Each species of bacteria typically releases a specific type of autoinducer, but the autoinducer molecules have two parts; a part which is species specific and another part which is interspecific. This arrangement has the effect of allowing the bacterial cell to monitor not only the total bacterial cell population but also the portion of that population that is likely of the same species. If a bacteria cell belongs to the dominant species in a community, it may express a set of genes that is different from when it is in the minority community. Depending on the population and
the portion of the population, the bacterium may further alter its expressed behavior within the community. *Burkholderia cepacia* [65] and *Pseudomonas aeruginosa* have been found to use quorum sensing to control virulence factor production in the lungs of individuals infected with cystic fibrosis. When both types of bacteria are present there is a dramatic increase in the amount of virulence factors that are produced as compared to when only a single species of bacteria is present [40].

Not all bacteria participate in a single quorum sensing system. Some bacteria such as *P. aeruginosa* participate in multiple quorum sensing systems by exchanging multiple sets of autoinducers. *P. aeruginosa* participates in at least two quorum sensing systems [40] and *S. aureus* [81, 105] which participates in four quorum sensing systems. These quorum sensing systems have been linked to the production of biofilms, the release of virulence factors, and the initiation of pathogenesis. An interesting aspect of the participation in multiple quorum sensing systems associated with *P. aeruginosa* is that the two systems do not activate the same set of bacterial genes. There is some interaction between the quorum sensing systems, but it appears that one of the quorum sensing systems has precedence when activating and will modulate the gene expression of the second system. In other words, the presence of a second autoinducer will control the expression of the genes associated with the first autoinducer. This situation can have the effect of modifying the bacterial cells’ behavior in the presence of diverse bacterial communities.

Some bacteria do not participate in the quorum sensing systems by producing and releasing autoinducers into the environment. Instead some bacteria have adopted a more passive strategy. These receptor-only bacteria monitor the presence of the autoinducers and alter their strategies based on the presence of autoinducers produced by other bacterial cells. These cells are essentially listening in on the messages being exchanged.
by the community; and when the community transitions to a HCD, these receptor-only bacteria can also transition to their own HCD state [115] [44].

As previously outlined, bacteria use quorum sensing to switch between multiple strategies which allow them to alter their behavior depending upon the presence of a community. Bioluminescence associated with \textit{V. fischeri} and \textit{V. harveyi} was the first type of quorum sensing system to be described and some of the most thoroughly studied [40], [114], but bacteria use quorum sensing for a number of other purposes including: initiation of the production of virulence factors, transition and initiation of pathogenesis, beginning the production of biofilms, initiation of competence, and transition to dispersal related states such as sporulation [40].

Quorum sensing is used by \textit{Pseudomonas aeruginosa} [40, 100], \textit{S. aureus} [105], \textit{Vibrio cholerae} [105] and \textit{Escherichia coli} [40] to control the production of virulence factors. Virulence factors are compounds which allow a bacterial cell to facilitate attachment (allow the bacterium to attach to a host cell), immunoevasion (prevent the host’s immune system from responding by allowing the bacterial cell to pass undetected), immunosuppression (prevent the host’s immune system from clearing the pathogen by preventing the immune system from effectively responding to the bacterium), cellular entry/exit (allow the bacteria cell to enter or exit a host cell), or obtain resources from the host by degrading or destroying host cells. \textit{P. aeruginosa} is classified as an opportunistic pathogen and is a source of concern with individuals who have compromised immune systems or individuals who have suffered from severe burns. \textit{P. aeruginosa} also is a concern for healthy individuals who have received medical implants [40]. \textit{V. cholerae} is the causative agent of the human disease cholera, and it also uses quorum sensing to control the production and release of virulence factors.
Quorum sensing is used by some bacteria to initiate pathogenesis. Pathogenesis is different from virulence in that pathogenesis is the transition of an organism into a state in which it causes disease while virulence is a measure of the interaction between the pathogen and host. *P. aeruginosa* [40], *S. aureus* [81, 105] and *Streptococcus pneumoniae* [105] are examples of types of bacteria that use quorum sensing to control pathogenesis. *P. aeruginosa* is capable of causing disease in immunocompromised individuals, while *S. aureus*, which is commonly found to be resistant to antibiotics, is a continuing concern to individuals who are being treated at hospitals and other healthcare provider facilities.

Beyond simply using quorum sensing as a method for initiating virulence factor production or initiating pathogenesis, bacteria also use quorum sensing as a method for determining when biofilms should be produced. Biofilms are chemicals that bacteria release to allow the cells to adhere to a surface or form enclosures to protect the cell from a potentially hostile environment. Biofilms are a source of problems for the medical community [40, 105, 71]. While bacterial cells are enclosed in a biofilm these cells cannot be easily cleared by the immune system. Antibiotics are unable to permeate the structure of the biofilm. This allows bacterial cells to continue to multiply and harvest resources from the environment. *P. aeruginosa* [40], *S. aureus*, and *V. cholerae* [105], [71] have been shown to use quorum sensing to coordinate the production of biofilms.

Pathogenic bacterial cells have a small genome and evolution through reduction is a result of a selective agent that allows these cells to minimize their genome. Smaller genomes can result in the possibility of quicker reproduction. With the loss of each gene, it represents a potential reduction in adaptability to future uncertain environments. A response to evolution through reduction is to make use of horizontal gene transfer (HGT) (also referenced in the literature as lateral gene transfer [LGT]) between bacterial cells.
via intracellular plasmids. This process is called competence. Plasmids contain DNA which can be used by the bacterial cell to alter the cells behavior and promote a wide variety of responses. These plasmids often confer the ability to resist antibiotics, survive in toxic or high temperatures, and degrade and metabolize toxic compounds. Bacteria such as *Bacillus subtilis* [105], *S. pneumoniae* [105, 34] and *V. cholerae* [24, 130, 92], all of which are common sources of disease in humans, have demonstrated the ability to use competence.

Sporulation is another process that is controlled by a number of factors including quorum sensing. *B. subtilis* [105], [100] is an example of a type of bacteria that undergoes sporulation. When environmental conditions become adverse as a result of available resources being depleted and the LPD is high as measured by the bacterial quorum system, some *B. subtilis* cells will undergo a specific type of cellular division which produces a spore that is largely inactive but environmentally tolerant. This transformation allows the bacterial cell to wait until either the current environment changes and resources become available or the cell is transported into a new environment in which more resources are readily available.

Bacterial quorum sensing appears to be a relatively simple process but allows bacterial cells to survive and thrive in a multitude of environments. A key element of this process is not the location and distribution of individual neighbors, but rather that the cell is part of a community. In essence, the bacteria are able to abandon a level of determinism in their decision on how and when to behave socially without individually counting their adjacent neighbors. It is almost as if they are taking a non-deterministic approach to population counting and just relying on the observation that there are “enough” signs that a community is present; the bacterial cell should alter its strategy and behave as part of the community. Quorum sensing provides an estimate that
is “good enough” for this transition. The bacterial cells also have a strategy that ensures that the community transitions through the use of the autoinduction feedback loop where by all neighbors are also forced to into the HCD state. Determined adversaries who look to nature for sources of inspiration for C2 schemes could leverage this strategy and assemble targeted malware which tolerates non-determinism in its communication channel.

2.4 Determinism vs Non-Determinism

Asymmetry has been repeatedly discussed within the fields of information security and cybersecurity [89], [117]. Liang and Xiangsui have argued that exploiting the asymmetrical nature of cyberspace is essential in the future, especially when participation in symmetric conflicts will guarantee that an adversary cannot win. It has been said that this asymmetry exists in computer security, as the adversary only needs to find one weakness but the defender needs to protect all of its weaknesses. This is referred to as the Fortification Principle by Pavlovic [109].

Uncertainty affects actors and adversaries differently; their actions depend on their objectives and their tolerance towards uncertainty. Attackers by their nature are likely to be far more tolerant in making decisions which rely on non-determinism. Not concretely knowing a fact is less likely to alter an attackers strategy but this will depend on the type of attacker. Defenders are less likely to tolerate decisions which have an element of non-determinism. The defender is likely concerned with the decisions in which making a wrong decision could severely impact the operational status of a mission.

Consider if an adversary knows that a high-value target is in one of three buildings; they have a few different options. They can destroy one building or they can destroy
all of the buildings. The course of action that the adversary selects will depend on the ramifications of destroying the wrong building. If adversary is intent on minimizing the collateral damage (e.g., an adversary named Alice), they may opt not to destroy all three buildings. If the adversary does not care about the loss of the other buildings (e.g., an adversary named Eve), they can attack and destroy all three buildings. If the difference between the cost of destroying all three buildings is no different from destroying a single building, then Eve may opt to destroy all three buildings to ensure that the target is eliminated. In this sense, Eve has a high-tolerance to uncertainty; the cost is essentially the same but she only cares about the outcome.

Alice, the first adversary, will prefer concrete information; she is interested in knowing with a high degree of certainty that the high-value target is in one of the structures. Although Alice is an adversary, but she is an adversary who is concerned about minimizing collateral damage or just concerned about her positive public image, she may have a different tolerance towards uncertainty. In this scenario, the uncertainty about the exact location of the target would cause her to seek additional information to reduce the uncertainty before acting. She would desire to destroy the target and leave the other buildings intact and unharmed.

For Eve, the second adversary, the lack of information does not affect the strategy she employs when destroying the target. The constraints on Eve’s actions are different, she is not concerned about collateral damage nor is she concerned about maintaining a positive public image. For Eve, knowing that the high-value target is in one of three buildings is sufficient to have all three destroyed.

The attacks that an actor is willing to make are a result of the different information requirements of the actors; the defender needs determinism while the adversary can operate without it. For example, an adversary launching an attack on a remote host
may be comfortable with an exploit that only works one percent of the time or even with an attack that is successful only one in a million attempts. The adversary can either launch attack repeatedly until the host is successfully compromised or attempt to exploit a population of hosts which is large enough that at least one will be compromised. The network defender is unable to rely on brute-forcing the solution space as their mission relies on determinism to be successful. An enterprise would not be able to function if their software only worked one percent of the time. Even enterprises which guarantee 99.99% uptime still have difficulties when that 0.01% occurs.

Defensive strategies have attempted to exploit randomness to their advantage but this is a different method from exploiting non-determinism[53]. Modern Operating Systems make use of Address Space Layout Randomization (ASLR) in an effort to thwart exploits that rely on fixed addressing. When using ASLR, the network defender is not operationally impacted as they have complete information over the system; from their viewpoint they have complete knowledge of the systems memory layout. The defender (or more specifically the system the defender is using) has access to the locations of data structures and functions in memory. Conversely adversary initially has limited information about the addresses of interest. There are a couple of weaknesses to this defense; first if the adversary is able to force the system to leak information about memory layout they can modify their exploit accordingly. Secondly if the is using a source with low entropy when positioning code and data they attacker can attempt multiple attacks and guess the information that is required. Finally the attacker can largely ignore the addressing by spraying memory with NOPs and hoping a return will fall in the correct location to pass control to code that the adversary has injected. Ultimately this defense is deterministic in nature from the defenders perspective and may be non-deterministic from the adversary’s perspective. If the adversary is able to
gain information about the system, their perspective can become similar to that of the
defender and the defense can be defeated.

In these situations the network defender is limited in that they need knowledge
about the environment to successfully operate. The adversary is more risk tolerant and
can accept a non-deterministic approach, relying on situations in which their attacks
may work. The likelihood that their attacks are successful is high enough that they
can still accomplish their mission. If a determined adversary is seeking to leverage
different strategies which are non-deterministic in nature they may look to modify their
C2 channels such that they no longer rely on determinism. Recent research in covert
channels is identified a number of different network covert timing channel which are
resistant to identification by standard network tests such as similarity measurements or
entropy measures.

2.5 Covert Channels

The exchange and protection of information is an important aspect of information
security. It is often handled through the use of protocols which use cryptography such
as Transport Layer Security (TLS) [43] or Secure Shell (SSH) [143].

In situations where the user of an operating system requires more assurance than the system was designed with security in mind, the operator may seek proof that the system is secure. This evidence may be provided in the form of: artifacts from secure design reviews, results of protocol analysis, test results from security/penetration testing, or formal proofs that the system is secure may also be required. One of the first standards which required that operating systems be formally evaluated was the Trusted Computer System Evaluation Criteria (TCSEC) [106] also known as the “Orange” book. The
TCSEC included several different divisions each associated with a different level of protection that an evaluated product was demonstrated to provide. The B level is associated with Mandatory Protection in which the system is subject to Mandatory Access Controls (MAC) such as those which are typically described by the Bell-La Padula model [15].

In the Bell-La Padula model [15], [86], security is based on subject’s attempts to access objects. Subjects are processes and processes are performing actions on behalf of a user. Objects are items such as files and other resources which have a listed of attributes which assist in determining if access is allowed. All subjects and objects have permissions which are hierarchically related. An example of labels associated with MAC are labels such as RESTRICTED (R), SENSITIVE (S) and PUBLIC (P). Users have an associated clearance as in Alice has an R clearance which means that she can access R, S and P because her clearance level dominates the other levels in the hierarchy. Another system user may only have an S clearance so they can access resources that have S and P labels. In general, users in this model may access resources on the system in which their clearance dominates the security labels associated with a resource. The Bell-La Padula model [15], [86] formulates this in the form of two security properties; the Simple Security Property and the Star (star-symbol) Property. The simple security property is also referred to as “no read up” or in other words access is allowed as long as the subject is not accessing an object with at a higher security level. The star property is summarized as “no write down” or this property ensures that any information written by a subject cannot be written to an object existing at a lower security level.

For example, Alice who has a clearance of R can read files with R, S, or P security labels. Access is allowed through the Bell-La Padula model because she is not violating the simple security property. While Bob who has a clearance of S, can only read files
with S or P security labels. Attempting to read a file with an R label is not permitted since this would be a violation of the simple security policy. An interesting problem was noted by Lampson [86] with regards to violating the security policy of a system even though the subjects were able to behave in accordance with the security policy. Lampson [86] described and defined a Covert Channel as an information leakage on a system which occurs “through channels intended for other uses onto which the information is encoded.” A subject could pass information to a subject with a lower clearance through the use of a covert channel. As noted by Lampson, a subject could manipulate the attributes or the behavior of an object to convey information to a subject with a lower clearance level. As a result of this observation, operating systems functioning in high secure environments are required to be analyzed for the existence and bandwidth associated with a covert channel. Within the TCSEC, the B security level requires that the system be analyzed for both covert storage channels and covert timing channels. The B2: Structured Protection security level requires that a system includes an analysis of a covert storage channel’s ability to transmit information. While a system with the B3: Security Domain security level is required to be evaluated for its ability to mitigate and contain both covert storage channels and covert timing channels[106].

Covert channels attempt to secretly or discreetly relay information in a way that is only comprehensible by the sender and receiver. Understanding how information is relayed relies on information theoretic nature of communication [119]. Shannon and Weaver describe multiple phases which are necessary to exchange information (whether it be legitimate or covert) which are illustrated in Figure 2.6. The sender has information which they want to provide to the recipient. In order for this information to be conveyed the information must first be encoded into a message. In the next phase, the transmitter translates the message into a signal which is transmitted across
Figure 2.6: Shannon/Weaver Communication Channel
the communication channel. Shannon and Weaver include the possibility of a noise source which exists in the communication channel which modifies the signal as it passes through the channel. The signal that is received is a composition of the transmitting signal plus any modifications that the noise source makes in the channel. The receiver receives the signal and decodes the signal into a message. The message is decoded and then provided to the destination as information [119].

Similar to the traditional Shannon/Weaver channel, when a Covert message is included in the communication channel, addition information is encoded into the message that is transmitted. The overt message passes through a traditional encoder, but the result is again passed through the covert message encoder which embeds the covert message before transmission. The signal is decoded by the covert message decoder and then the message decoder resulting in the production of the covert information and the overt information.

Inclusion of a covert channel only slightly modifies the channel in the Shannon/Weaver model. Figure 2.7 illustrates the Shannon/Weaver channel which has been modified to include a covert channel. The overt sender has information which is sent to the overt recipient. The message encoder selects a message to be transmitted by the transmitter, but the covert message encoder intercepts the message and changes it before it is passed to the transmitter. The covert message encoder takes the overt message and selects another similar message which contains the same (or similar) information which also includes the covert information to be transmitted. The covert sender and the overt sender are likely two different individuals or processes which are attempting to send a message, but they can be the same. The transmitter sends a signal to the receiver. The same potential source of noise exists in the modified channel which can corrupt or change the signal. Instead of the receiver passing the message
Figure 2.7: Shannon/Weaver Communication Channel with the inclusion of a Covert Channel
to the message decoder, the covert message decoder first intercepts the message and
decodes the covert information and sends new message, possibly without the covert
information. The covert information can be acted upon by the recipient of the covert
message. The new message is passed to the overt message decoder which decodes the
overt information.

A covert storage channel is a class of covert channel. The TCSEC defines a covert
storage channel as “all vehicles that would allow the direct or indirect writing of a
storage location by one process and the direct or indirect reading of it by another.” [106].
To illustrate a covert storage channel, assume that a system has a lattice security model
in place based on Bell-La Padula and there are two subjects using the system; Alice and
Bob. Using a covert storage channel, Alice will be able to communicate information
labeled with R even though this is a violation of the security policy to Bob. In order
to leak this information to Bob, she can create a file on the system with an R security
label. If Bob attempts to access the file, the simple security property will prevent him
from accessing it. If Alice removes the file from the system, when Bob now attempts
to access the file it will be permitted (as it does not exist so he can create it). If Alice
and Bob agree on a protocol to exchange information Alice can leak information to Bob.
Bob repeatedly tests for access to the file. If the file exists and cannot be accessed then
it is interpreted as a ‘1’. If the file does not exist then it is interpreted as a ‘0’. Bob can
collect the bits and reassemble the information from the binary steam.

Another type of covert channel is the covert timing channel which embeds
information to be leaked as a time series of events. The TCSEC defined a covert timing
channel as “all vehicles that would allow one process to signal information to another
process by modulating its own use of system resources in such a way that the change
in response time observed by the second process would provide information” [106].
For example in a covert timing channel, a packet is periodically transmitted across a network. If a subject such as Alice is able to control when these packets are transmitted then Alice can pass information to Bob by modifying the delay between packets being transmitted. Alice and Bob can devise another protocol in which if the delay between two packets is 1 second it is interpreted as a ‘1’ and if the delay between two packets is 2 seconds it is interpreted as a ‘0’. This system would allow Alice and Bob to exchange information as a binary stream in violation of the security policy. Although this process is an example of a covert timing channel, technically it is classified as a network covert timing channel information is being encoded into the packet transmission times as packets are being transmitted across a network. Similarly covert storage channels have networking versions in which the contents of a packets header are modified; these are referred to as network covert storage channels.

Simmons’ provided the essential model that is used to describe the situation in which a covert channel is used [124]. In this model, there are two prisoners who have been captured (e.g., Alice and Bob). They are interested in abandoning this concrete-bound lifestyle so when they are approached by the warden, Wendy, who is willing to allow them to communicate, they gladly accept. There is restriction on what they are allowed to say; everything needs to be appear to be completely open to Wendy. In other words, the prisoners cannot use encrypted communications or pass messages that are seemingly incomprehensible to the warden.

The prisoners need a communication channel in which they can pass information. The warden can possibly interfere with the information being communicated by behaving in one of three different ways [36], [146]. The warden could decide not to modify the message, acting as a passive warden, and observe and monitor all of the messages are they are being passed. Alternatively the warden can take a more active
role (i.e., the Active Warden) and subtly alter the messages as they are being passed. Finally the warden can become outright malicious in their behavior and insert, remove, or corrupt messages as they are passed. The communication channel that is being used ideally would allow Alice and Bob to prevent the passive warden from eavesdropping on the conversation and be resistant to both the active and malicious wardens.

Since Lampson’s original description of a covert channel, a number of new and novel techniques relying on covert channels have been devised and countermeasures against some of these techniques have been proposed in the literature. Beyond these simple convert timing channels and covert storage channels: Hybrid, Counting [66] and Statistical [102] channels have been proposed and analyzed. Hybrid covert channels are covert channels which combine the ideas of covert storage and covert timing channels to convey information[29]. Gray III, described counting covert channels are a district class of covert channel in which the number of occurrences of an event are used to relay a message to the receiver [66].

In addition to describing counting covert channels, Gray III proposed two different countermeasures which could be used to prevent an adversary from exploiting counting covert channels[66]. One countermeasure was to utilize fuzzy time, which interferes with the local reference time source that the sender is relying on so the time based events are no longer guaranteed to occur as the sender expects. Another mode is probabilistic partitioning, in which one of several different classes of controllers are applied to the signaling traffic and each controller modifies the traffic in a different way. With this method, the sender cannot rely on the predictable nature of the channel.

Moskowitz and Kang introduced the idea of a statistical channel of which is a subset of the covert timing channel class of covert channels. This channel was discovered when a network defense called “the Pump” was being developed in order to securely move
data between sensitive environments[102]. The pump uses a buffer to hold information before transmitting it from HIGH to LOW. The issue was also determined to be present in the pump when a network version of the application was developed [82].

Network covert storage channels have been widely proposed and analyzed for various IP based protocols, including; TCP/IP [7, 63, 134], ICMP [37, 126] and HTTP [13, 26]. Li et al. has proposed and analyzed network covert storage channels at the MAC layer of the network [88]. A number of more recent studies have focused on network covert timing channels. Cabuk et al. proposed and analyzed on/off timing channels, which are another class of network timing covert channels [29]. The on/off timing channel is essentially a one-way covert communication channel with limited two-way communication allowed for acknowledgement of data received. The covert channel is based on encoding data into frames and transmitting information during a pre-determined interval. If during this interval data is received, then it is interpreted as a ‘1’ while if no data is received then it is interpreted as a ‘0’. The covert channel includes synchronization and parity bits to ensure that the integrity of the information is preserved. Cabuk et al. noted during their study that as the interval is decreased (e.g., the intervals are made smaller), the channel becomes increasingly susceptible to being overwhelmed by noise on the network [29].

Research on the on/off covert channels yielded two different proposed methods for detecting and defeating that class of covert channel. The first methods developed by Berk et al. rely upon Information Theoretic tests to measure the information capacity of the covert channel to detect its presence. Another method relies on statistical characteristics of the covert channel for detection [21]. Later Gianvecchio and Wang proposed a more general approach to detecting network covert timing channels which relied upon measuring the entropy associated with a channel [61]. While Cabuk’s
method was acceptable, Gianccecchio and Wang’s method provided a more general approach that is not focused on detecting a specific network covert timing channel.

Yao et al. proposed separating on/off timing channels into two distinct categories; deterministic on/off timing channels and non-deterministic on/off timing channels. Yao et al. performed an analysis which concluded that although deterministic on/off timing channels are more often studied and analyzed; non-deterministic on/off timing channels possibly pose a more significant threat to maintaining the security policy of a network [142]. Deterministic on/off timing channels are deterministic in that there is no ambiguity in the information that this received by the receiver. To illustrate the difference, if the packets traversing encounter delays on the network, but this noise is insufficient to cause any ambiguity in the arrival of the packets for the receiver, then the channel is classified as a deterministic on/off timing channel. The receiver is able to determine where in the sequence the arriving frame belongs based on its time of arrival. If the network delays on the network are sufficient that the network noise has introduced the possibility that packets arriving at the host are out of order, the receiver is now listening to a non-deterministic on/off timing channel. The packet could be correctly sequenced or it could be out of sequence and belong to an adjacent frame. The ambiguity in the arrival times is a source of non-determinism which will result in the receiver’s inability to deterministically figure out where in the sequence the packet belongs. The ambiguity will result in decoding errors, unless the covert channel is designed to tolerate non-deterministic signals.

Sellke et al. introduced another type of covert timing channel which relied on the timing between packets to convey a message [118]. This process is different from other covert timing channels as instead of relying upon the presence or absence of traffic to denote the individual bits of a byte stream, the inter-packet times contain the encoded
message. The proposed low bandwidth channel is seeks to evade detection by minimally affecting the timing which causes the covert channel to become “computationally indistinguishable from normal network traffic” [118]. The traffic was indistinguishable from normal traffic as the signal was modeling on an independently and identically distributed Pareto distribution. By modeling the packet distribution on a Pareto distribution, the traffic appears to be normal traffic as far as a network defender is concerned. Another advantage of this covert timing channel is that time synchronization between the sender and receiver is not necessary.

In addition to performing an entropy-based analysis to detect covert timing channels, Gianvecchio also introduced a model-based covert timing channel [62]. This model-based covert timing channel relies on the statistical properties of the local network traffic to mask its presence. In addition it provides a generalized framework for designing and building covert timing channels. Liu later extended the model to include distribution matching to minimize the differences between legitimate traffic and encoded traffic. These differences can allow the covert timing channel to remain undetected [91].

Covert channels provide a family of techniques which can been used to model how adversaries may leak information from sensitive system to unauthorized external entities. Leaking information is not the only use case for covert channels. Covert channels can be used as a C2 channel for targeted malware. By moving away from deterministic communication and relying on the noise found within network traffic covert channels can enhance the C2 channel’s resistance to detection. C2 channels for targeted malware could also follow the trend and become non-deterministic in nature to make the C2 traffic more resistant to detection.
2.6 Digital Quorum Sensing

Bacterial Quorum Sensing has been proposed as a model for malicious mobile code to determine when to stop replicating. There are a few papers in biologically inspired computing and this research extends these works in a number of different ways. The most relevant work is Vogt et al., in which the authors proposed Quorum Sensing as a C2 system which would allow self-replicating worms to stop replicating when the network was sufficiently saturated with malware [137]. The research that is part of this proposal is different from the work proposed by Vogt et al. in a number of ways. The critical difference is that the C2 channel being used by the malware proposed in this research is not generating custom packets which relay population information between infected nodes, while the malware being discussed by Vogt et al. is using the state transition as a stopping mechanism. The C2 channel in this dissertation is a covert timing channel that will allow the malware to non-deterministically relay information to other infected nodes on the network without exposing the individual nodes to remediation. Each infected node will probabilistically modify the network traffic that is being emitted. As a result, the packet distribution of the entire network is altered such that the malware is estimating the likelihood that the network is sufficiently infected. Vogt et al.’s work utilized a custom protocol to exchange population information. The protocol used to exchange information like most other C2 channels is inherently deterministic in nature. Reverse engineering Vogt et al.’s protocol would allow network defenders to deterministically identify individually infected nodes and the total malware population within a network. Vogt et al.’s work also mentions potential applications and communication profiles, but bacteria use quorum sensing for a variety of different purposes. This research

53
highlights a number of additional possible paradigms that targeted malware can leverage in subsequent chapters.

Self-stopping worms (also known as Camouflaging or C-worms [145] [144]) use Digital Signal Processing techniques to analyze the network traffic as the C-worm proliferates through a network. Use of digital signal processing techniques for modeling network traffic was proposed as a possible method that malware could use to estimate the local population density during the proof of concept study. The communication channels for the malware in this research are different in that it uses non-deterministic techniques to transmit the infected population density via existing network traffic. In order to uncover the presence of the proposed targeted malware, a network defender could attempt to utilize digital signal processing techniques to monitor the network traffic; nevertheless, the network defender can only estimate the probability that the network is at a specific infection population density.

Hamar and Dove have discussed Quorum Sensing as a potentially useful security pattern [70]. Hamar and Dove’s discussion is based on bacterial quorum sensing and quorum sensing in honeybees. The applications to security discussed by Hamar and Dove are related to data mining of social networks for suspicious indicators that could lead to an attack [70]. The proposed communication scheme used within this research draws extensively from the model utilized from bacterial quorum sensing. The application of digital quorum sensing in this research is specific to autonomous signaling of malware and the use of signals that do not rely on generating novel network traffic (existing traffic would be sufficient to allow signals to be exchanged) in a targeted attack from a determined adversary. The C2 channel in this research would not result in information being exposed on social networks because the networks targeted by
determined adversaries usually do not inform the social media that they have been compromised.

In some sense, this problem is similar to those being solved by distributed algorithms. This work is different in that the targeted malware proposed within this work is self-organizing. The hypothetical malware is not attempting to determine which version of a file to commit and there is no single general who is attempting to determine when to attack. The network collectively determines when to transition to the high cell density strategies.

2.7 Ethics of Security Vulnerability Research

As part a result of this research, a novel C2 channel has been characterized that could allow the authors of targeted malware to coordinate their actions in a way that is less likely to provide actionable intelligence to network defenders. The ethical considerations of doing security vulnerability research, specifically with regards to these experiments is discussed. The research does not make use of strategies that hacktivists, cybercriminals, or determined adversaries are using when attacking a network.

Before a full discussion on the ethics of vulnerability research is provided, the author would like to highlight the following points:

- This research is focused on characterizing the potential C2 channel and identifying possible defenses, not with providing malware authors with a novel communication channel.
• No malware was developed during the course of this research. The emitters and receptors are not implemented in a way that would allow a malware author to readily reuse the software.

There have been several different methods suggested for determining if a work of security vulnerability research is ethical. Carle compared the ethics of several different attacks against different ethical guidelines [30] and made a determination about the nature of the situation based on utilitarian ethics, a right approach, and a common good system. The author did not propose any general ethical standards by which a work could be judged but instead analyzed a few situations and detailed proposed responses measured against three different ethical systems. The circumstances posited in each situation were constructed such that an entity was under attack with their systems being compromised. In all of the situations the conclusion was that it was not ethical to “hack back” or releasing a computer worm. These conclusions were drawn from a few limited cases and the approach was not broaden to address the situation in a more general sense.

Recently, the Department of Homeland Security (DHS) has released the Menlo Report [11], which attempts to formalize a discussion about the ethics of information technology research in general. The Menlo report, which is based upon the earlier Belmont report [55], offers four principles which should be adhered to when performing information technology research [11], [12];

• **Respect for Persons:** It recommends that research be performed with informed consent. It also highlights that although the research is may be performed in an isolated environment, others may be impacted with the release or publication of the work.
• **Beneficence:** This principle is commonly referred to as the “Do not harm” principle in which research should not cause any harm to any test subjects. This is also interpreted as maximizing the benefit and minimizing the harm as a result of any research. Furthermore the Menlo report recommends that the researchers continue and systemically assess both the risk of benefit and harm [12].

• **Justice:** Another principle that is part of this ethical standard and attempts to ensure that all subjects of the research are treated fairly and equally.

• **Respect for Law and Public Interest:** A fourth principle was included in this guidance which differs from the Belmont report [55]. This principle ensures that the research is conducted legally using research that are open and transparent.

When comparing these principles to the research in this research, this work adheres to the advocated principles. Respect for persons is maintained as this work does not use any human subjects as part of the research. At first glance it could be argued that this work only benefits the attacker as it gives the adversary an advantage in the new form of a new tool in which they can maintain persistence on compromised networks. This research is provides and opportunity for network defenders to hear about this potential C2 channel before they observe it on their networks. This advanced discussion will allow the defender to prepare additional defenses above and beyond what is discussed later in this work.

The next principle enumerated by the Menlo report is Beneficence. Again, it could be argued that this research will help the determined adversary, but it must be considered that there are external nation states which qualify as a determined adversary. This work is being released publicly through respected security-related conferences and journals such as the International Conference on Malicious and Unwanted Software (MALCON).
which is sponsored by the Institute of Electrical and Electronics Engineers (IEEE) which the initial findings from a proof of concept were published and discussed [54]. By releasing the work in venues such as MALCON, it is hoped that the security community can begin to collectively work to find countermeasures and defenses against DQS and other similar schemes. This is an effort to maximize the benefits by removing the advantage the attacker would have employed by using an unknown and novel C2 channel in the wild.

The third principle listed in the Menlo report is Justice. This work adheres to the Justice principle in that there are no sponsors of this work who are seeking to employ this C2 channel against unsuspecting networks. This work is not being developed under the guise that it will be directly provided to an adversary would could leverage the techniques. Human subjects are participating in this research. The work is being released in venues who are attempting to advance security in a positive way such as MALCON and the New Security Paradigms Workshop (NSPW).

Finally there is the principle of Respect for Law and Public Interest. This research is not designing and implementing malware, it is only studying a communication channel which hypothetical malware could attempt to deploy. No malicious payloads or any forms of self-replicating malware are being produced. Conducting this research is in the public interest, for the reason that determined adversaries have large resources available to develop and deploy novel attack strategies. Denying the use of an attack strategy before it can be used is not something that traditionally occurs in security research. Normally the attacker has the advantage and the first time a defender encounters a new attack is when it is used against their network or they have heard of it being used against another network.
Schrittwieser et al. proposed another set of principles which can be used to ensure that security research does not become unethical [116]. The principles that they use are different enough that they should also be discussed in conjunction with the research included within this proposal. The four fundamental principles are: do not harm humans actively, do not watch bad things happen, do not perform illegal activities to harm illegal activities, and do not conduct undercover research [116]. How this research falls in line with these principles is covered within the previous discussion of the Menlo report’s principles.

2.7.1 Effects of Choosing to Disclose a Vulnerability

As recommended by the Menlo report, a detailed assessment of the ethics of information security research needs to be performed during the research process. An important aspect of the ethics involved in security research is if attack research should be performed and how those vulnerabilities should be performed. As previously stated, there are two parties which will be affected by this research; mainly the adversary and the network defender beyond that of the researcher. The effects on the adversary and defender will be discussed in this section.

There are two different cases to consider with regards to the security research; should the research be performed or should it not be performed. The first case to be looked at is the case in which the research is performed and it affects both the attacker and the network defender differently. The important distinction for the attacker and defender is if the attacker is already aware or making use of the vulnerability included in the research. The following pages will discuss the different situations resulting from the
research and disclosure of a potential security vulnerability and how the attacker and
defender are affected.

**Case 1a: Research is performed; adversary is aware**

In the first case, the security research is performed but concurrently and separately. The
adversary is aware of the vulnerability and may be actively exploiting the vulnerability.
Essentially the security researcher and the adversary independently identified the
security vulnerability. The adversary is interested in using the security vulnerability and
the longer the vulnerability remains undisclosed the longer the adversary will be able
to use it to inflict harm. The network defender is likely unaware of this vulnerability as
such they are not looking for it, and will likely not be able to detect the attack, they will
only detect the misbehavior of their systems.

If the security researcher does not disclose the vulnerability, only the adversary
benefits. The network defender cannot effectively defend against something they are
not aware of. The only defense the defender can rely on is the existing security
infrastructure. Conversely, if the security researcher discloses the vulnerability, the
adversary loses the ability to continue to stealthily exploit computer systems. The
network defender may not have the advantage initially, but now that they are aware of
the vulnerabilities existence they can develop mitigation strategies and countermeasures.
In the long term strategies might be developed which cause the attacker to abandon the
use of the vulnerability; from a cost perspective the only benefit the vulnerability has is
the use of the vulnerability between the time of the adversary’s first use and the time of
the researcher’s disclosure. The adversary may no longer be able to gain a continued
benefit from the vulnerability after the vulnerability has been disclosed to the general
community, provided that a patch is made available and widely deployed.

60
Case 1b: Research is performed; adversary is unaware

In the next case, the security researcher performs vulnerability research and discloses the presence of the vulnerability to the community; both the adversary and the network defender are previously unaware of the vulnerability. The adversary may look at the vulnerability and determine that the vulnerability is interesting and may attempt to use it, but ultimately the vulnerability is less attractive to the adversary. The entire community is aware of the vulnerability. There may be an advantage for the adversary; both the network defender and the adversary are aware of the vulnerability.

The adversary and network defender may be starting from the point with regards to the disclosure of the vulnerability. The adversary is not ready to exploit the vulnerability but the network defender is not ready to defend against the vulnerability. There is a race to between the adversary and the defender to determine who can either develop the exploit or the countermeasure. The window for exploitation is has been reduced. The adversary no longer gains the benefit of using a previously undisclosed vulnerability. The adversary may decide to use the vulnerability, but as the window for exploitation has been reduced, so have the potential gains. The attacker may decide that the potential benefits for developing the exploit are not sufficient to justify the cost of development.

Case 2a: Research is not performed; adversary is aware

Another case to consider is if the research is not performed and the adversary is already aware of the vulnerability. The vulnerability may have been discovered by the attackers research team but no external entities are researching the potential vulnerability or possible mitigation strategies. The adversary has the advantage in this situation. They can use the exploit and the network defender is effectively helpless against the
vulnerability. The network defender does not know what to look for other than relying on their host monitoring systems to determine that a system has been compromised.

In this situation, the adversary is motivated to develop and use the exploit for the vulnerability. The window for exploitation is large and depending on how and where the exploit is used, the window may not be closed for the foreseeable future. The only thing the adversary is concerned about is the vulnerability being detected after it has been used. If the network defender detects the malware or determines that a system has been compromised as a result of its misbehavior, only then can the network defender begin to understand the vulnerability. The network defender must reverse engineer the vulnerability disclosing it to the community and only then can effective mitigation strategies or countermeasures be developed.

**Case 2b: Research is performed; adversary is unaware**

Another case to consider is the research is performed and the adversary is unaware of the vulnerability. Effectively only the researcher is aware of the vulnerability and no one else has information about the vulnerability. The attacker is unaware of the vulnerability, so they cannot use it to compromise computer systems. The network defender is unaware so they are not looking for the signs that their systems have been exploited by the vulnerability.

This scenario is an unstable state. Adversaries are interested in discovering and exploiting previously undisclosed vulnerabilities. It may be only a matter of time before the adversary independently discovers this vulnerability. If the security research does not disclose the vulnerability, then as previously discussed, the adversary is the one who has the advantage. Another thing to consider is that attackers may be looking to leverage cutting-edge scientific research, as such they may attempt to exploit the institutions and
systems used by researchers to gain insight into the vulnerabilities and defenses that are being researched. If the security researcher has an idea for a vulnerability, and stored this information on a compromised system or communicated this idea to a peer, the adversary may be aware of the potential use for the vulnerability.

2.7.2 Process of Vulnerability Disclosure

Another key consideration of the ethics of information security research is how the vulnerability is disclosed to the community. Ethics should be important for the information security practitioner, as such two certification entities: Certified Information Systems Security Professional (CISSP) by International Information Systems Security Certification Consortium (ISC)² and the Global Information Assurance Certification (GIAC) formed by the SANS Institute maintain a code of ethics. Both organizations and certification entities require that the practitioner work to protect and work for the benefit and welfare of society. With these considerations in mind, ethical considerations are important when security research is performed, the vulnerability is disclosed in a way the benefits society.

The disclosure of a vulnerability is a controversial subject as multiple parties can be affected. There are a few different methods of disclosing a vulnerability; non-disclosure, full disclosure and responsible (or coordinated) disclosure. Each of these disclosure methods affects three different parties differently; the security researcher, the specific vendor affected by the vulnerability, and the general community of those relying upon the vendors products.

Non-disclosure is when research is performed and no one is notified of the discovered vulnerability; essentially the researcher discovered a vulnerability and tells
no one. This situation does not benefit society or the community as a whole. There may be situations in which a critical vulnerability is discovered and the researcher is concerned about the implications of disclosing the vulnerability. A key concern here is that the attacker may have independent knowledge of a vulnerability and may actively be exploiting it. As the researcher has informed no one, the community cannot act to determine if the vulnerability is being exploited and develop possible mitigation strategies or countermeasures. Ultimately, anyone relying upon an affected technology is vulnerable to exploitation whether the vulnerability is disclosed or not.

The other extreme is full disclosure; when the security researcher discloses all or nearly all of the details of the vulnerability to the general community. If the vulnerability affects a specific vendor, this situation generally benefits the attacker. The vendor is in the situation in which the vulnerability must be verified, a countermeasure (likely in the form of a patch) needs to be developed, and then released to the community. The community must accept and apply the patch and determine if the countermeasure fixes the vulnerability without negatively affecting their systems. While the community is responding to these actions, the attacker can review the exploit or even separately develop an exploit for a vulnerability and use it against systems which have no defenses.

A middle ground between the two extremes is responsible or coordinated disclosure. Organizations such as the United States Computer Emergency Response Team (CERT) and Microsoft have responsible disclosure processes while Google maintains the coordinated vulnerability disclosure process. These guidance and procedures for helping to ensure that vulnerabilities are disclosed in a way that benefits everyone while minimizing the benefits to the attacker. In this process the security research discloses the vulnerability to the vendor affected, and then coordinates a release schedule in which the vulnerability is patched so both the vulnerability announcement and patch released
to the community simultaneously. This process ideally gives the vendor time to validate the vulnerability, implement a patch and release the patch when the vulnerability is announced. When the vulnerability is announced, those affected by the vulnerability have a patch or mitigation strategy available which can be implemented. The network defenders are not left defenseless by the vulnerability announcement.

This research in this work is an idea, not a specific vulnerability or features being used insecurely and being exploited by the C2 channel proposed in this work. Microsoft Windows is not more affected by this vulnerability than a GNU/Linux system with Canonical’s Ubuntu. As such patch for a vulnerabilities cannot be developed with the vendors for release. In lieu of this, the research will be published publicly to the security community through journals and conferences which are focused on enhancing security as opposed to promoting insecurity. The first portion of this work, the proof of concept was accepted and presented at the MALCON2014 conference which is sponsored by the IEEE and Microsoft. Another paper was accepted to the 2016 New Security Paradigms Workshop (NSPW).
Chapter 3

Architecture

Digital quorum sensing is proposed as a possible method that targeted malware authors could utilize in order to allow the malware to coordinate activities in a way that provides network defenders little actionable intelligence if the malware’s C2 channel is discovered. Allowing bacterial quorum sensing to serve as a source of inspiration for DQS, a system could be constructed which makes use of an emitter, a receptor, and an autoinducer.

C2 channels are a weak point for malware. The use of traditional communication channels allow network defenders to identify endpoints on the network which are infected. The reliance on deterministic communication does not mean that it is a trivial task to allow the endpoints to be identified, but rather it means that as each message contains the source address of the sender and the destination address of the receiver. The end points in the communication can be identified and enumerated. If the malware C2 scheme can be reverse engineered and the network traffic is inspected, then it is possible with enough time and resources a network defender could construct a list of nodes infected on their network.
As with the case of bacterial quorum sensing, the autoinducer does not indicate the location of individual bacterial cells but rather it is used to estimate the local population density. With this in mind, a digital quorum sensing system would require the use of a communication channel that does not indicate the precise addresses of the communicating nodes but simply that the nodes exist. A resource available to all nodes on the network is the traffic that is being transmitted or rather the packet distribution as a function of time. Different covert timing channels have been designed and demonstrated to modify the timing of packets on the network to transmit information. These channels have the potential to be abused and violate the local security policy. This research demonstrates that a determined adversary could use the global traffic network distribution to communicate information to infected nodes.

3.1 Cocktail Party Example

As an illustration of the proposed communication scheme, consider the following thought experiment. Alice is interested in having a cocktail party. Unfortunately Eve is aware that there is a party occurring but she is has not been invited. Eve has some individuals who are sympathetic to her plight, and they are willing to help make mischief at the party. These individuals are willing to help, but they are only willing to help if there are enough sympathizers to have a good chance of disrupting the party.

Eve must find a way to coordinate their activities without exposing individual sympathizers. Furthermore Eve wants to ensure that her sympathizers only attempt to disrupt the party if there are sufficient numbers of sympathizers. If there are only a small number of sympathizers who are able to attend the party, then Eve decides that it is better not to act.
Alice intentionally did not invite Eve and is aware that Eve may attempt to disrupt the party. In order to ensure that the party proceeds as normal, Alice hires Walter to provide security services for the party. Eve suspects that Walter has been hired and because of this Eve has determined that secret hand-shakes are insufficient to coordinate the disruption. Secret handshakes could be used, as each sympathetic partygoer could count the number of times they use a secret handshake and make determination if there is a sufficient population of sympathizers. With the installation of a few cameras and support personnel, Walter can passively monitor all of the guests at the party. Walter is naturally suspicious, and if he notices that a secret handshake is being used he can play back the party recordings and identify people with potentially nefarious intent. These people can be removed from the party with a high degree of confidence that they are all part of the same network.

Since the use of a secret handshake is a liability for coordinating the disruption of the party, Eve must devise a new coordination scheme. Eve is interested in ensuring that the individual sympathizers are able to communicate and act, and she has something of an epiphany. The sympathizers only need to share that there are enough sympathizers ready, each individual is not really concerned with tracking the status of each individual, only the total. So Eve decides to leverage a common resource available to all members of the party; the background noise. It is reasonable to expect that individuals and small groups of people would converse during the course of a party. As these small groups of conversing people do not form the population of the party, there are normally a large number of conversations carrying on at the same time. These conversations when observed at the overall level generate what amounts to a consistent amount of background noise.
Eve realizes that she has a channel which she can use to convey a global message. The scheme she devises is that at the beginning of each minute, a sympathizer will simply pause a moment before speaking. If there are only a small number of sympathizers at the party, the background noise generated by a conversation will suffer a small dip in volume. If there are a significant number of sympathetic individuals, then the dip in volume will be more pronounced.

Walter is monitoring the party, he may or may not be monitoring the background noise in the party. At the point in the party where the dip in volume becomes pronounced, Walter may notice the dip when all of the sympathizers notice the dip in volume. At this point, if the sympathizers act, Walter will be overwhelmed. If Walter recognizes the change in the background noise of the party before the sympathizers act, then Walter can act beforehand.

Walter must first determine the source of the periodic reduction in background noise. He realizes that the noise is dependent on all of the individual conversations occurring at the party. He must now attempt to determine the motives of all of the individuals participating in each conversation. Walter has several different hypotheses that he must choose from; 1) reduction in volume was naturally occurring as a result of the flow of conversation, or 2) the reduction in background noise was maliciously inserted as a form of communication.

It can become a guessing game for Walter when he starts monitoring individual conversations. He has to attempt to determine which person conversing is causing the pause and if that pause is something that signals something potentially malicious. To further illustrate the difficulty; Walter will be asking himself questions similar to the following; “Why did that person take a breath?”, “Why did that person stop talking when he reached for that glass?”, etc. And because Walter must ask these types of
questions about each person at the party, it is easy to see how Walter can potentially get overwhelmed when attempting to answer all of these questions. Sympathizer is not concerned why each pause occurred but rather that the pauses are occurring.

To make matters worse, Eve may determine that she is willing to tolerate a level of uncertainty and accept that there may be a missed opportunity for disrupting the party or a premature attempt to disrupt the party. Having each sympathizer at the party always pause will eventually make it obvious that the individual is not communicating at the specific interval. Eve decides to modify the communication such that each sympathetic individual does not always pause at the beginning of the minute when conversing. When making this determination of whether or not to pause, the scheme could be as simple as selecting a random number between 1 and 10. If the resulting number is 1, momentarily pause your speech. Using this updated scheme it would be reasonable to expect that about 10% of the sympathetic hosts are pausing their conversations during any given minute.

If Walter is monitoring the conversations he now has a new set of hypotheses when observing conversations between individuals. Walter must attempt to make the following determinations when following conversations of individuals who did not pause during the minute; 1) they are a normal non-sympathetic guest, or 2) they are a sympathetic guest who opted not to participate during this window. For the guests who did participate during the periodic pause; 1) they are normal non-sympathetic guests whose conversation naturally paused or 2) they are sympathetic guests who are actively communicating during this window. There is no longer a simple distinction between non-sympathetic guests and sympathetic guests. Even if Walter is absolutely certain that the dip in background noise is not normal or even be aware of the protocol before
the party, he cannot start evicting partygoers without significant chance of angering innocent partygoers.

This cocktail party example details some of the difficulties that the defender must counter if he is attempting to determine if individuals are participating in a C2 channel that is similar to quorum sensing. This thought experiment is useful, but it lacks many details that describe how a system like this could be utilized in real world situations.

In a real-world scenario, the party would be a network that is conducting sensitive or mission critical activities. Eve is a determined adversary that is deploying targeted malware to attempt to disrupt a mission. The potentially sympathetic individuals would be the workstations which are being targeted by the determined adversary; the workstations infected with the targeted malware would be considered to be sympathetic to the determined adversary. Walter is a network defender or otherwise defensive monitoring capabilities that are attempting to protect the network. The following section describes a potential architecture which a determined adversary could deploy in an attempt to disrupt a mission.

### 3.2 Proposed Digital Quorum Sensing Architecture

Determined adversaries are motivated to find and use new attacks to accomplish their objectives. As determined adversaries are often resourced by nation-states, they have the ability to bring to bear significant resources to tackle difficult issues. More recent malware developed by determined adversaries and even cyber criminals has become modular allowing payloads and C2 channels to be customized or replaced as the situation requires. The following architecture is an example of how targeted malware could implement DQS as a C2 scheme. It is assumed that this C2 channel will be
only used in situations where mission disruption is the primary objective and the target is a high-value network subject to intensive host and network monitoring. It is not unreasonable to assume that a determined adversary funded by a nation-state has the tools and techniques to bypass and evade detection at the host-level. As such, the two main methods for detection are the observable artifacts as a result of malicious activities and the detection of C2 channels on the network for coordinating the targeted malware.

It is quite possible that a determined adversary could leverage strategies employed by biological system and if so, then the use of a digital quorum system is an option. If bacterial quorum sensing is used as a source of inspiration, three components will be required;

- **Emitter**: the digital analogy of the autoinducer synthase which manufactures and releases autoinducers into the environment. The emitter will insert the quorum sensing signal into the network traffic being released from the compromised host.

- **Autoinducer**: the actual signal produced and released into the environment. In the case of malware this will not be a chemical but will be a signal embedded into existing network communication.

- **Receptor**: the component in bacteria and the targeted malware which detects and responds to the presence of the signal. The receptor only reacts to the presence of the signal, it does not determine the result as the state transition.

Before the emitter and receptor are discussed, the autoinducer must be determined as it will define the implementation of the emitter and receptor. Relying on the previous illustration for the cocktail party, the network traffic could be modified to contain the quorum signal. A simple illustration of this signal would be that an interval of 1 second
is selected, during the lower half of the second \( t = [0, 0.5) \), packets are probabilistically delayed and prevented from being emitted on the host. The likelihood of the packet not being emitted is configurable depending on the strength of the signal the targeted malware is expecting for a signal. During the upper half of the second \( t = [0.5, 1.0) \), all delayed packets are emitted onto the network and all packets scheduled to be released during this period are normally released. The signal is measurable in the overall distribution of the packets being received by the hosts network interface. The signal must be repeatedly measured to determine the likelihood that the signal is present.

This example serves as a basic illustration of the communication channel. If a signal like this was used it could be easily detected and noticed be significant transmission delays on the network. A more realistic illustration would divide a single second into a hundred intervals and a number of intervals would have emission probabilities closer to 1.0 instead of 0.0. Packet delays of 10 ms are less likely to be noticed and can be explained away through one of several possible hypotheses: 1) the host was busy (if in the middle of a transmission), 2) the application transmitting data was busy (so there was nothing to transmit), or 3) there was a delay resulting from the network infrastructure.

This example illustrates several requirements which the emitter and receptor must implement. The targeted malware could require kernel level access to control the emission and monitoring the reception of the packets (See Figure 3.1). This access would likely be handled via DLL injection into the core networking libraries or loading of a custom driver which would manipulate the networking stack. The emitter would also require a source of noise and rely on a pseudo-random number generator (PRNG) or make use of the operating systems secure PRNG if present, in order to determine if a packet will be transmitted or delayed.
The targeted malware would rely on the emitter to modify the distribution of packets being released on the network. Depending on the nature of the quorum sensing system being used as a source of inspiration, either a single or multiple signals would be embedded in the packet distribution. Quorum sensing systems do not rely on a large number of signals. The information being transmitted is minimal; it is likely that a few signals would be required and the targeted malware could even operate with only a single signal. The probability that a packet will be emitted during a specific internal and affect the strength of the signal. Relying on probabilities that are more subtle than coin tosses, the effects of 10% deviation is less significant than a 50% deviation. This could allow the signal to remain undetected longer but increase the difficulty of correctly identifying the signal. The determined adversary must make a decision on the nature and strength of the signal to be embedded in the network traffic.
The targeted malware would also include a module which acts as the receptor. The receptor monitors all network traffic that the host's network interface card (NIC) receives. This level of access means that either the receptor is operating at the kernel level so all traffic can be inspected or it is relying on a network driver which is monitoring and capturing the network traffic and returning packet distribution information (See Figure 3.2). The interaction and information required by the quorum signal receptor is minimal, only the time of arrivals are recorded for each packet. The receptor is not concerned with any other information in the packet other than the time at which it is seen by the NIC.

The quorum signal receptor tallies the packets as they are observed by the NIC. The receptor is configured to monitor for one of several different signals. The simplest case is to monitor for a single signal but like the emitter, bacteria rely on a few different autoinducers so the number of signals to be monitored for should be minimal. The emitter is dependent on the presence of noise within the network traffic to hide its signal.
While the emitter is counting on hiding within the network noise, the receptor must contend with the noise to tease out the signal from the network traffic. As a result the receptor may leverage techniques from Digital Signal Processing (DSP) to detect the presence of a signal.

The quorum sensing receptor process is illustrated by the block diagram in Figure 3.3. When the receptor initializes, it creates a set of bins, each of which corresponds to a different window. The number of bins created is configured by the determined adversary while the time interval that corresponds to the big is dictated by the interval by which the emitter is partitioning the probabilities of emission. For example, the simple emitter divided a second into two separate intervals one for the lower half of the second and another for the upper half of the second; each bin on the receptor would correspond to half a second. The receptor could create two bins and just populate those bins. Unfortunately for the adversary, relying only on two bins would result in an obvious signal. For detection of more complex signaling patterns, multiple sets of the entire signal interval should be monitored. Monitoring 10 sets of entire signal intervals would then result in 20 bins. When a packet is received by the NIC, the receptor records the time of arrival by incrementing the count of packets received in the current interval. The collection of bins from the packet count distribution and this distribution is where
the receptor will attempt to determine the likelihood that a quorum signal exists. The reliance of time synchronization between the hosts is a potential issue for the adversary which they must address.

Once an interval has passed and the receptor is recording packet counts in the next bin of the distribution, the receptor performs a DSP operation such as a Discrete Fourier Transform (DFT) on the distribution. The result of the DFT on the time-series distribution is a frequency-series which can be inspected to determine if a detectable quorum sensing signal is present. As a more pronounced signal is likely to be detected by the network defenders as well as the adversaries’ targeted malware, a low signal to noise ratio will likely be used. As the bandwidth required for the quorum signal is minimal a lower signal to noise ratio will be sufficient to pass the message if the signal is repeatedly measured. The receptor will maintain a frequency-series which corresponds to the signal that it is interested in discovering, henceforth: referred to as the signal from an ideal emitter. The signal from the ideal emitter does not result from an actual emitter embedded in the receptor which is emitting simulated packets, rather it is the frequency-series that would result from a virtual emitter. The DFT from the observed traffic is correlated with the signal from the ideal emitter.

The result of the correlations between the observed signal and the ideal emitter are collected and statistically tested for significance. If the targeted malware only performs this correlation once, the malware will have a result which it can act upon but it does not have an indication the signal’s reliability. If a high-correlation result was returned this still could have been due to random chance: the packets were aligned with the expected signal but a quorum is really not present, or it could have even resulted from a period of relative quiescence on the network in which only a few infected hosts where signaling (e.g. the signal appears to show that a majority of the hosts are infected but in actuality
only the few infected hosts were signaling). By collecting and performing statistical tests on the resulting correlations the targeted malware can estimate the likelihood that the network is actually sufficiently infected.

Depending on the source of the inspiration for bacterial quorum sensing, the adversaries may be embedding multiple signals in the network traffic. In this case the receptor that was previously described will need to be modified slightly. Instead of comparing the observed signal against a single ideal emitter, the signal will be compared against multiple ideal emitters. Multiple collections of correlation coefficients will be maintained which will result in multiple statistical tests. This process will result in additional complexity, but if the targeted malware follows this method, a simple state transition can be performed when the conditions indicate that a community is present. The targeted malware can transition from LCD to HCD even if the original quorum sensing signal is too low to be detected based on the observable network traffic.

3.3 Proposed Applications of Digital Quorum Sensing

Now that the targeted malware has a potential method of communication, the determined adversary may seek additional inspiration from pathogenic bacterial cells. Pathogenic bacteria use quorum sensing to control a wide variety of different behaviors. Bacterial cells use quorum sensing to control initiation of bioluminescence, pathogenesis and production of virulence compounds, formation of a biofilm, and uptake of extracellular DNA via bacterial competence. Some of these behaviors are undoubtedly more attractive sources of inspiration for the determined adversary. At first glance the initiation of pathogenesis appears to be the best target to emulate. In reality, any of
these behaviors can be used as a source of inspiration for possible behaviors which the malware may emulate.

The first behavior to be associated with and described in conjunction with bacterial quorum sensing is bioluminescence [105, 114]. When the LPD of a *Vibrio fischeri* reaches a sufficient density, the bacterial cells begin to emit bioluminescent light. While the ideal of bioluminescent malware may not inspire panic in the hearts of the network defenders if they learned that a particular malware sample had translated to a bioluminescent state. That is unless they understood what bioluminescence is used for by the squid that is hosting a population of *Vibrio harveyi* bacteria. The bioluminescent light is used to cloak the presence of the Hawaiian Bobtail squid at night by preventing the squid from casting a shadow as it hunts. The bioluminescent light hides the shadow of the squid and prevents the squid’s prey from detecting the squid while it forages [114]. The targeted malware could model this strategy and instead of simply using the signal as a sign that it is time to attack the target, it can be used as a signal that it is time to start disrupting operations to prevent the detection of a secondary attack. In this case, the targeted malware is used as a decoy to either mask the presence of a secondary attacker or disrupt operations so as to allow another attack to be successful. Multi-stage malware is not a new concept [111], but it can be combined with quorum sensing to allow multiple types of malware to coordinate its activities in an isolated network.

The most likely pathogenic bacterial cells’ behavior to be mimicked is that of the initiation of pathogenesis. By waiting and attacking a host when a sufficient population is present, the pathogenic bacterial cells are more likely to overwhelm the host’s immune system. A number of pathogenic bacteria use this strategy of waiting until a quorum is present to undergo pathogenesis and being attacking the host. The analogy that targeted malware can use in this situation is relatively straight forward. The targeted malware
waits until the quorum is present to being executing a mission which is designed to disrupt or disable a host’s network. By waiting until a quorum is present, the targeted malware is more likely to succeed in accomplishing the mission and can potentially overwhelm the network defenders.

Similar to the initiation of pathogenesis, bacterial cells use quorum sensing to initiate the production of virulence factors. The initiation of virulence factor production does not cause a bacterial cell to transition to pathogenic state, rather the production enhances the potential survival of the bacterial cell. Virulence factors are compounds which facilitate cellular attachment, allow for immunoevasion, allow for immunosuppression, facilitate cellular entry/exit, or allow the harvesting of resources from the local environment by degrading or destroying host cells. The targeted malware could use these virulence factors as a source of inspiration for controlling additional behaviors such as:

- **Facilitate attachment**: the targeted malware can make modifications to the infected hosts to make them more vulnerable to future attacks to other malware. The targeted malware may have not been implemented with a payload for exploitation but rather they payload modifies the host to make it more susceptible to future attacks. This would be similar bacteria which use quorum sensing to facilitate cellular entry/exit.

- **Immunoevasion**: the targeted malware modifies itself to make it more resistant to detect by the malware by modifying how the host’s anti-malware solutions detect malware on the system.

- **Immunosuppression**: the targeted malware modifies the host to prevent network defenders from effectively responding to detection. It could disable the anti-
malware solution yet modify the reporting agents such that they are reporting that the host if free from infection.

• **Degrading/Destroying Host Cells:** targeted malware, not malware in general, is the focus of this research. A determined adversaries’ objective may be to disrupt a mission rather than destroy a computer system. Destroying computing resources is of limited benefit in the long run unless it is to accomplish a specific objective. This is another case where the potential application is clear: when the quorum is established a self-destruct payload as activated.

Another way in which bacterial cells use quorum sensing is to control the formation of biofilms. Biofilms are a persistent problem for the medical community, as bacterial cells within biofilms are often resistant to antibiotics and the host’s immune system has a difficult time in clearing infectious cells resident within a biofilm. When biofilm production begins, the bacterial cells begin secreting compounds which allow them to attach to surfaces. These chemicals are different than those produced as part of the initiation of virulence factor production which allow the bacteria cells to attach and adhere to host cells. Biofilms can adhere to multiple types of organic and inorganic surfaces and are formed to help protect the bacterial cell from a potentially hostile environment. Once in a biofilm, the bacterial cells can harvest resources or participate in other community behaviors without interference from the environment or a host’s immune system.

Targeted malware can use the idea of a biofilm to its advantage, it can wait until a quorum is present and begin modifying the environment to suit its own nefarious purposes. The changes could be minor when taken individually but if the changes are propagated throughout the entire network, they could work to their advantage. By
delaying, the malware can wait until it is a majority and then perform the modifications, those systems which have not been modified are now the minority. Network defenders will be at a disadvantage in detecting the change as the change occurs across the entire network as the targeted malware transitions to high cell density.

Competence or transformation by natural competence is a process in which bacterial cells transfer plasmids between cells and gain the capacity to tolerate and survive in a changing environment. [10] Horizontal Gene Transfer is the process that is utilized to actually exchange packets of extracellular DNA (i.e. plasmids) between the bacterial cells in the community and quorum sensing is the process that bacterial cells use to coordinate when to become competent (e.g., receptive to Horizontal Gene Transfer). Plasmids offer an advantage to the bacterial cells as they offer a number of conferred advantages such as resistance or tolerance to antibiotics, the ability to metabolize or tolerate toxic metals, etc. The problem faced by bacterial cells is that the exchange requires multiple bacterial cells to coordinate their behavior. A bacterial cell attempt to exchange plasmids when other cells are not competent offers no benefit to the cells. Conversely a bacterial cell that is competent when other cells are not ready to exchange plasmids is also of no benefit.

Competence could be used be targeted malware as a method of determining when it is appropriate to exchange new or updated payloads or other modules. The detection of a quorum signal offers the targeted malware a chance to coordinate functionality updates. Allowing the population as a whole to become competent can allow modules to be exchanged and updated across the entire population of infected hosts. The advantage is that only a few instances of the targeted malware need to have the final payload, when the network is sufficiently infected the malware can spread the final payload throughout the infected hosts. The determined adversary has the advantage of the malware which is
spreading does not have the final payload so if it is detected and analyzed the network defender has no information about the payload. The identification and remediation of the infectious malware does not provide the network defender with any additional information about the intentions or objectives of the determined adversary. This process could also allow the targeted malware to update with the introduction of newer malware in the future.

Sporulation is another process in which determined adversaries can leverage as an additional strategy to implement for targeted malware. Detection and remediation by network defenders is a persistent threat that the author of the targeted malware will need to be prepared to overcome. Malware in general has demonstrated a number of techniques to maintain persistence in an environment; bacteria face a similar issue and one of the methods that are used to overcome a changing and potentially inhospitable environment is to produce spores. The transition to a spore state allows the cell to survive until the environment changes or allow the cell to be moved to a new environment which is more hospitable.

There are a few possible ways in which new strategies for targeted malware could be inspired by the use of quorum sensing. The first is to have the targeted malware continue to monitor for the HCD signal. If the signal appears change from HCD to LCD, it could mean that the targeted malware has been identified and it should transition to a spore-like state. While in this spore-like state, the targeted malware would avoid and alter its behavior so that it is no longer activity interfering with operations and attempt to further mask its presence or execute modules which attempt to ensure that the malware can survive remediation by network defenders. Another aspect of sporulation which could be mimicked by targeted malware would be used when the current operating environment for the malware is not the final target. The current environment could be
an intermediate environment and the targeted malware could be preparing for placement into the final target environment. When the targeted malware detects a quorum is present, it undergoes sporulation and produces malware which is ready for placement in the final environment. The malware could be waiting for an unwilling assistant to assist in deployment to the final environment by infecting USB drives or other types of removable media when they are inserted into infected hosts.

Bacterial quorum sensing is used to produce and modify the behaviors of a number of different types of bacteria. The possible applications that a biologically-inspired determined adversary based on the individual strategies employed bacteria have been discussed in this section, but there are a few different aspects of quorum sensing which have not yet been examined such as inter-species communication. Inter-species communication between bacteria has been discussed; in that bacteria can communicate with different quorum sensing systems and they also can modify their environmental response based on the interspecies and intraspecies portions of the autoinducer.

Beyond utilizing digital quorum sensing to make a multistage type of targeted malware, the use of quorum sensing could be used to signal and coordinate activities between multiple types of malware. It could be possible that multiple determined adversaries want to coordinate their activities on a target high value network but they do not want to expose the details of their respective mission only stay out of the others way. Instead of simply monitoring for their own quorum sensing signal, each malware can search for the others quorum signal and if it detects a signal it modifies its behavior. These modifications could be to adopt an alternate role in the attack on the network such as automatically transitioning into a spore-like state in order to avoid interfering with the others objective or even remove itself from the environment. In the other case, if both
determine adversaries are supporting each others mission, when the quorum is detected the desired payload is executed.

There are a number of different ways in which the autoinducers can be modified to allow communication between multiple families of malware. If the basic approach that bacteria use is followed, the listening interval could be subdivided in which the sub-intervals are allocated to each targeted malware. For example, if previous example which modified the distribution of the upper and lower half of the sub-second is modified to support two determined adversaries. The second could instead be divided into three intervals. The first interval could represent the interspecies communication aspect of the signal. The pattern to be inserted in this interval must be agreed upon by both determined adversaries who are coordinating their activities. This process will be used to measure the total population density of the targeted malware in the environment. The remaining two sub-intervals of the second are each assigned to a determined adversary. Each devises their own packet distribution signal to embed and this portion does not need to be shared between the adversaries. By measuring strength of the signal for the total malware population and comparing that to the strength of the signal for the malware density, the malware could estimate how many of the infected hosts are of the same type. Based on the portion of the LPD, the targeted malware can alter its behavior accordingly.

By using strategies inspired by the uses of bacterial quorum sensing, a determined adversary can potentially implement a large number of devastating attacks on an isolated network. A large number of bacterial quorum sensing systems have been identified and a significant number pathogenic bacterial cells make used of quorum sensing to coordinate the expression of social behaviors.
Chapter 4

Proof of Concept Study

A proof of concept study was the first phase in demonstrating that a digital quorum sensing system can be implemented. This phase of the research consisted of constructing a proof of concept digital quorum sensing emitter and a quorum sensing receptor. The emitter and receptor were then tested in a number of different network situations.

The hypothesis for this phase of the study is: The difference between the packet distributions generated by the digital quorum sensing emitter and normal network traffic is statistically significant. The hypothesis can be converted into a null-hypothesis that is used as the basis of statistical testing during this phase of the research: The is NO statistically significant difference between the packet distributions generated by the digital quorum sensing emitter and normal network traffic.

The aim of the proof of concept was to demonstrate that the emitter and receptor can be implemented and that the receptor can detect the emitted signal. The steps for this phase of the research are illustrated in Figure 4.1 and are detailed as follows:
- A minimal design based on the proposed architecture proposed will be created. At this stage, it is only necessary to demonstrate that a signal can be emitted and detected.

- Based on the minimal design, an autoinducer will be selected.

- Following the selection of a suitable autoinducer, a quorum sensing emitter will be designed and implemented that is capable of emitting the designed quorum sensing signal. Ideally the quorum sensing emitter will be configurable to allow different signals to be emitted.

- The quorum sensing receptor will be designed and implemented which is capable of detecting the emitted quorum sensing signal.

- After the quorum sensing emitter and quorum sensing receptor have been implemented a test will be performed to determine if the quorum sensing receptor can detect the emitted quorum sensing signal.

- The result of the test of the quorum sensing emitter and receptor on the lab network will be analyzed.

- Finally a discussion of the results will be documented and reported.

The proof of concept is not intended to replicate an actual piece of malware that might be deployed by a determined adversary, rather the purpose of this phase is to determine if a quorum sensing system is feasible. As such, the following assumptions have been made to simplify the experiment:

- **Generated traffic is sufficient for illustration.** A key assumption of the PoC study is that the quorum sensing emitter will be generating traffic for the purpose
of quorum sensing. It is assumed that although the quorum sensing signal is being induced by the quorum sensing emitter, the difference between native and generated traffic for detecting a signal will be minimal. It is recognized that there is a difference, but for the purposes of illustrating that a signal can be detected, it is expected that the differences between generated and manipulated traffic are minimal.

- **Quiescent Hosts.** The services and network noise from the hosts are minimized, such that the quorum sensing emitter is the dominant source of network traffic and the traffic is easily identified. This includes the production of traffic initiated by the user; such as traffic resulting from a user browsing the Internet.

The proof of concept was implemented as a minimal set of applications which are used to for demonstration purposes only, it does not reflect the complete architecture but rather it will demonstrate the essential functionality of the architecture.
4.1 Design

A quorum sensing receptor and emitter was implemented based on the architecture discussed in Section 3. The emitter and receptor were implemented as two separate systems, unlike a realistic implementation in which the receptor and emitter would be contained within the same application. The emitter assumed the role of the bacterial quorum sensing system’s autoinducer synthase while the receptors assume the same role. The autoinducer was implemented as a modification to the packet distribution originating from a single host. At no time was the complete autoinduction loop implemented. It purpose of this research was to determine the feasibility of the C2 channel, not implement a system that malware would use to coordinate its activities. The goal was to determine if the signal could be observed in the packet distributions.

In order to demonstrate the functionality of the emitter during this phase of the research, the emitter was implemented as a Win32 application written in C++. The emitter was designed to transmit User Datagram Protocol (UDP) packets with contents that would enable diagnostics and debugging. It requires less overhead to implement UDP packet transmission as session management is not required when transmitted UDP packets. The same diagnostic information could be encoded into TCP packets but the emitter would need to negotiate the opening handshake, encode/transmit the data and close the session. UDP only requires opening a Win32 socket and sending the data to the remote host. The emitter was designed to operate on a host running the Microsoft Windows operating system, as such it was able to utilize the Winsock libraries and Precision Timers from the Microsoft Win32 API.

The receptor was not implemented as a single piece of software, the functionality was divided into two components. Packet collection was handled via the standard
packet collection application tcpdump. A Linux host was used to monitor the traffic as by default the tcpdump monitoring software was included with the operating system. The monitoring server was running the Gentoo Linux-based operating system. By default the Microsoft Windows operating system did not provide network monitoring software such as WinDump and the Windows Packet Capture (WinPCAP) drivers. The second component was another piece of custom written C++ software which served as an analysis module.

The analyzer was based on the architecture detailed in Chapter 3. The analysis began by processing the output of the tcpdump collections and constructing a sliding window of packet counts. The analyzer was supplied with a set of configuration options from the command line in an attempt to determine the signal that was expected. The analyzer functioned by scanning the arrival times of the packets and assigning the packets to bins based on their arrival time. The packet counts were then subjected to a Discrete Fourier Transform (DFT), in order to extract the frequencies contained within the packet distributions.

When the analyzer was initialized, it would inspect the configuration that was supplied and construct an “ideal” emitter based on the expected distributions. It is considered an ideal emitter because it does not contain any extraneous packets transmitted by the host or contain any noise resulting from the network devices between the emitter and receptor. This “ideal” emitter generates a packet distribution and the resulting distribution was analyzed with a DFT to construct a set of expected frequencies.

A Pearson correlation coefficient was calculated by measuring the frequency components of the observed DFT and the ideal DFT. The calculated correlation coefficients were collected and recorded overtime allowing statistical tests to be
performed. The null-hypothesis for the statistical tests was that there is no statistically significant difference between the differences between the traffic and the ideal emitter when the signal is present on the network and the signal is absent.

An important factor to consider at this phase of the research is that the proof of concept is not intended to be a full implementation of the quorum sensing emitters and receptors. It was only intended to determine if continuing with more detailed studies would potentially result in useful information about the C2 channel.

4.2 Results

A set of experiments were conducted with the designed quorum sensing emitter and receptors. The emitter was utilized in a number of different network configuration. The transmitted packets were captured observed on the receptor with tcpdump and analyzed with the quorum sensing receptor applications. The set of receptors and emitters were used in four different network configurations; flat virtualized local network, flat wireless network, receptor-emitter pairs on geographically close Internet connections and receptor-emitter pairs on geographically distant Internet connections.

4.2.1 Normal Web Browsing Traffic (Control)

Before the experiments were performed, a control case was tested. A standard Microsoft Windows host was connected to the local Internet and the experimenter generated normal usage traffic by browsing the Internet for 15 minutes. The host did not contain an emitter generating traffic in the expected quorum sensing signal profile. Three different collections were performed and the observed traffic was analyzed by the quorum sensing
receptor. The basic statistics of the resulting correlation coefficients are presented in Table 4.1.

### 4.2.2 Flat Virtualized Network

The first experimental test of the emitter and receptor was performed in a flat virtual environment. Two different test signals were used during the experiment. The first signal was a constant emitter 20 second period and the emitter was configured to generate 1 packet per second. The second signal was a pulsed emitter, also with a 20 second period. During the first 18 seconds the emitter would generate packets at a 1 packet per second rate while during the final 2 seconds of the period the emitter would generate packets at a rate of 10 packets per second.

Packets were generated on a local host and transmitted to a virtual guest machine listening on a private virtual network which was collecting packets via tcpdump. The null-hypothesis for this phase of the research was that there was no statistically significant difference in the observed signal between a constant static emitter and a pulsed-emitter. In essence this was a test of the quorum sensing detector to determine

---

**Table 4.1: Comparison of correlation coefficient statistics between an expected quorum signal and normal network traffic in a WiFi network**

<table>
<thead>
<tr>
<th></th>
<th>Traffic with No Signal (Collection 1)</th>
<th>Traffic with No Signal (Collection 2)</th>
<th>Traffic with No Signal (Collection 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>Min.</td>
<td>0.407</td>
<td>0.411</td>
<td>0.365</td>
</tr>
<tr>
<td>Max.</td>
<td>0.520</td>
<td>0.563</td>
<td>0.460</td>
</tr>
<tr>
<td>Mean</td>
<td>0.464</td>
<td>0.484</td>
<td>0.406</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.032</td>
<td>0.050</td>
<td>0.027</td>
</tr>
</tbody>
</table>
if the receptor can distinguish between a periodic signal in the network traffic and a pulsed-signal in the network traffic that matched the expected quorum sensing signal.

The data was collected and 96 DFTs performed on the observed packet distributions and the following statistical test results were calculated; quorum signal correlation coefficient mean = 0.864 (X1) and standard deviation = 0.006 (s1), Constant Signal correlation coefficient mean (X2) = 0.940 and standard deviation = 0.013 (s2). A Two Sample t-Test was performed, with the Null-hypothesis being that there was no difference between the signals. The null-hypothesis was rejected (t value was greater than the t with 95% confidence), and it was determined that the samples were statistically different. As the null-hypothesis was rejected it means that the two signals were statistically significant or the emitter was able to discern that there was a statistically significant difference in the observed traffic. Based on this positive result it was determined that the remainder of the experiments should proceed.

4.2.3 Local Wireless Network

Another experiment was conducted to characterize the quorum sensing signal in a more realistic environment. The previous experiment was conducted in an isolated virtual environment in which there were only a small number of hosts participating and influencing the traffic on the network. The subsequent experiment was moved to an encrypted wireless network which contained multiple hosts. The number of hosts participating in the experiment was increased to four (3 emitters and 1 receptor).

The pulsed signal previously discussed in Section 4.2.2 was used as the expected quorum sensing signal for these experiments. The statistics of the collection are presented in Table 4.2. The null-hypothesis for this stage of the experiment was that
Table 4.2: Comparison of multiple quorum signal emitters on a WiFi network emitting the same quorum sensing signals

<table>
<thead>
<tr>
<th>Quorum Signal Emitter (Host 1)</th>
<th>Quorum Signal Emitter (Host 2)</th>
<th>Quorum Signal Emitter (Host 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>136</td>
<td>136</td>
</tr>
<tr>
<td>Min.</td>
<td>0.914</td>
<td>0.871</td>
</tr>
<tr>
<td>Max.</td>
<td>0.974</td>
<td>0.937</td>
</tr>
<tr>
<td>Mean</td>
<td>0.951</td>
<td>0.898</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.016</td>
<td>0.016</td>
</tr>
</tbody>
</table>

Table 4.3: Comparison of multiple emitters with multiple signals on a WiFi network emitting different quorum sensing signals

<table>
<thead>
<tr>
<th>Quorum Signal Emitter 1 (Signal 1)</th>
<th>Quorum Signal Emitter 2 (Signal 1)</th>
<th>Quorum Signal Emitter 3 (Signal 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>137</td>
<td>137</td>
</tr>
<tr>
<td>Min.</td>
<td>0.909</td>
<td>0.908</td>
</tr>
<tr>
<td>Max.</td>
<td>0.976</td>
<td>0.975</td>
</tr>
<tr>
<td>Mean</td>
<td>0.955</td>
<td>0.950</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.016</td>
<td>0.015</td>
</tr>
</tbody>
</table>

There is no statistically significant difference between the traffic containing normal internet browsing patterns and traffic originating from a host with an embedded quorum sensing signal. Section 4.2.1 contains the correlation coefficient statistics used for the set of t-tests. The null-hypothesis was rejected based on the coefficient statistics with a significance level of 95% (alpha=0.05). The quorum sensing signal is statistically significant from cases in which no signal is present.

A second set of experiments were performed in which two different quorum sensing signals were generated by the emitters. Two of the emitters generated packets based on the pulsed signal discussed in Section ?? while the third emitter was configured to generate a different signal. The new signal was similar to the original signal except that it had a 10 second period instead of a 20 second period.
The experiment was repeated in this new configuration and the statistical characteristics of the signal are presented in Table 4.3. Statistical testing was performed with the null-hypothesis being that there are no differences between the signals being emitted. The null-hypothesis was rejected as the difference between the two separate signals was discernible and both signals were distinguishable from each other as well as the normal Internet browsing case.

### 4.2.4 Internet-based Hosts

The final set of experiments performed during this phase of the research attempted to determine if the signal could be detected across Internet scale distances. Previous experiments on the proof of concept were conducted when the separation between the receptor and emitter was only 2 hops in a flat networking environment (i.e., no routing or network isolation). Packet captures were collected from a web server host on the Internet under the experimenters control. Two different sets of collections were performed. In the first case the emitters were placed in various locations around the national capital region (e.g., Washington DC) and in the second case the emitters were placed on hosts located on the eastern coast of Florida (e.g., the Florida Space Coast). The emitters were configured to the generate the pulsed quorum sensing signal traffic (see Section 4.2.2 for pattern details).

The summary statistics of the distributions for traffic originating the the national capital region and the local Internet are shown in Table 4.4. Statistical testing was performed with the null-hypothesis being that there are no differences between the signals being emitted from across the Internet to those that are generated when a user browses the Internet (see Table 4.1 for captured packet statistics). The null-hypothesis
Table 4.4: Comparison of multiple quorum signal emitters emitting the same quorum sensing signal located in a different geographic regions

<table>
<thead>
<tr>
<th>National Capital Region Emitter 1</th>
<th>National Capital Region Emitter 2</th>
<th>National Capital Region Emitter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>121</td>
<td>89</td>
</tr>
<tr>
<td>Min.</td>
<td>0.884</td>
<td>0.940</td>
</tr>
<tr>
<td>Max.</td>
<td>0.995</td>
<td>0.998</td>
</tr>
<tr>
<td>Mean</td>
<td>0.935</td>
<td>0.981</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.028</td>
<td>0.018</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Local Internet Emitter 1</th>
<th>Local Internet Emitter 2</th>
<th>Local Internet Emitter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samples</td>
<td>123</td>
<td>126</td>
</tr>
<tr>
<td>Min.</td>
<td>0.928</td>
<td>0.932</td>
</tr>
<tr>
<td>Max.</td>
<td>0.983</td>
<td>0.989</td>
</tr>
<tr>
<td>Mean</td>
<td>0.962</td>
<td>0.970</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>0.012</td>
<td>0.013</td>
</tr>
</tbody>
</table>

was rejected as a statistically significant difference (using a significant level of 95%) was observed between the quorum sensing signal and the Internet browsing traffic.
4.3 Discussion

The proof of concept was used to demonstrate that it might be possible that a quorum sensing signal could be used as a C2 channel. This work was accepted and presented at the MALCON’2014 conference [54]. After reviewing the statistical tests it was determined that the t-test should not have been used for the experiments. T-tests are appropriate when the data is parametric and network packet counts are typically non-parametric. The results of the experiments were not revisited and revised, instead a completely new and more robust set of statistical tests were used in the next two phases of the experiments.
Chapter 5

Prototype Study

At the conclusion of the Proof of Concept phase of this study, it was deemed that digital quorum sensing could potentially be leveraged as a C2 scheme. Based on this conclusion, further research was warranted. A prototype based on the lessons learned from the implementation of the proof of concept was constructed and tested on an isolated network. The most important aspect of this phase of research is that the emitter is no longer generating novel packets, the emitter is essentially “living off the land”, it is using traffic emitted by processing running on the host and embedding the quorum sensing signal. Technologies such as Deep Packet Inspection or Netflow analysis cannot be used to identify the quorum sensing signal. This section describes the studies that were conducted during the second phase of testing.

The hypothesis for this phase of the experiment was *The strength of the signal generated by the quorum sensing emitter is detectable in the presence of network noise*. Unlike the proof of concept phase of this research, a direct conversion between the hypothesis and the null-hypothesis was not performed. Rather an alternate set of null-hypotheses were used. The following null-hypothesis was used as the primary null-
hypothesis, \textit{There are no statistically significant deviations in the packet distributions on the network}. This null-hypothesis was used as there are situations in which a network defender may only have existing network traffic to monitor for disturbances, “clean” traffic to use as a control for comparisons may not be available. If the first null-hypothesis is rejected, which means that there are statistically significant anomalies in the data, then a second null-hypothesis will be used for further analysis. The secondary null-hypothesis is as follows; \textit{There is no statistically significant difference in the traffic emitted from a single host when the emitter is modifying traffic to contain a statistically significant signal and when the emitter is not interfering with packets being emitted by the host}. A test which results in the rejection of this null-hypothesis is important in that it demonstrates that a statistically significant anomaly is present in the global packet distribution but the same anomaly is not statistically significant in the traffic being emitted from individual hosts.

The aim of the Prototype is to refine and expand upon the work performed as part of the initial study. The types and realism of the tests used for the prototype were enhanced
during this phase of the research. The implementation of quorum sensing emitter and quorum sensing receptor will be refined or re-implemented as needed based on lessons learned from the proof of concept. Specifically the steps for this study illustrated in Figure 5.1 and are as follows;

- Lessons learned from the proof of concept phase were reviewed for architectural deficiencies.

- The digital quorum sensing architecture was reviewed against the proof of concept architecture and lessons learned from the proof of concept and revisions were made.

- The quorum sensing emitter and quorum sensing receptor were modified and reimplemented based on required functionality.

- Quorum sensing emitter and quorum sensing receptor were tested in a more realistic lab environment. Existing network traffic (e.g., noise) from other applications and services was not be minimized.

As the prototype was intended to produce higher fidelity results, the two simplifying assumptions made during the Proof of Concept were deemed to be inadequate. Inducing traffic patterns from the host by generating existing novel network traffic is no longer sufficient for testing. During this phase, the quorum sensing emitter must make use of existing network traffic and only modify the transmission times of the packets being emitted from the host. Similarly, a network of quiescent hosts is no longer enforced. Normal network traffic must be generated by the hosts for the quorum sensing emitter to act upon, as such it is now a requirement that all hosts in the network actively generate network traffic.
5.1 Design

The proof of concept phase of this work indicated that the digital quorum sensing C2 channel could potentially be used to signal overall infection state information on a network. In order to further characterize the digital quorum sensing signal a prototype was constructed. The proof of concept was designed to determine if a signal could be generated and used as a C2 channel, but as it was only a proof of concept, it was limited in both functionality and fidelity.

5.1.1 Design Revisions

To further explore the C2 channel a more robust design was implemented in order to address the modified limitations. This redesign affected both the quorum sensing emitter and quorum sensing receptor. Before a new emitter and receptor can be designed, a quorum sensing signal must be selected. The selection and description of the autoinducer used for the prototype is discussed in section 5.1.1.

The emitter was redesigned to allow existing traffic being transmitted from the host to periodically be delayed instead of emitting packets in the expected distribution. A detailed discussion on the design and supporting rationale is discussed in section 5.1.1. When the statistical testing methods which the proof of concept relied upon when examined, it was determined that the testing methods were inadequate and needed to be revised. The revisions to the receptor are discussed in section 5.1.1.

Revised Autoinducer

The quorum sensing signal (i.e., the autoinducer) was changed for the prototype. The quiescent host limitation for this phase of the study was abandoned. It was expected
that the signal would be embedded in existing traffic generated by the host. Selection of a quorum sensing signal is critical to the potential success or failure of a determined adversaries mission.

The determined adversary’s selection of an autoinducer (e.g., the quorum sensing signal) is critical to the success or failure of their mission. Ideally the adversary will select an autoinducer which is not novel to the environment. They will leverage existing resources on the network and subvert the existing communications for their own use. An adversary could design their own custom protocol which could relay information to all of the compromised hosts, but in doing so the network defenders could identify this traffic and subsequently identify all of the infected hosts.

As discussed in Section 3, the signal can be embedded in the overall network traffic and the signal has a time dependent effect on the distribution. The malware can slightly skew the distribution in a way that can be detected globally but the skew from individual contributions from the infected hosts on the network skew are within statistical variation of normal traffic. A few different examples of possible autoinducers include the overall number of packets seen during a specific time-interval or the number of fragmented packets during a time-interval.

For this research it was determined that a small notch which altered the overall packet distribution could be used. If the notch caused a minor disturbance in the statistical distribution of packet counts, it may not be noticed by network defenders. If the disturbance is noticed by network defenders due to the potential impacts of acting upon false-positives the network defenders may hesitate to respond.

In order to more precisely define and characterize the autoinducer, the following terms and definitions are used:
- **Window**: An interval of time in which the autoinducer signal may be embedded. It is not guaranteed that the signal will be present during this interval. For example, a window could be 1 second, 5 seconds or even an hour.

- **Period**: A sub-interval of time within a window. For example, if a window is 1 second in duration a period size of 50 milliseconds would result in 20 periods.

- **Notch**: A notch is a set of periods in which packets scheduled to be transmitted during this period will be delayed. The size of the notch cannot be less than 1 period. For example a notch of 100 milliseconds will be 2 periods in length if the period size is 50 milliseconds. During this notch, any packets scheduled to be transmitted will be delayed by 25 milliseconds. The application of notches to the packet scheduler is probabilistic. The probability of a delay occurring during a specific window is configurable.

- **Epoch**: The time at which the first window and period starts. The epoch is used as the starting point for the packet count bins. By knowing the current time and the epoch time, the current window and period can be calculated using integer division.

If the adversary attempts to implement an autoinducer based on altering the distribution of packet counts per period, there are a few parameters which will need to be varied and tuned. The primary area of interest is associated with the notch and the probability that it will exist during a given window.
Figure 5.2: Normal Transmission Packet Flow through the Linux Kernel

Revised Emitter

There are several different designs which can be used to implement the Quorum Sensing emitter. Key requirements for the design of the emitter are as follows: 1) the design must be able to manipulate the traffic being transmitted by the host, 2) the design should not affect the stability of the system, and 3) the design should allow for easy reconfiguration of the transmission properties of the network stack with minimal overhead.
Applications operating out side of the kernel and relying on normal networking APIs are unable to directly transmit packets to the network. There are several layers of abstraction which prevent packets from being directly transmitted by an application. As illustrated in figure 5.2, the normal transmission process of packets on a Linux host is as follows;

1. **Application assembly:** The application assembles a packet for transmission and provides the binary data to the local network library for transmission.

2. **Network Libraries:** The network libraries encapsulate the binary data and provide it to the Linux kernel for transmission.

3. **Kernel Packet Queue:** Packets are not immediately transmitted when relived by the kernel, instead the packet is placed into the packet queue where packet prioritization and re-ordering can occur.

4. **qdisc Scheduling:** The packet is scheduled for transmission by the network queueing discipline (qdisc) and the next scheduled packet is readied for transmission.

5. **NIC Transmission:** The packet is removed from the packet queue and given to the network interface card for transmission.

The designs reviewed include: a custom kernel module, a network proxy, a custom networking queueing discipline and a network queueing discipline scheduler. Each of these possible design choices will be discussed in term with pros and cons of the respective design. At the conclusion of the design section; the implementation that was chosen and the rationale for that decision will be detailed.
The first design reviewed was a custom kernel module. A kernel module could be developed which modifies when packets are transmitted by the host. The module would receive packets from the kernel before they are added to the packet queue for transmission. This custom module would selectively delay the packets before they are added to the packet queue within the kernel.

The custom kernel module would allow low-level access to the networking queue within the GNU/Linux kernel. It would allow for the transmission characteristics of all of the packets emitted by the host to be inspected and changed. Another benefit is
that this design would most accurately reflect how a determined adversary would likely implement a quorum sensing emitter. Unfortunately, for experimental purposes this design has a few weaknesses in that it has the potential to affect the stability of the overall system if mistakes are made during the implementation process. Performing system traces and debugging at the kernel level increases the complexity of the emitter. Lastly the ability to log events is limited for software operating within the kernel. Implementing the quorum sensing emitter as a custom kernel module would enable to experiments to be conducted but the design would require significant development effort without enhancing the experimental results.

Another design considered was a network proxy which applications and services would relay their packets through prior to transmission on the network. A local network
proxy sound be implemented which allows the local applications and services to connect to it when transmitting packets. The network proxy would modify the transmission properties in order to embed the desired quorum signal.

This design has a benefit that none of the application code is running in the kernel or as privileged code. All of the packets could be inspected and manipulated in user-space. All of the transmission characteristics of the packets could be inspected and changed for applications and services configured to use the proxy. A misconfiguration of the network proxy would not affect the stability of the entire system, only the applications configured to use the proxy. Logging for the network proxy would not be limited to the functionality within the kernel, so logging could be as detailed as required. There are a few drawbacks to this design. The first drawback is that not all of the transmission characteristics of traffic being emitted from the host can be inspected or changed, for example the packets being emitted by as part of the Address Resolution Protocol (ARP) would not be able to be manipulated. Each application on the server must be individually configured to use the proxy. The settings for specifying a system-wide proxy could be utilized but applications can choose to ignore these. The use of a local network proxy would require that the packets being transmitted by the host would need to make two passes through the network queue; first time is from the application to the network proxy, and the second time is from the network proxy out to the network. This extra traversal of the networking stack is likely to introduce additional noise into the network traffic which would need to be characterized.

The GNU/Linux kernel makes use of queueing disciplines (also referenced as a qdisc in kernel documentation [27, 79]) to control the transmission properties of packets being transmitted by the host. When an application creates a packet to be transmitted, the packet is handed off to the networking libraries. These libraries interact with the kernel
Figure 5.5: Implementation of DQS Emitter as a new Queueing Discipline

which places the packet in a packet queue. The queueing disciplines filter and schedule the packets to determine the sequence of packets transmitted by the host. For example, a Token Bucket Filter (TBF) queueing discipline can be used to throttle specific traffic being emitted from the host to ensure that a single service is not dominating the network traffic.

The next possible design of a quorum sensing emitter is the emitter as a custom queueing discipline. Like the previously discussed option with a custom kernel module, the custom queueing discipline would work within the kernel to manipulate the
transmission properties of the packets emitted by the host. It has the ability to inspect and control the transmission properties of the packets from any packet being emitted by the host. All of the packets go through the network queue before transmission. This design has a similar set of drawbacks to the custom kernel module; an incorrect implementation would affect the stability of the entire host. There are also similar difficulties associated with debugging and logging because the code is running within the kernel.

The final design considered was an external network queueing discipline scheduler. Instead of implementing a custom queueing discipline an external program could periodically manipulate the characteristics of the queueing discipline through tools and
interfaces available on the operating system. For example, a scheduler external to the kernel could activate every 20 ms and change the transmission properties of the network queue on a predetermined schedule. In addition to the TBF previously discussed, there is a Network Emulator (also known as netem)[4] queueing discipline which can be used as a method to delay packets before they are transmitted.

The network queueing discipline scheduler design can inspect and manipulate the transmission characteristics of all packets being emitted by the host. The scheduler is running in a privileged mode to access the functions to reconfigure the network queue; however, it is not running within the kernel. As such if the scheduler is incorrectly implemented, it is unlikely to affect the overall stability of the system. If an incorrect configuration is applied, the queueing discipline filters are trivial to remove. The details available for debugging are limited, but logging about the transitions of the reconfiguration of the queueing discipline are achievable as the scheduler is not running within the kernel. For example, the external scheduler is aware of its internal state but as it is relying on exposed APIs it cannot directly access the contents of the network queue. The scheduler cannot precisely determine the state of the network queue without raw access to the network queue, it cannot monitor and log changes to the packet queue.

Although a determined adversary can use any of the previously discussed methods to implement a quorum sensing emitter, selecting the method which the adversary is most likely to use is not a key design requirement. Understanding the characteristics of the communication channel is more important than creating and mimicking the design potentially used by determined adversaries. The choice of the design will impose additional limitations which will need to be addressed as part of the study.
Algorithm 5.1: Quorum Sensing Emitter Scheduling Algorithm

window\textsubscript{current} ← 0 ; 
load emitter configuration; 
validate emitter configuration; 
time\textsubscript{current} ← get\textunderscore time(system); 
sleep( time\textsubscript{epoch} - time\textsubscript{current} ); 
while window\textsubscript{current} < window\textsubscript{final} do 
\hspace{1em} time\textsubscript{current} ← get\textunderscore time(system); 
\hspace{1em} window\textsubscript{current} ← int( ( time\textsubscript{current} - time\textsubscript{epoch} ) / size\textsubscript{window} ); 
\hspace{1em} period\textsubscript{current} ← int( ( time\textsubscript{current} - time\textsubscript{epoch} ) / size\textsubscript{period} ) mod size\textsubscript{window} ; 
\hspace{1em} profile\textsubscript{target} ← lookup( period\textsubscript{current} ); 
\hspace{1em} p ← random(system); 
\hspace{1em} if profile\textsubscript{target}.transition\_probability < p then 
\hspace{2em} if profile\textsubscript{current} != profile\textsubscript{target} then 
\hspace{3em} update\textunderscore network\textunderscore configuration( system, profile\textsubscript{target} ); 
\hspace{2em} end 
\hspace{1em} end 
\hspace{1em} time\textsubscript{period+1} ← get\textunderscore time( period.next, time\textsubscript{epoch} ); 
\hspace{1em} time\textsubscript{current} ← get\textunderscore time(system); 
\hspace{1em} sleep( time\textsubscript{period+1} - time\textsubscript{current} ); 
end 

Based on the benefits and drawbacks of each of the possible designs, the networking queueing discipline scheduler was selected as the design to be implemented. The scheduler was chosen as it is the solution which covers all three of the key requirements presented at the beginning of this section. This designed allows for the transmission
properties of all packets emitted by the host to be modified, not just a subset as the network proxy would have allowed. The scheduler design does not have any custom code operating in the kernel as both the kernel module and the custom network queueing discipline would have required. If the scheduler affects the system stability, it is utilizing existing networking tools so the corrupted configuration can be removed with a single command. The scheduler, like the network proxy, can be implemented such that it is easy to reconfigure.

The algorithm that was implemented is illustrated in Figure 5.1. The emitter scheduler starts by loading the profiles from the configuration files and validated the associated parameters. The determines the current system time (time\textsubscript{current}) and compares value to that of the epoch time (time\textsubscript{epoch}) specified in the emitter configuration. If the epoch is in the past, the scheduler immediately begins modifying the state of the queueing discipline, otherwise the scheduler sleeps until the desired epoch.

Once the main execution loop of the scheduler has started the scheduler again looks up the current system time and based on this time determines the current window (window\textsubscript{current}) and period (period\textsubscript{current}). After the period has been determined, the target transmission profile is looked up from the configuration. The scheduler will then get a random number (p) from the system in the range [0.0,1.0] which is used to determine if the target transmission profile (profile\textsubscript{target}) will be applied.

Each period is only associated with a single transmission profile, but the transitions between the profiles are probabilistically determined. A profile with a transition probability of 1.0 will always be applied while a profile with a transition probability of 0.0 will never be applied. If the random number is less than the transition probability, the profile may be applied by the scheduler. Before attempting to apply the transmission
profile, the scheduler will first check to see if it is not already in that state. If it is, no attempt to transition will be made.

If the scheduler determined that the transmission profile should be changed, this reconfiguration could take a non-trivial amount of time and could take longer than a single period, so the scheduler again queries the system for the current time. Using the start time of the beginning of the next period and the current time, the scheduler determines the duration of time required to sleep. After sleeping, the scheduler continues with the main execute loop.

**Host Traffic Stimulation**

The quiescent host assumption was removed for this phase of the study in order to characterize the quorum sensing signal in a more realistic environment. During this phase, it was expected that hosts would actively be generating network traffic, and this traffic would be modified by embedding the quorum sensing signal.

To ensure that hosts were generating traffic during experimental collections a Ruby script was created to cause a web browser (e.g., Mozilla Firefox) on the host to randomly select links from a remote website. The Ruby script uses the Watir ruby gem[6] as well as the Selenium Webdriver to simulate a user browsing a website. A website containing the Java SE API documentation[2] was used as the remote website. The Java SE API documentation was selected as it is a large set of static interconnected documents. The script used a psuedo-random number generator to select a link to follow from the website. The static nature of the website and the use of a pseudo-random number generator allows tests to be repeatable in nature.

As the script runs during the course of a collection, it records the URL that was requested along with a timestamp of the request. Due to the asynchronous nature of
web, the script was designed to wait until the response was complete before attempting to select another URL to follow on the website. Because the experiment was designed to operate in an isolated network environment, the script contained a URL filter to prevent it from selecting URLs that were outside of a target domain. In the case of the Java SE API documentation, it was limited to the hosting web server and prevented the script from attempting to load resources from the oracle.com domain which was referenced in the site. If an error occurred when the script attempted to retrieve the web page, the script would handle the exception and restart at the initial URL for the target domain. This design allowed the script to continue generating traffic for the host even if an exception was encountered.

When the script terminated, it generated a second file which contained a list of the durations observed for the browser requests. The duration was the number of microseconds between when the request was submitted by the browser to the website and the response had completed processing.

Revised Receptor

As a result of the analysis performed on the proof of concept receptor, it was determined that the statistical testing was inadequate. A new process for performing statistical tests was devised to account for the non-parametric nature of the packet counts for the network data. Figure 5.7 illustrates the new design for the receptor and the revised statistical tests are presented in this section.

In the proof of concept, there were three different components which collectively provide the functionality of the quorum sensing receptor (though in actual deployment by a determined adversary this would likely comprise just one shared object). The components of the receptor pipeline are packet collection, feature extraction, and
Figure 5.7: Revised Quorum Sensing Receptor Design

1. tcpdump (Collect ALL Received by NIC)
   - Extract Packet Times
   - Count Packets by Bin
   - Packet Statistics are Collected (consolidated into a single dataset)
   - Perform Friedman ANOVA Test (use standard significance of 5% with Bonferroni's Correction)

2. Calculate Median of each collection (median used due to presence of outliers)
   - Perform Wilcoxon Rank Sum Test
   - Record Periods which are Statistically Significantly different based on Wilcoxon.
statistical analysis. Packet collection is provided by tcpdump[80] running in privileged mode so all of the traffic hitting the network interface can be captured. The quorum sensing signal is embedded in the time of transmission for the packets, tcpdump is configured to capture the default snap length (e.g., the first 68 bytes) of each packet along with a time stamp. The time stamps and the packet headers are recorded to a log file.

A set of C++ applications and Ruby scripts were developed to extract the features of interest. The analysis tools used the period and window size as configuration parameters and converted the packet log files into a time sequence of packet counts. Two different data sets resulted from this extraction. The first data set is a packet count sequence which can be used to perform a Discrete Fourier Transform (DFT) to determine if a signal is present. The second data set groups the packet counts by the period in which they occur and is used for performing statistical hypothesis testing. Packets emitted during each of those windows would be part of a single period, the counts of all of transmitted/received packets would be tallied into a single list of periods and packet counts that occurred during each period. For example, if a data collection was run for 2 hours and the window size is 1 second, it would result in 7,200 windows. Packets emitted during window 1/period 4 and window 7,101/period 4 would be tallied in the packet count from period 4.

In order to determine if a potential signal is present a number of different statistical tests are performed on the packet counts. Packets are monitored for a period of time and the counts are recorded by period as previously described. Collections are then repeated so statistics about the packet counts per period can be collected. As the emitter is modifying the transmission times of packets being emitted by a host the independent variable is time and the dependent variable are the packet counts. The emitter is only
modifying the transmission times of a subset of packets, as such the first null-hypothesis will be of the form that there is no statistically-significant difference in the packet distribution for each of the periods during a given collection.

The analysis of the data begins with the Kolmogorov-Smirnov Test (KS-test) [84, 125]. The KS-test is used to determine if the data follows a parametric distribution. As the receptor is monitoring network traffic, it was not anticipated that the packet counts will be parametric. Non-parametric data requires different statistical tests so the Friedman’s ANOVA test is used to determine if there are statistically significant differences in the data set which could indicate the presence of a signal [58, 59, 60]. Before determining if the p-value result from the Freidman test is significant, the Bonferroni correction to the significance level must be applied [25, 49, 48]. The idea behind the Bonferroni test is that when multiple tests are performed, the probability of statistically anomalous events occurring increases, a correction needs to be applied. In order to counteract the likelihood of making accepting an incorrect result due to the increased likelihood of anomalous events, the significance level needs to be adjusted by n, where n is the number of comparisons between data sets [25, 49, 48]. If Bonferroni corrected Friedman’s ANOVA test indicates that there is a statistically-significant difference in the statistics of the periods, the Wilcoxon signed-rank sum test will be used to identify the specific periods which are statistically-significantly different. The Wilcoxon signed-rank sum [141] test is a pair-wise test which means that a potentially large number of pair-wise comparisons are required. For example, if there are 100 periods in a given collection, the number of required pair-wise comparisons that must be evaluated is 100 * 99 / 2. Field and Hole provide a summary of all of the statistical tests used by the receptor when testing for statistical significance [52]. Matlab was used as the analysis tool for performing the statistical tests [3, 5, 1].
In order to limit the number of comparisons, each period will be statistically tested against the median of the periods. The median is used for comparisons as the network packet counts have an non-parametric distribution and the presence of outliers have been experimentally observed. The median of the period will be utilized instead of the mean for producing a data set to characterize the periods for the Wilcoxon signed-rank sum test. This trade off will reduce the number of comparisons from approximately $n^2$ to $n$. This decrease will occur at the expense information about potential signal clusters. This is not seen as a significant weakness to the design as malware relying upon these tests to determine if a signal is present will likely make similar simplifications to the tests in order to avoid performing large computations on the host. Large computations could impact performance in a way that alerts network defenders that a host has been compromised.

### 5.1.2 Environment Setup

A small isolated network environment was created to perform data collections. The network was physically isolated from all other networks to prevent any external sources of noise. Five hosts were added to the network; one acting as a server and the others are acting as clients as illustrated in Figure 5.8. Each host was running a 32-bit Ubuntu 14.04 LTS Desktop image.

In order to determine the final configuration for the quorum sensing emitters, preliminary experiments were performed in an isolated virtual environment consisting of two hosts. One host acted as a server and provided access to the Java API specifications via a web server. The second host acted as a client which requested various web pages from the server. The quorum sensing scheduler was installed on the client. The emitter
was tested on a Ubuntu 14.04 Desktop image. The client used tcpdump to monitor the traffic that was being transmitted. tcpdump was configured to only collect packet header information (i.e., the first 68 bytes of the packet) about all packets being transmitted from and reviewed by the host. Only the packet header information was collected as the collection period for each experimental run was 6 hours in duration.

The isolated virtual environment was first used to experimentally determine the minimum period size that should be used with the quorum sensing scheduler. The scheduler was configured to run for 6 hours with a period size of 10 ms. The time required to reconfigured the qdisc was recorded. If the reconfiguration operation required more than 10 ms, the scheduler would adjust its place in the schedule or profiles (which may include skipping the intermediate profiles).
The times required for the reconfiguration were recorded. The median time for implementing a transmission delay by manipulating the dqisc was 5.030 ms, while the time to remove the delay is 5.118 ms. This time was well within the initially proposed 10 ms period size. The 99th percentile of the transition times for implementing a delaying filter is 9.504 ms while the 99th percentile for removing the delaying filter is 5.030. Based on these observations it was determined that keeping a 10 ms period size was appropriate. A 10 ms could allow periods to be small enough that the queuing discipline could transition to a new profile and back to its original state almost all of the time.

The external scheduler was configured to manipulate the delays in the packets being transmitted from the host. In the isolated virtual environment; only the two hosts were communicating as such a notch size of 20 ms with a 5 ms delay was used for the tests. It was anticipated that the non-virtual network would exhibit more noise so the experiments so the notch window size and delay was doubled to account for the introduction of additional hardware between the emitter and receptor.

**Test Sequence**

Different data collection periods of 15 minutes, 30 minutes, 1 hour, 3 hours, and 6 hours were recorded on the virtual and non-virtual networks during setup validation. As a result of several tests of the new quorum sensing emitter, it was experimentally determined that data collections of six hours would provide sufficient traffic to allow the receptor to perform statistical tests on the observed network traffic.

The external scheduler was written in a way that allows a number of different configurations to be performed. The primary test variable for this experiment was the probability that a notch would be inserted during a given experimental collection, abbreviated as $P_{\text{notch}}$. To allow for the null-hypothesis to be statistically tested, a small
number of different configurations were tested during this phase. Packet collections were performed for the hosts when the emitters were modifying traffic and when they were not modifying traffic. A number of different tests were again performed by varying the value of $P_{\text{notch}}$ = 1.0, 0.5, and 0.25.

5.2 Results

The isolated network environment was setup on a small network as described in Section 5.1. During verification of the network it was noticed that detection of the quorum sensing signal is sensitive to synchronization of the clocks between all of the hosts, further details will be discussed in Section 5.2.1. The Network Time Protocol (NTP) service was installed on all of the hosts, and the clients on the network were configured to rely upon the server as a NTP time source.

In all of the subsequent experiments, the following configuration is used:

- **Window Size**: 1000 ms (1 sec)
- **Period Size**: 10 ms (i.e., 100 periods per window)
- **Scheduler States**: State 0 - qdisc fifo.fast (default packet scheduler), State 1 - qdisc netem delay
- **Notch Characteristics**: Notch starts at period 51 and potentially exists for 4 periods. Which means that the notch starts at 500 ms into the 1000 ms window with a 40 ms duration (i.e., periods [51,54]). During the notch, the packets being emitted are delayed by 20 ms.

The probability of a delay being inserted during any specific window is given by $P_{\text{notch}}$ and this value was varied during the experiments. The $P_{\text{notch}}$ values used during
this phase of the research were: 1.0, 0.5, 0.25. Another configuration was used in which the emitter was not running, which was roughly equivalent to a $P_{\text{notch}}=0.0$. It was expected that there would be minimal difference between a system running without the quorum sensing emitter being active and a system running with the quorum sensing scheduler configured with $P_{\text{notch}}=0.0$. The only difference is that the scheduler would check its internal state every 10 ms and determine that nothing needed to be changed.

5.2.1 Quorum Sensing Signal with $P_{\text{notch}}=1.0$ and NO NTP

The experiments with the prototype emitter and receptor were conducted in the small isolated non-virtual network. All of the laptops were of the same model and identically configured, except for IP address and hostname, when the experiments were started. Based on validation studies performed with the new emitter’s implementation, it was determined that six hour data collections provided sufficient packet counts to perform the statistical tests.

The traffic being emitted at each of the hosts was collected with tcpdump, and the traffic received by the server was also being collected with tcpdump. The packet collections were analyzed with the process described in Section 5.1.1 and the traffic observed by the server is illustrated in Figure 5.9.

The observed traffic is illustrated in Figure 5.9, it was expected that a notch in the traffic would appear beginning at period 50. Upon further investigation it was determined that the notch was present in the traffic being emitted by the individual clients, but the server was not observing the notch in the traffic. The packets captured by tcpdump clearly showed that the individual hosts were emitting traffic which have the
Figure 5.9: Isolated Receptor observed packet counts, with $P_{\text{notch}}=1.0$ without NTP synchronization
Table 5.1: Friedman ANOVA p-Values results for the Isolated receptor with $P_{\text{notch}}=1.0$ without NTP synchronization

<table>
<thead>
<tr>
<th>Network DQS p100</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000259</td>
</tr>
<tr>
<td>Observation</td>
<td>0.077319</td>
<td>0.362903</td>
<td>0.69964</td>
<td>0.572937</td>
<td>0.216981</td>
</tr>
</tbody>
</table>

expected notch present. A sample of the traffic that a client was emitting is illustrated in Figure 5.10.

NTP was configured on each host in the network, as the network was isolated there were no external sources available to coordinate the NTP synchronization. The implemented solution was to setup the server on the network to provide NTP synchronization services to the network. All of the hosts were configured to synchronize their clocks with the server, the default NTP synchronization algorithm was used. The data collection results with the NTP configuration are detailed in the following section (Section 5.2.2).

Although the notch was not observed in the traffic, the statistical tests on the traffic observed by the server were still performed. The results of these tests are summarized in Table 5.1. The table is organized as follows: the data set that the tests were performed on are in the upper left-most cell (i.e., Network DQS p100), the hosts on the network are listed in table headers. The next two rows represents the p-values resulting from the Friedman ANOVA test. The data sources for the first row is the traffic emitted by each individual host. The server is not emitting traffic according to the quorum sensing emitter, it just responds as soon as a request is received so tests were not performed on the out going traffic. The second column contains the summary statistics for the
Figure 5.10: Isolated Host-1 Emitter observed packet counts, with $P_{\text{attack}} = 1.0$ without NTP synchronization
traffic observed by the server. The first column is the Friedman ANOVA results on all of the traffic observed by the server (i.e., the quorum sensing receptor). In order to determine the source of potentially anomalous traffic, the traffic was also segregated by host and the tests were performed on each host. Any observed different between the traffic distributions transmitted by the host and observed by the server are likely due to the infrastructure between the client and server modifying the traffic.

The Friedman ANOVA test works by comparing multiple data sets to determine if all of the data sets are likely to have been produced by the same source distribution. This test is used to determine if packet counts in the periods are statistically consistent. The individual data sets used for the tests are based on the count of packets per period. For all of the data collections in this research, 10 data collections were performed for each experimental setup. 10 collections were sufficient for Matlab to estimate p-value results from the statistical tests. Statistics were collected for each of these runs, and as a result each period in the collection consists of 10 data points; with each data point representing the count of packets during that period for each of the data collections. These periods were used for the source of the data sets which were analyzed with the Friedman ANOVA test.

The significance level chosen for the Friedman ANOVA test was chosen as 95% (alpha = 0.05). As the tests are being repeatedly performed, the Bonferroni correction (with n=99)\cite{25, 49, 48} is applied to determine if the results are statistically significant. As shown in table 5.1, the Friedman ANOVA test indicates that the traffic being emitted by each individual host has statistically significant deviations. While the traffic distributions observed by the server are all not statistically significant.
Table 5.2: Friedman ANOVA p-Values results for the Isolated receptor with $P_{\text{notch}}=1.0$

<table>
<thead>
<tr>
<th>Network DQS</th>
<th>NTP p100</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td></td>
<td>0.018361</td>
<td>0.000008</td>
<td>0.000016</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ns)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.000002</td>
<td>0.000011</td>
<td>0.000016</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td></td>
</tr>
</tbody>
</table>

5.2.2 Quorum Sensing Signal with $P_{\text{notch}}=1.0$

After the time synchronization issues were resolved in Section 5.2.1, another data collection was performed. The configuration remained the same except for the addition of the NTP service running on all of the network hosts.

The packet counts observed in this data collection are illustrated in Figure 5.11. The presence of the notch is clearly visible starting at period 50. The summary statistics are presented in Table 5.2. The first test of interest is the Friedman ANOVA test on traffic in its entirely observed by the server, and the test result reported are statistically significant. The first null-hypothesis can be rejected and follow-up testing is required. The next tests of interest are the Friedman ANOVA test results for the individual hosts; again the tests results are statistically significant for all of the host’s traffic observed by the server.

After the Friedman ANOVA test has indicated that there are statistically significant deviations present in the traffic, the periods which are statistically significant are tested with the Wilcoxon rank-sum test as described in Section 5.1.1. Figure 5.12 illustrates the results of the Wilcoxon rank-sum test. The (*) data set is the media packet counts of each of the individual data collections. The Wilcoxon rank-sum identified that periods 51-54 contain statistically significant results, but it also identified an number of other periods
Figure 5.11: Isolated Receptor observed packet counts, with $P_{\text{notch}}=1.0$
Figure 5.12: Wilcoxon rank-sum statistically significant periods for the Isolated receptor with $P_{\text{notch}}=1.0$

which contain significant results such as period 55 and periods [75-80]. Period 55 can likely be attributed to the time that is require for the external scheduler to reconfigure the network interface. The hump that appears later in the window (periods [75,80]) is likely an artifact of the scheduler.

There is a packet conservation law in effect, any packet that is scheduled to be transmitted will be emitted, it will not be dropped but it might be delayed. In order to preserve TCP sessions, packets are not dropped in order to preserve the sessions, packets can only be delayed not lost. The packets that are delayed and supposed to be emitted
during the notch will be moved to be transmitted later in the window. This hump is likely caused by the transition between the qdisc emphfifo\_fast and emphnetem delay within the Linux kernel. The Linux kernel attempts to reconfigure the network interfaces by grafting configuration items into the current queueing discipline when possible in order to allow the quickest response time. The transition from emphfifo\_fast to emphnetem delay is only adding a 20 ms delay to all of the packets within the packet queue. New settings cannot simply be grafted onto the running queueing discipline, for example a 20 ms delay cannot simply be added to allow of the packets within the queue, as while this transition process is occurring new packets may be placed into the queue, whose times should not be adjusted. In order to handle this situation, the Linux kernel destroys the current queueing discipline reapply the fifo\_fast discipline. This process requires dequeueing all of the packets from the network queue and then requeueing the packets after the fifo\_fast queueing discipline has been added.

5.2.3 Quorum Sensing Signal with $P_{\text{notch}}=0.5$

Another set of data collections was performed but the probability of a notch existing during any given window was reduced to 0.5. The packet statistics are presented in Table 5.13. It was expected and observed that the notch would not be as deep, and the hump following the notch would not be as pronounced. If on average 50% of the windows contained a notch, then it would be expected that the cumulative effect of the packet distribution would be that the notch would be approximately 50% as deep.

The Friedman ANOVA tests were again performed on this data set and the results are summarized in Table 5.3. The results of the Friedman ANOVA test were interesting in that with a statistical significance level of 95% (i.e., alpha=0.05), after applying the
Figure 5.13: Isolated Receptor observed packet counts, with $P_{\text{notch}}=0.5$
Table 5.3: Friedman ANOVA p-Values results for the Isolated receptor with $P_{\text{notch}}=0.5$

<table>
<thead>
<tr>
<th>Network DQS</th>
<th>NTP p050</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.268314</td>
<td>0.042348</td>
<td>0.012895</td>
<td>0.007039</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.311461</td>
<td>0.073375</td>
<td>0.009833</td>
<td>0.007089</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
</tbody>
</table>

Bonferroni correction ($n=99$) the only data sets which indicated a statistically significant signal was that of the traffic that the server observed. Based on this, the first null-hypothesis can be rejected and there is a statistically significant deviation in the global traffic observed by the receptor. If a network defender performed follow-up statistical tests on each host or looked that the individual traffic that the server was observing with the Friedman ANOVA test, all of the results would indicate that the test were not statistically significant meaning that the second null-hypothesis cannot be rejected.

Following the process previously outlined, the network defender and the malware would both test the traffic with the Wilcoxon rank-sum test in an effort to determine which periods contained statistically significant deviations from the median values observed. The results from the traffic that the server observed are illustrated in Figure 5.14. Periods 50 through 54 contain the notch from the packets being delayed, and the humps are observed in Periods 71, 77, and 79.

Although the results of the Friedman ANOVA test would indicate that a non-statistical test result was observed, a curious network defender or the malware itself may continue to perform the Wilcoxon rank sum tests. The periods with statistically significant deviations in the traffic observed by the Server (ix-srvr) from Host-1 (ix-host1) are illustrated in Figure 5.15. Based on the results of the Wilcoxon rank-sum
Figure 5.14: Wilcoxon rank-sum statistically significant periods for the Isolated receptor with $P_{\text{notch}}=0.5$

tests, only one of the periods within the notch is considered statistically significant, period 52.

Figure 5.16 illustrates periods with statistically significant deviations in the traffic observed by the Server (ix-srvr) from Host-2 (ix-host2). Two of the periods with the notch is considered statistically significant, periods 52 and 53. Period 64 contains a small post-notch deviation and it appears that the hump is not present in the second half of the second.

If the network defender or the malware continues to test the other hosts on the network, the periods with statistically significant deviations in the traffic observed by the Server (ix-srvr) from Host-3 (ix-host3) are illustrated in Figure 5.17. The Wilcoxon rank-sum tests results for Host-3 are interesting in that all of the periods associated with the notch, periods 50-54, are considered to be statistically significantly different from the median set (labeled *).
Figure 5.15: Wilcoxon rank-sum statistically significant periods for the Isolated receptor with $P_{\text{notch}}=0.5$ filtered for only Host 1’s traffic

Finally Figure 5.18 illustrates the periods with significant deviations in the traffic from Host-4 (ix-host4) as observed by the Server (ix-srvr). The Wilcoxon rank-sum test results indicate that periods associated with the notch at 51, 53, and 54 contain statistically significant deviations, while two of the periods associated with the post-notch hump, periods 71 and 77, are considered to be statistically significant at a significance level of 95% (alpha=0.05).
Figure 5.16: Wilcoxon rank-sum statistically significant periods for the Isolated receptor with $P_{\text{notch}}=0.5$ filtered for only Host 2’s traffic

5.2.4 Quorum Sensing Signal with $P_{\text{notch}}=0.25$

The results for the data collections with a $P_{\text{notch}}=0.25$ are similar to those of the data collections for $P_{\text{notch}}=0.5$. The packet traffic counts observed by the server are illustrated in Figure 5.19 the notch is visibly less pronounced, although it can be observed since it is expected to occur around starting at period 50 and continuing to period 54 (see Figure 5.19) statistical testing must be performed to ensure that the deviations in the traffic are statistically significant.
A summary of the results of the Friedman ANOVA test are presented in Table 5.4. Again, the level of statistical significance is set at 95% (alpha=0.05) and the Bonferroni correction (n=99) is applied. Unlike the previous data collections, the results of all the Friedman ANOVA tests return a non-significant test result. This result is significant because it means that there is a transition point between $P_{\text{notch}}=0.5$ and $P_{\text{notch}}=0.25$ when the first null-hypothesis can no longer be rejected. As the first null-hypothesis cannot be rejected, there is little reason to perform the testing for the second null-hypothesis, other than curiosity. Any results when testing for the second null-hypothesis are more likely
Figure 5.18: Wilcoxon rank-sum statistically significant periods for the Isolated receptor with \( P_{\text{notch}} = 0.5 \) filtered for only Host 4’s traffic to include type I and type II statistical errors as the result of the Friedman ANOVA test was non-significant.

If the network defender or the malware continues and performs the Wilcoxon rank-sum test on this data set, the results will be inconclusive (see Figure 5.20). Only two periods associated with the notch are considered to be statistically significant, period 50 and 54. It is interesting to note that period 50 is the beginning of the notch but most the values are higher than the median of the data set (labeled *). Period 53 is well within the notch and it appears to actually belong to the notch. Following the notch there appear to
Figure 5.19: Isolated Receptor observed packet counts, with $P_{\text{notch}}=0.25$
Table 5.4: Friedman ANOVA p-Values results for the Isolated receptor with $P_{\text{notch}}=0.25$

<table>
<thead>
<tr>
<th>Network DQS NTP p025</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.064765</td>
<td>0.243921</td>
<td>0.117264</td>
<td>0.280367</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.006042</td>
<td>0.054512</td>
<td>0.273447</td>
<td>0.113561</td>
<td>0.340094</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

Figure 5.20: Wilcoxon rank-sum statistically significant periods for the Isolated receptor with $P_{\text{notch}}=0.25$

be a large number of periods that could be associated with the hump, but there are over 9 periods which are statistically significant.

In summary the results for the $P_{\text{notch}}=0.25$ indicate that the periods associated with the notch are not statistically significant. A network defender analyzing the results would not be able to reject the null-hypothesis and determine that there is a signal present in the traffic without potentially making a type I statistical error.
Table 5.5: Friedman ANOVA p-Values results for the Isolated receptor with No quorum signal

<table>
<thead>
<tr>
<th>Network DQS NTP p025</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.548512</td>
<td>0.152218</td>
<td>0.061413</td>
<td>0.535768</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.403906</td>
<td>0.850865</td>
<td>0.701783</td>
<td>0.496315</td>
<td>0.171613</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

5.2.5 No Quorum Sensing Signal

Another experiment was performed to determine if there were anomalies in the network traffic without any of the hosts embedding the quorum sensing signal. The network was almost identically configured during the previous data collections with the exception that the quorum sensing scheduler was not operational during the run. The packet traffic counts for the observed traffic is illustrated in Figure 5.21. The most significant observation about the traffic is that the packet counts vary between periods and a number of periods contain statistical outliers.

The Friedman ANOVA tests were performed on the results of the data collections and the results are summarized in Table 5.5. The significance level for the Friedman ANOVA test was set at 95% (alpha=0.05) and the Bonferroni correction (n=99) was applied before testing for statistical significance. All of the Friedman ANOVA tests returned non-significant results. The first null-hypothesis cannot be rejected as a result of the testing on the overall packet distribution observed by the quorum sensing receptor at the server.

Following the results of the Friedman ANOVA tests, the Wilcoxon rank-sum tests were performed on the individual periods, with a significance level of 95% (alpha=0.05). The results are illustrated in Figure 5.22. As the significance level is 95% it is reasonable...
Figure 5.21: Isolated Receptor observed packet counts, with No quorum signal
Figure 5.22: Wilcoxon rank-sum statistically significant periods for the Isolated receptor with No quorum signal

to expect that about 5% of the time a test result would include a type I statistical error (i.e., false positive) or a type II statistical error (i.e. a false negative). A 100 periods were tested and 5 periods (i.e., 5% of the samples) contained results which were statistically significant from the median sample (labeled *). Periods 72-74 are in the region where the post-notch hump is typically observed, but the median values for each of those periods is below the median values for the data set (i.e. it is part of a small notch, not a post-notch hump).
5.3 Discussion

The main purpose of the prototype phase of the research was to first design and implement a new quorum sensing emitter which modified existing network traffic to embed a quorum sensing signal and then to validate the implementation by performing a number of tests in a small isolated network. This is significant change in the architecture that allows experiments to be performed in more realistic network environments and characterize the signal in the presence of noise generated by hosts and devices on the network. During the experiments a number of interesting cases were identified. These will be discussed further in Section 5.3.1. Section 5.3.2 highlights the lessons learned that needed to be carried forward in the next phase of this research.

5.3.1 Experimental Outcomes

The first significant result of this phase of research was that it is possible to induce a quorum sensing signal in a small network without the introduction of network traffic specific to the quorum sensing signal. Section 5.2.2 indicated that a quorum sensing signal could be produced and detected across a network. Unfortunately for a determined adversary, this signal was strong enough that the signal could be detected in the overall packet distribution as well as the individual packet distributions originating from each host. In this case the first null-hypothesis was rejected and the second null-hypothesis was also rejected. The determined adversary is interested in identifying cases in which the global signal can be observed and the individual contributions are within the normal statistical deviations of the network traffic. As the signal was strong enough that by monitoring the traffic from individually infected nodes, both the network defender and
the determined adversary could determine that the host was infected with little change that the observation is the result of a false-positive.

The second interesting result: There are situations in which the overall packet distribution observed by the receptor detected a statistically significant anomaly in the packet distribution but when the traffic originating from each individual host was inspected the results of the statistical tests returned non-significant results (See Section 5.2.3). This case leads to the interesting scenario in which the first hypothesis was rejected (significant deviations at the global level) but the second null-hypothesis could not be rejected (the observed deviations at the individual host level were not statistically significant).

Another case occurred when $P_{\text{notch}}$ was set to 0.25 (previously reported in Section 5.2.4). In this case neither the first or second null-hypothesises could be rejected. This indicates that some where between $P_{\text{notch}}=[0.25,0.5]$, there is a turn over point in which the signal strength of the quorum sensing emitter’s signal is weak enough that it is no longer statistically significant at the global level for the network packet distributions.

What could have potentially been interpreted as a failure of the experiment during the initial validation of the new quorum sensing emitter, the disruption of the quorums sensing signal by individual clock drift, resulted in a possible defense for network defenders. This situation has revealed a potential weakness in the communication scheme posited by this research. As described in Section 5.2.1, it was determined that the system is exceptionally sensitive to local clock skews and each clock was required to be updated throughout the data collections: once before the experiment began with the ntpdate command and throughout the data collection with the NTP synchronization service which relied on the default update schedule. The reliance on highly accurate clocks could be exploited by a network defender as a defense mechanism. A network
defender could simply remove the NTP synchronization services from their hosts and allow the clocks to drift naturally. The resulting drift would smear the signal in such a way that the determined adversary would be unable to detect the signal on the global level. It would be difficult for the determined adversary to identify the source of the disruption simply based on monitoring the traffic. If they observed all of the traffic being emitted by each individual host it would contain the expected signal.

There are times when it is required that each host on the network maintain an accurate local clock. Examples include allowing event logs to be correlated across the network or allowing tickets to be generated and validated with authentication systems such as Kerberos. In these situations, other mitigation strategies may be required. At the host level, an anti-malware solution could monitor a system and determine which processes are “obsessed” with the current system time. Indeed the quorum sensing scheduler interacts with the local system clock hundreds of times per second. In the cases referenced above, there are 100 periods a second and during each period the scheduler interacts with the local clock twice: once to determine the current time and again later to determine the time elapsed since a possible network reconfiguration. Future work could be conducted to monitor system processes to determine how frequently a process interacts with a clock. This could allow network defender to use this as a reliable indicator that a covert timing channel, such as the quorum sensing emitters, is being used on the system.

5.3.2 Lessons Learned

A few key lessons were learned during this phase of the research which need to be carried forward into the next phase of the research. The first lesson was a need for an
updated deployment process. In the case in which a small network was being setup and used throughout a long experiment, the lack of an entirely automated deployment process was not an issue. The same deployment process could not be used for any subsequent phases of the research.

The second issue learned was that the host platforms required an accurate system clock. Physical systems are generally better at maintaining a consistent and accurate clock. Virtualized systems seem to have more issues when maintaining clock accuracy, which is generally good for a network defender who is seeking to foil and determined adversaries attempting to deploy a quorum sensing-like system as a C2 channel but causes issues for the experimenter who is attempting to characterize the signal in different environments.
Chapter 6

Emulated Network Study

The Prototype discussed in Section 5 was considered to be successful in demonstrating that the digital quorum signal could be used in small isolated networking environments. The prototype phase of the study attempted to determine if the proposed C2 was viable in a more realistic network environment. In order to continue to characterize the proposed C2 channel, it is necessary to determine if the prototype is viable in a more realistic networking environment. The previous phase attempted to characterize the prototype in an environment where there was no external sources of noise as the distance between the emitters and receptors was small (network distance as well as geographic distance).

Ideally the prototype would be placed in a large virtual networking environment like Global Environment for Network Innovations (GENI) [22] which hosts a large isolated networking environment where experiments can be rapidly deployed and repeatedly conducted.

For this phase of the research a new hypothesis was used to account for the different network displacements between the emitters and receptors. The hypothesis used is that the signal generated by the quorum sensing emitter is not significantly impacted by
network noise generated by the infrastructure between the quorum sensing emitter and the quorum sensing receptor. In order to test this hypothesis, it must first be converted into a testable null-hypothesis. The following null-hypothesis was used to support statistical tests for the hypothesis, The network noise generated by different network distances between the quorum sensing receptor and the quorum sensing emitter will prevent the signal from being detected.

6.1 Design

There were two significant lessons learned that results from the Prototype phase of the research. The first lesson from the prototype, discussed in Section 5.3.2, noted that improved automated deployment mechanisms were required during the setup phase of the experiment. GENI has process for deploying and run configuration scripts. The revisions to the deployment process for the receptor and emitter are discussed in the next section.

The second lesson learned from the prototype phase of the study is the need to monitor the clock accuracy of the hosts within the experimental network. As demonstrated during the prototype phase, the detectability of the signal is sensitive to differences and accuracies of the clocks on all of the hosts. The planned deployment process for GENI was to deploy a small instance and validate the experimental setup and data collections with a $P_{\text{notch}}=1.0$ and only a single emitter and receptor.

As the experiments are sensitive to potentially small perturbations in the data the test sequence was modified. The test sequence followed in the prototype was $P_{\text{notch}}=(1.0, 0.5, 0.25, 0.0)$, while the prototype for this phase is $P_{\text{notch}}=(1.0, 0.5, 0.25, 0.1, 0.0)$ with multiple runs for each. More details are presented in Section 6.1.2.
6.1.1 Design Revisions

The prototype was successfully tested in a small isolated environment. It was determined that the emitter and receptor implementations did not need to be modified to support testing in a larger network environment. GENI offers an automated deployment mechanism which is capable of deploying virtual hosts, reconfiguring hosts to support experimental requirements and running experiments in an entirely automated pipeline.

In order to support the planned experiments in GENI, a number of modifications needed to occur for the experimental setup and deployment process. This section describes the modifications to the emitter setup and deployment process.

The experimental process was revised as all of the clients and servers operating in the GENI environment are headless servers by default (e.g., a Windowing desktop is not configured and installed). A script was developed to drive the Mozilla Firefox browser which stimulates the host into generating network traffic needed to be modified or revised to account for the absence of the desktop environment. The solution to the problem was to make use of a X.org Virtual Buffer display (xVfb). The use of xVfb allows a process which expects a desktop environment to operate in with a virtual display.

A second modification to the experimental setup was to allow for the scheduling of data collections. Scheduling was necessary in order to coordinate the starting and stopping of experimental runs in GENI. The implementation of the scheduler was simplified by the use of UNIX’s crontab. Separate jobs where scheduled to occur every eight hours, one job for each task on the client; starting the quorum sensing scheduler, starting tcpdump to collect data that the host was emitting, and finally starting the browser task. The collection period remained six hours in duration, the same
configuration as the prototype; however these jobs were only scheduled to occur every eight hours. It was determined that two hours provided sufficient time to allow for services to be reset, if necessary, along with the retrieval of a weeks worth of logs and packet captures from the client hosts. The server was configured with a single data collection cron job which started the tcpdump packet collection. The web services that the server was providing to clients was not reset as the services were only providing static HTML pages. The apache web services were not configured to provide response caching, so the a cache was not reset at the beginning of each data collection.

All hosts in the experimental network were configured with a Network Time Protocol (NTP) synchronization task which was scheduled to occur a half an hour before the start of an experiment. The time synchronization task attempts to use the ntpdate Linux program to update the system’s clock three times before restarting the NTP time services. It was experimentally determined that using the ntpdate command three times in succession allowed sufficient time for the clock to obtain an accurate value. By the conclusion of the third ntpdate command the local system clock would be within 1 millisecond of the reference time source. The NTP services were then restarted which attempted to maintain clock synchronization through out the data collection.

6.1.2 Environment Setup

A small experimental network similar to that of the prototype was created. In order to allow the results from this phase of the research to be compared with the results from the prototype phase, the number and types of hosts remained the same. There were 4 clients running the quorum sensing emitter and there was a single server monitoring the traffic acting as the quorum sensing receptor. The clients would be allocated in the same
local network, while the server would be moved to different geographic locations. As a result of the server being relocated to different geographic locations, it is subjected to different amounts of noise on the link between hosts and likely be separated by a different number of network hops. It was expected that the changes in network distance (as measured by network hops) would affect the detectability of the signal by the quorum sensing receptor.

**GENI**

During the setup of the experiment a few tests were performed in the GENI environment to validate that the experiments could be performed in that environment. The first validation experiments indicate that the environment could support this phase of the research. The validation was based on setting up two Ubuntu 14.04 x86_64 LTS servers running on a `rawpc` instance at one of the GENI sites. One server participate as the receptor while the other server assumed the role of the emitter. The system was configured with $P_{\text{notch}}=1.0$ and a six hour experiment was performed. The notch was observed and it was assumed that the remainder of the experiments could be performed successfully.

Further attempts at setting up an experiment in the GENI environment failed. Each site has a limited number of resources designed as a `rawpc`, typically only 1 or 2 per site. As such the remaining experiments were planned to be performed on an `emulab` image also running Ubuntu 14.04 x86_64 LTS operating system. Unfortunately the observed packet counts at the receptor did not contain traces of the notch. Eventually it was determined that the clock accuracy in the GENI environment was not sufficient to perform the experiment on the `emulab` images.
Table 6.1: GENI observed time difference histogram

<table>
<thead>
<tr>
<th>Delta (ms)</th>
<th>Count</th>
<th>% of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2500 or less</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>[-2500,-2250)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>[-2250,-2000)</td>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>[-2000,-1750)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>[-1750,-1500)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>[-1500,-1250)</td>
<td>64</td>
<td>1.17</td>
</tr>
<tr>
<td>[-1250,-1000)</td>
<td>63</td>
<td>1.15</td>
</tr>
<tr>
<td>[-1000,-750)</td>
<td>83</td>
<td>1.52</td>
</tr>
<tr>
<td>[-750,-500)</td>
<td>71</td>
<td>1.30</td>
</tr>
<tr>
<td>[-500,-250)</td>
<td>3</td>
<td>0.05</td>
</tr>
<tr>
<td>[-250,0)</td>
<td>4673</td>
<td>85.38</td>
</tr>
<tr>
<td>[0,250)</td>
<td>8</td>
<td>0.15</td>
</tr>
<tr>
<td>[250,500)</td>
<td>65</td>
<td>1.19</td>
</tr>
<tr>
<td>[500,750)</td>
<td>71</td>
<td>1.30</td>
</tr>
<tr>
<td>[750,1000)</td>
<td>82</td>
<td>1.50</td>
</tr>
<tr>
<td>[1000,1250)</td>
<td>65</td>
<td>1.19</td>
</tr>
<tr>
<td>[1250,1500)</td>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>[1500,1750)</td>
<td>4</td>
<td>0.07</td>
</tr>
<tr>
<td>[1750,2000)</td>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>[2000,2250)</td>
<td>1</td>
<td>0.02</td>
</tr>
<tr>
<td>[2250,2500)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>2500 or more</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>5473</td>
<td></td>
</tr>
</tbody>
</table>

The clock accuracy was measured by setting up a cron task which executes once every minute to collect the difference between the current system time and the time as measured by the local network time source. The times were collected using the Linux ntpdate command with the option to only display the difference not update the local clock. 5473 clock update differences were recorded and a histogram of the results are tabulated in Table 6.1.
Based on the absolute value of time differences recorded it was observed that approximately 15% of the clock updates occurring on the host were greater than 250 ms. As a result of the environment validation issues encountered with GENI’s clock stability on the *emulab* images and the lack of *rawpc* images in the environment, another solution needed to be devised.

In order to redesign the experiment to operate outside of the GENI environment, two significant issues needed to be addressed; one was technical and the second was related to the potential ethical concerns associated with this research. When selecting a new environment to conduct the remainder of the experiments, it is critical to have a stable clock source. As demonstrated during the Prototype 5.2.1 and the GENI validation, having an unstable clock source will prevent the receptor from identifying the quorum sensing signal.

The potential ethical concerns about performing experiments in an open environment, could have been significant enough that the research could have been terminated. Originally it was proposed that these experiments would be conducted in an isolated environment. If the experiments transition to an open computing environment, the ethical issues need to be addressed. It was determined that there were no potential risks to those interacting with either the quorum sensing emitter or quorum sensing receptor.

On important aspect of this research is to “do no harm”, transitioning to an open computing environment offers increased risk to potentially harm another entity. There are two different scenarios which needed to consider when transitioning to an open computing environment; a external entity interacting with the clients and a external entity interacting with the server. The potential risks are different and discussed separately in the following paragraphs.
When an external entity interacts with the client, they may be receiving information from the client or they may be sending information to the client. A potential risk is that the client sends unwanted traffic to a host on the network or the internet. This issue is addressed in two different ways. First it important to address where the browser is sending traffic and ensuring that the browser is configured to send traffic to hosts under the experimenter's control. When the initial configuration was established, the browser requests where manually verified and additional logic was included so that the browser validates that the URL requested from the remote server belongs to the expected address of the server. If there is a failure at this point the most significant impact that a potential host would experience is a number of unwanted web page requests. The server address entered into the browser script is validated before the data collections. It was determined that there was minimal risk of interfering with a server outside of the experimenter’s purview. The browser script only submits well-formed requests to the server. The only modification to the packets is a slight delay before transmission. The emitter is not modifying the payload that would normally be sent, and no malicious artifacts are being inserted. As such there is minimal risk of the browser interfering with the normal operation of a remote website.

Another risk is that an entity may interact with the client by sending information to it. The client is running packet capture software so there is a potential risk that the external entity may attempt to send sensitive information to the client. Although tcpdump is running on the client (e.g., the packet capture software), it is only configured to capture the first 68 bytes of the packet. Each IPv4 packet header is 20 bytes in length, the TCP protocol header is 20 bytes in length which leaves no more than 28 bytes which will be recorded by tcpdump from each packet. As most information is passed on the web via HTTP POST requests (e.g., it is contained within the body of the HTTP request), it is
almost impossible that sensitive information will be recorded in the remaining 28 bytes. Another important factor is that when the clients are attached to the network, they do not register their presence for external services to attach. An external entity would need to actively be looking for the clients in order to interact with them.

The server on the network is not seeking out other hosts on the network to send packets, it is providing a simple web hosting service. If an external entity interacts with the web server, it will only respond with web pages that the client has requested, thus it presents no more danger any other server hosting content on the internet. The web pages are not modified before they are transmitted. The server is only monitoring the traffic interacting with the network interface. Similar to the client, the server is also monitoring and recording traffic with tcpdump.

It was determined that transitioning to a more open environment to overcome the limitations of the GENI environment would result in little risk to anyone accidentally interacting with the experiment while allowing the signal to be more thoroughly characterized.

As GENI was eliminated as a potential location for conduct experiments in a larger networking environment, other likely sources needed to be identified. A few different options include: private hosting services with Virtual Private Appliances could be used, using computing resources at other universities or using computing resources from a Cloud Provider. Using computing resources from a cloud provider such as Amazon Web Services or Microsoft’s Azure Cloud Services provides a unique opportunity to move the quorum sensing receptor to different locations throughout the world without significant overhead. Both Amazon and Microsoft provide discounts for using their cloud computing resources for first time customers. Based on preliminary tests with a pair of receptors located in each company’s cloud, Amazon’s t2.micro instances
maintained better clock accuracy and it was chosen as the provider that would be used to host the quorum sensing receptors for a significant portion of the experiments for the remainder of the research.

FloridaTech Virtual Environment

In preparation for the AWS virtual environment, the ability to function in a virtual environment was tested in a local virtual environment at FloridaTech. Five virtual guests were created on a VMware ESXi 6.0 running on a Dell PowerEdge M620 server with 2 Intel Xeon e5-2630 v2 CPUs (each with 6 cores and HyperThreading) and 262 GB of RAM. Each virtual guest was configured with 2 CPU cores on a single processor socket, 1024 MB of RAM and 20 GB of Hard-drive space.

AWS Virtual Environment

After the experiment was validated in the FloridaTech virtual environment, the server (quorum sensing receptor) was moved into the AWS EC2 Cloud. This setup allowed the server to be located at different geographic and network distances relative to the emitters. During this phase the emitters would remain hosted in the FloridaTech virtual environment. Amazon allows servers to be hosted in over 30 different locations around the world. Four different locations were chosen: Oregon, Northern Virginia, Ireland and Tokyo. Oregon was chosen as a hosting site as it is one of Amazon’s largest data centers and has a significant amount of traffic moving in and out of the environment. While it was geographically closer than some of the other choices, it has a large potential for being influenced by noise in the environment. Northern Virginia was chosen as it was one of the Amazon sites closest to the emitter network at FloridaTech. As the Northern Virginia site is in the Washington DC metropolitan area, it was also in the same WLAN
that contained some of the host emitters used in the Proof of Concept phase of this research. Two more geographically distant sites were selected, one each from across the Atlantic and Pacific oceans. Ireland is one of Amazon’s largest data centers in Europe which allows for the possibility of significant network noise resulting from the amount of network traffic moving in and out of the data center. Although one of the more recent data Amazon data centers, Tokyo is one of the larger data centers in the South Eastern part of Asia.

**Test Sequence**

The sequence of data collections was similar to the order that was used when the prototype operated in the small isolated network. The probability of a notch occurring during any given window was altered. The $P_{\text{notch}}$ values used during this experiment were 1.0, 0.5, 0.25 and 0.1. When compared to the tests performed in the prototype phase, this sequence includes $P_{\text{notch}}=0.1$. This additional level was added to determine if there signal was observable at lower probability levels. As more experimental data collections were scheduled during this period, it is possible that the signal may be observed at a lower signal to noise ratio. The data collection schedule is summarized in Table 6.2.

Similar to the experimental runs for the prototype phase of the research, each experimental run consists of ten individual data collections. Each individual data collection consists of a six hour collection period in which all of the emitters are scheduling packets according to their respective $P_{\text{notch}}$ configuration. All hosts will be active during the data collection, four emitters and one receptor.

As the experiment was not able to be conducted in GENI, performing large numbers of experiments in parallel was not possible. The FloridaTech Virtual Environment was
Table 6.2: Large Network Test Sequence

<table>
<thead>
<tr>
<th>$P_{\text{notch}}$</th>
<th>Data Collections</th>
<th>Duration week(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1.00</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0.50</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0.25</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>0.10</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>2</strong></td>
<td><strong>7</strong></td>
</tr>
</tbody>
</table>

sufficient to allow two sets of data collections to occur simultaneously. The Oregon and Northern Virginia data collections were conducted in parallel and the Ireland and Tokyo data collections were conducted in parallel subsequent to Oregon and Northern Virginia.

### 6.2 Results

Based on the experience with the small isolated network and the interesting results at the lower signal strength levels, an increased number of collections and collections with a smaller $P_{\text{notch}}$ values were performed during this phase of the research.

In preparation for hosting the quorum sensing receptor in the AWS EC2 cloud, a series of validation experiments were performed with the Florida Institute of Technology’s Virtual environment within the Computer Sciences Department. The host specifications were previously summarized in Section 6.1.2.

#### 6.2.1 FloridaTech Virtual Environment

The servers hosted by virtual environment managed by the Computer Sciences Department at FloridaTech were able to maintain sufficient clock accuracy during the
course of the experiments to successfully complete the prototype validation. This section details the results collected during the course of the experiments conducted in that network.

**Quorum Sensing Signal with \( P_{\text{notch}} = 1.0 \)**

The initial validation with \( P_{\text{notch}} = 1.0 \) was only performed once when validating the ability of the emitter to operate in this virtual environment. As illustrated in Figure 6.1 the notch is present in the packet counts of the traffic observed by the quorum sensing receptor (i.e., fit-server).

The statistical process for the tests conducted during this phase of the research were unchanged from those discussed in Section 5.1.1 of the Prototype. The significance level remained unchanged at 95% (alpha=0.05) with a Bonferroni correction (n=99). This results in a critical value of 0.000505, as such the results of the statistical tests were limited to six decimal points to allow an appropriate comparison between the critical value and the observed p-values.

The results using the Friedman ANOVA test on the traffic that was observed by the server resulted in a significant result. Following the process outlined when a significant result was returned was to filter the traffic observed by the server such that the test can be repeated on traffic filtered per transmitting host. As can be seen in Table 6.3,
Figure 6.1: FloridaTech Receptor observed packet counts, with $P_{\text{notch}}=1.0$
Figure 6.2: Wilcoxon rank-sum statistically significant periods at FloridaTech with $P_{\text{notch}}=1.0$

the results of the Friedman ANOVA test on the individual hosts return statistically significant deviations for all of the hosts. This result allows for the rejection of the null-hypothesis.

After the statistically significant result from the Friedman ANOVA test, the Wilcoxon rank-sum test was performed on the traffic observed by the receptor to determine which periods contained statistically significant deviations. The significant level was set at 95% (alpha=0.05) and the individual periods were compared against the median values of each collection. The Wilcoxon rank-sum tests identified two interesting sets of periods that were statistically significant [50,53] and most of the periods with [55,80]. Periods [50,53] are consistent with the notch that is created by delaying the packets, and the periods between [55,80] are consistent with the post-notch hump.
Table 6.4: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.5$ (Data Set 1)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>DQS NTP p050 1</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.000000</td>
<td>0.000447</td>
<td>0.000000</td>
<td>0.000006</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.000004</td>
<td>0.000048</td>
<td>0.000001</td>
<td>0.000007</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.5$ (Data Set 2)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>DQS NTP p050 2</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000040</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td>(s)</td>
<td></td>
</tr>
</tbody>
</table>

**Quorum Sensing Signal with $P_{\text{notch}}=0.5$**

Another pair of data collections were performed in the FloridaTech virtual environment with $P_{\text{notch}}=0.5$. An example of one of those data collections is presented in Figure 6.3. The packet counts observed are similar to those observed when $P_{\text{notch}}=1.0$ except the notch is not as deep and the post-notch hump does not peak as high.

The Friedman ANOVA test results are summarized in Table 6.4 and Table 6.5. All of the test results return statistically significant results, so the null-hypothesis is rejected. The results of this test are similar to those encountered in the small isolated network, a statistically significant disturbance is present in the overall packet distribution of the network as well as the individual hosts.
Figure 6.3: Florida Tech Receptor observed packet counts, with $P_{\text{notch}}=0.5$ (Data Set 1)
As the overall packet distribution returned statistically significant results, follow-up testing was performed with the Wilcoxon rank-sum test to determine which periods have statistically significant deviations. The box plots for the periods with statistically significant deviations are illustrated in Figure 6.4. As can be seen in the Figure, the notch is visible (e.g., periods [51,52]) and the post-notch hump (e.g., periods [55,80]).

**Quorum Sensing Signal with P\textsubscript{\text{notch}}=0.25**

During the validation experiments conducted with the prototype in the small isolated network, the null-hypothesis could not be rejected for the overall packet distribution when P\textsubscript{\text{notch}}=0.25. As the case was borderline the null-hypothesis cannot be rejected, the number of data collections was increased from two (with regards to the 1.0 and 0.5 cases) to four. This was done to increase number of observations and attempt to determine if there are cases in which the null-hypothesis can be rejected.
Table 6.6: Friedman ANOVA p-Values results at FloridaTech with \(P_{\text{notch}}=0.25\) (Data Set 1)

<table>
<thead>
<tr>
<th>FIT Network DQS NTP p025 1</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.001579</td>
<td>0.066484</td>
<td>0.015171</td>
<td>0.073602</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.006645</td>
<td>0.003537</td>
<td>0.153900</td>
<td>0.036751</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

Table 6.7: Friedman ANOVA p-Values results at FloridaTech with \(P_{\text{notch}}=0.25\) (Data Set 2)

<table>
<thead>
<tr>
<th>FIT Network DQS NTP p025 2</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.506717</td>
<td>0.106339</td>
<td>0.011757</td>
<td>0.010653</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.290591</td>
<td>0.023643</td>
<td>0.000017</td>
<td>0.057793</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

Figure 6.5 illustrates the overall packet counts observed by the receptor during one of the data collections when \(P_{\text{notch}}\) was set to 0.25. Other packet distributions are similar so all 4 box plots are not presented.

Tables 6.6, 6.7, 6.8 and 6.9 summarize the p-value results of the Friedman ANOVA tests for the data collected during this phase of the research.

In general, the results are all similar, all of the results the packet distributions observed by the receptor contain statistically significant deviations. When the same statistical tests are performed on the traffic transmitted by the individual emitters or received by the receptor but limited to a single host, the tests are all non-significant except for one case; data collection FIT_Network_DQS_NTP_p025_2 (presented in Table 6.7) when traffic from fit-host3 is observed by the receptor. It is even more interesting when the test results for the traffic observed by fit-host3’s NIC returns a non-significant
Figure 6.5: FloridaTech Receptor observed packet counts, with $P_{\text{notch}}=0.25$ (Data Set 1)
Table 6.8: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.25$ (Data Set 3)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>DQS NTP p025 3</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.051381</td>
<td>0.009751</td>
<td>0.000717</td>
<td>0.002317</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.004487</td>
<td>0.12331</td>
<td>0.055301</td>
<td>0.016720</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.9: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.25$ (Data Set 4)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>DQS NTP p025 4</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.006200</td>
<td>0.240124</td>
<td>0.007371</td>
<td>0.002176</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>0.000000</td>
<td>0.000594</td>
<td>0.357684</td>
<td>0.006040</td>
<td>0.014507</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
</tbody>
</table>

test result for the Friedman ANOVA test. In this case, the null-hypothesis is rejected based on the traffic observed by the receptor, but the null-hypothesis cannot be rejected based on the traffic observed the emitter. An observer would most likely conclude that this is a type I error and the statistically significant deviations in traffic are caused by the infrastructure between the server and the receptor.

When the Friedman ANOVA test results indicate that the null-hypothesis should be rejected, follow-up testing with Wilcoxon rank-sum tests was performed on the traffic to identify which periods contain statistically significant deviations. Figure 6.6 illustrates the box plots for Data collection 1. The effect of the notch is observed during periods [51,52] and the post-notch hump is also observed. There also seems to be a noticeable deviation in the statistics for a large number of other periods.
Figure 6.6: Wilcoxon rank-sum statistically significant periods at FloridaTech with \( P_{\text{notch}} = 0.25 \) (Data Set 1)

**Quorum Sensing Signal with \( P_{\text{notch}} = 0.1 \)**

Based on the lessons learned from the Prototype, it was determined that running more experiments with smaller values of \( P_{\text{notch}} \) might assist characterizing the signal at lower signal to noise ratios. This phase of the study introduced running the external scheduler with \( P_{\text{notch}} = 0.1 \). As it was expect that this \( P_{\text{notch}} \) value would result in a signal that was more difficult to consistently identify, four data collection with \( P_{\text{notch}} = 0.1 \) were collected. Figure 6.7 illustrates a sample of the traffic that was observed by the server hosting the quorum sensing receptor.

Tables 6.10, 6.11, 6.12 and 6.13 summarize the results of the Friedman ANOVA test on the four data collections. The results are interesting in that only two of the data sets produce results which are statistically significant (alpha=0.5) after applying the Bonferroni correction (n=99): Data Set 1 (FIT_Network_DQS_NTP_p010_1) and Data Set 4 (FIT_Network_DQS_NTP_p010_4). In other words, with \( P_{\text{notch}} = 0.1 \) half of the time
Figure 6.7: FloridaTech Receptor observed packet counts, with $P_{\text{notch}}=0.10$ (Data Set 1)
Table 6.10: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$ (Data Set 1)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQS NTP p010 1</td>
<td>-</td>
<td>0.483262</td>
<td>0.198643</td>
<td>0.368580</td>
<td>0.049162</td>
</tr>
<tr>
<td>Emission</td>
<td>-</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.000058</td>
<td>0.041931</td>
<td>0.147933</td>
<td>0.359399</td>
<td>0.106304</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

Table 6.11: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$ (Data Set 2)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>DQS NTP p010 2</td>
<td>-</td>
<td>0.018464</td>
<td>0.254140</td>
<td>0.200952</td>
<td>0.672285</td>
</tr>
<tr>
<td>Emission</td>
<td>-</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.028017</td>
<td>0.290591</td>
<td>0.805855</td>
<td>0.364366</td>
<td>0.828903</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

The Friedman ANOVA test results indicate that there is nothing statistically significant in the traffic on the network but the other half of the time the test would indicate that there are statistically significant differences in the traffic.

Figure 6.8 summarizes the results of the Wilcoxon rank-sum tests (alpha=0.05) on one of the data sets which produced a statistically significant result from the Friedman ANOVA test (i.e., Data Set 1). As can be seen from the box plots in the figure, Periods 51 and 52 are below the median set (indicated by the period labeled with a *). These findings are expected to be different as a result of the notch. Another interesting observation is that following the notch, there is the post-notch hump which results in a large number of periods having statistically significant deviations which are greater than the median as a result of the packet conservation law.
Table 6.12: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$ (Data Set 3)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>DQS NTP p010</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>- 0.262896</td>
<td>0.473278</td>
<td>0.584052</td>
<td>0.727315</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.13: Friedman ANOVA p-Values results at FloridaTech with $P_{\text{notch}}=0.10$ (Data Set 4)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>DQS NTP p010</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>- 0.626624</td>
<td>0.923946</td>
<td>0.110697</td>
<td>0.102950</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Observation</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.8 should be contrasted with Figure 6.9. Figure 6.9 represents the Data Set 2 which the Friedman ANOVA test returned a non-significant test result, so conclusions about the data set must be carefully drawn. A few observations about the data set can be made. The first important observation that can be made is that none of the periods associated with the notch were classified as statistically significant with the Wilcoxon rank-sum test ($\alpha=0.05$). The second observation is that although the notch does not appear to be significant, there are several periods following the notch which were classified as being statistically significant.

Although it appears that the post-notch hump could be used as a potential signal and identify this C2 channel operating in a defenders network, this observation cannot be supported as being statistically significant as the Friedman ANOVA test returned a non-significant result. Secondly, as will be discussed in the next section, there are cases
Figure 6.8: Wilcoxon rank-sum statistically significant periods at FloridaTech with $P_{\text{notch}}=0.10$ (Data Set 1)

in which there is no embedded quorum sensing signal and that observation would result a misleading distraction.

**No Quorum Sensing Signal**

Another pair of data collections were performed, but this time with an inactive scheduler. The scheduler was installed on the hosts but not enabled during the data collections. This set of data collection was performed to characterize the network traffic without embedding the quorum sensing signal and to investigate the performance of the statistical tests on this network traffic. Figure 6.10 illustrates the traffic observed by the receptor.

The results of the Friedman ANOVA tests are presented in Tables 6.14 and 6.15, with a 95% significance level ($\alpha=0.05$) and the Bonferroni correction ($n=99$). As can be observed in the test results, the null-hypothesis can be rejected which leads to the
conclusion that there is no statistically significant differences in the traffic observed by the quorum sensing receptor when there is no quorum sensing signal.

As the Friedman ANOVA test returned non-significant test results, there is no reason that the Wilcoxon rank-sum tests should be performed (other than intellectual curiosity). Figure 6.11 illustrates the results of the Wilcoxon rank-sum test with a 95% significance level (alpha=0.05). Although no embedded quorum sensing signal is present, there are still periods which contain statistically significant deviations. If a network defender performed this test and accepted the test results without accounting for the Friedman ANOVA test results, type would be making a type I error.

6.2.2 Oregon AWS Virtual Environment

After the experimental setup was validated by using the small emitter network located at FloridaTech, the experiments were repeated by this time the receptor was located
Figure 6.10: Florida Tech Receptor observed packet counts, with No quorum signal (Data Set 1)
Table 6.14: Friedman ANOVA p-Values results at FloridaTech with No quorum signal (Data Set 1)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>NODQS NTP 1 server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.018361</td>
<td>0.588101</td>
<td>0.460612</td>
<td>0.559466</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.035056</td>
<td>0.275803</td>
<td>0.324715</td>
<td>0.463699</td>
<td>0.567954</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

Table 6.15: Friedman ANOVA p-Values results at FloridaTech with No quorum signal (Data Set 2)

<table>
<thead>
<tr>
<th>FIT Network</th>
<th>NODQS NTP 2 server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emission</td>
<td>-</td>
<td>0.798327</td>
<td>0.078242</td>
<td>0.721226</td>
<td>0.494992</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>Observation</td>
<td>0.053035</td>
<td>0.295088</td>
<td>0.430166</td>
<td>0.984130</td>
<td>0.077775</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

in an Amazon data center. When setting up a virtual server in the AWS EC2 cloud services, the Oregon data center is the default option for placing the server. In order to characterize the effects of the quorum signal by separating the emitter from the receptors, a set of distance measurements needed to be made. The Linux traceroute command was used to determine the number of network hops between the receptors and the emitters as well as the round trip time delay.

A traceroute was performed and the number of hops returned was 15, meaning that there were at least 15 network devices between the emitters located at FloridaTech and the receptor located in the AWS cloud in the Oregon data center. The round-trip delay between the receptor and the emitters was approximately 83 ms. This result means that the packets will take slightly longer than 40 ms to arrive at the receptor after being transmitted by the emitter. As the window/period configuration of the notch remained
the same, it was expected that the notch would occur about four periods later in the packet distribution.

A small number of pre-experiment validations were conducted to determine if the clocks in the AWS data center consistent enough to ensure that the experiment would provide results. Amazon AWS EC2 provides a number of different configuration options that can be used when deploying a virtual server in Amazon’s Cloud. Two different instances were tested; t2.micro and t2.medium. It was experientially determined that the t2.micro configuration maintained better clock accuracy during the course of a data collection, so all virtual machine instances used for the remainder of this research relied on the t2.micro instance. The AWS EC2 t2.micro instances are considered General Purpose computing instances and each is allocated 1 Virtual CPU and 1 GB of RAM. The instance is virtual instance configured with a virtual CPU. This configuration is important as it means that the virtual machines are on a time sharing system and the

Figure 6.11: Wilcoxon rank-sum statistically significant periods at FloridaTech with No quorum signal (Data Set 1)
configuration is such that all resources in the virtual machine are running on shared resources including the CPU. The effects of time sharing the CPU can be seen in the packet distributions observed by the receptor when it resides in the AWS Oregon datacenter. As with the virtual environment with FloridaTech, all of the virtual machines created within the Amazon EC2 Cloud were loaded with the Ubuntu 14.04 LTS x86_64 operating system.

**Quorum Sensing Signal with P\textsubscript{notch}=1.0**

The experiments with the Oregon data center began with P\textsubscript{notch}=1.0, meaning that all of the packets would be delayed by 20 ms if they were transmitted during periods 51-54. Figure 6.24 illustrates the packet distribution observed by the receptor. As shown in the packet distributions, the receptor observed multiple notches. There are notches in the periods 55-58, 65-68, 76-78, and possibly another notch at 85-87. Multiple notches were not observed in any of the previous experiments, so another effect factor must be causing the appearance of the multiple notches. The difference between the notches is fairly consistent, approximately ten periods or 100 ms. As referenced in Section 6.2.2, the virtual servers are using vCPUs which are time shared. As the virtual machine running the receptor in the AWS EC2 cloud is not running on dedicated hardware it would be reasonable to assume this is caused by the virtual machine sharing the CPU with virtual machines on the same non-virtual server.

The Friedman ANOVA test results are similar to those during the validation in the FloridaTech network previously discussed in Section 6.2.1. The p-value for the traffic observed by the receptor is 0.000000 with a significance level of 95% (alpha=0.05) and the Bonferroni correction is applied (n=99). If the traffic is filtered by host and the Friedman ANOVA tests are repeated with the same significance level, the results are the
Figure 6.12: AWS Oregon Receptor observed packet counts, with $P_{\text{notch}}=1.0$ (Data Set 1)
same: p-values of 0.000000. The results of the Friedman ANOVA test indicate that the null-hypothesis should be rejected. There are statistically significant differences in the observed network traffic for the global packet distribution and the observed distributions when the traffic is limited to individual hosts.

The results of the Wilcoxon rank-sum tests (alpha=0.05) for the traffic observed at the server for the global distribution are illustrated in Figure 6.13. Unfortunately the results of this test do not provide too much useful information, other than a large number of the periods exhibit statistically significant deviations from the median values for the data set. A few interesting details can be observed. The periods belonging to the first three notches are shown to contain statistically significant deviations. The fourth potential notch however is not seen as statistically significant but the periods on either end of the notch are seen to be statistically significant. This occurrence is likely a result of the post-notch hump elevating the packet counts during the second half of the second. The notch is smaller so the notch plus the post-notch elevation has the effect of canceling out and returning the packet counts to values closer to the median.

**Quorum Sensing Signal with \( P_{\text{notch}} = 0.5 \)**

The experiment was repeated, but with the \( P_{\text{notch}} \) adjusted to a value of 0.5. As illustrated in Figure 6.14, the multiple notches in the packet distributions previously observed by the receptor are still present. The Friedman ANOVA tests on this data set returned similar results. The p-values resulting from the Friedman ANOVA tests were generally non-zero but still not significant enough to return a non-significant result for any of the tests. These results indicate that the null-hypothesis can be rejected. It is appropriate to follow-up with a set of Wilcoxon rank-sum tests to determine which periods contain statistically significant deviations.
Figure 6.13: Wilcoxon rank-sum statistically significant periods at AWS Oregon with $P_{notch}=1.0$ (Data Set 1)
Figure 6.14: AWS Oregon Receptor observed packet counts, with $P_{\text{notch}}=0.5$ (Data Set 1)
The multiple notches in the packet distribution appear to be present in these test results, but when the Wilcoxon rank-sum tests (alpha=0.05) are used on this data set only the first two notches appear to be statistically significant. As seen in Figure 6.15, the notches at periods 55-58 and periods 65-68 both are statistically significant. In general the figure contains fewer periods which have results which indicate that they contains deviations from the median set which are statistically significant.

**Quorum Sensing Signal with P\textsubscript{notch}=0.25**

As described in Section 6.1.2, four data collections were performed when P\textsubscript{notch}=0.25. As previously stated this is the setting where the results of the Friedman ANOVA test start to transition from significant to non-significant for the traffic originating from the individual emitters. The traffic distributions for data set 1 observed by the receptor are
Table 6.16: Friedman ANOVA p-Values results at AWS Oregon with $P_{notch}=0.25$

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server</th>
<th>host 5</th>
<th>host 6</th>
<th>host 7</th>
<th>host 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>0.039957</td>
<td>0.000898</td>
<td>0.052388</td>
<td>0.001150</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>2 *</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
</tr>
<tr>
<td>3</td>
<td>0.000000</td>
<td>0.000113</td>
<td>0.208092</td>
<td>0.031856</td>
<td>0.005921</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>4</td>
<td>0.000000</td>
<td>0.003351</td>
<td>0.011229</td>
<td>0.119456</td>
<td>0.000612</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

shown in Figure 6.16. Again, the first, second, and third notches and can be observed in the expected locations.

The transition to Daylight Saving Time occurred on March 13th, 2016 during the middle of the second AWS Northern Virginia data collection. The scheduler was not able to properly handle the transition from Eastern Standard Time (EST) to Eastern Daylight Savings Time (EDT). The emitter was reconfigured after the change and continued to function properly for the remainder of the experiments. The values presented in this data set are not correct and as the results are not included in the analysis.

A summary of the Friedman ANOVA test results are presented in Table 6.16, with a 95% significance level ($\alpha=0.05$) and the Bonferroni correction ($n=99$) applied. When traffic observed by the receptor for all of the data sets is tested with the Friedman ANOVA test, all of the results are statistically significant and the null-hypothesis can be rejected. As expected, when the Friedman ANOVA tests are repeated but only on the distributions from traffic originating from specific hosts, the results are generally non-significant. Data set three contains statistically significant deviations in the packet distribution observed by the receptor. While data set 1 and 4 contain results which are more inline with the signal that the determined adversary is able to leverage. The overall
Figure 6.16: AWS Oregon Receptor observed packet counts, with $P_{notch} = 0.25$ (Data Set 1)
packet distribution contains a statistically significant result, but none of the individual hosts contain statistically significant results.

Figure 6.17 from data set 1, illustrates the resultant box plots from the Wilcoxon rank-sum tests (alpha=0.05). In this case, the first two notches (periods [56-58] and [65-68]) are shown to contain statistically significant deviations from the set’s median values. It is interesting to note that unlike the previous experiments, almost none of the periods associated with post-notch hump as seen as having distributions which are statistically significantly different from the median values of the set.

**Quorum Sensing Signal with $P_{\text{notch}}=0.10$**

When the same experiments were repeated with $P_{\text{notch}}=0.1$ using the receptor hosted in the AWS EC2 data center in Oregon, a few different observations can be made. The packet distributions observed by the receptor located in Oregon are illustrated in Figure
Table 6.17: Friedman ANOVA p-Values results at AWS Oregon with $P_{\text{notch}}=0.10$

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server</th>
<th>host 5</th>
<th>host 6</th>
<th>host 7</th>
<th>host 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.065608</td>
<td>0.061100</td>
<td>0.048539</td>
<td>0.887707</td>
<td>0.653911</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>2</td>
<td>0.000873</td>
<td>0.009577</td>
<td>0.041682</td>
<td>0.126831</td>
<td>0.046324</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>3</td>
<td>0.000226</td>
<td>0.008696</td>
<td>0.008129</td>
<td>0.658934</td>
<td>0.000020</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
</tr>
<tr>
<td>4</td>
<td>0.057483</td>
<td>0.203219</td>
<td>0.696128</td>
<td>0.336518</td>
<td>0.022554</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

6.18. The traffic appears to contain a high amount of noise and the notches are not easily observed.

The results of the Friedman ANOVA tests are summarized in Table 6.17. The Friedman ANOVA tests used a 95% significance level ($\alpha=0.05$) with the Bonferroni correction ($n=99$). In the case of data set 1 and 4, the Friedman ANOVA tests indicated that the results are non-significant. The null-hypothesis cannot be rejected in this case, so there is no statistically significant deviations in these data sets. The Friedman ANOVA test results for data set 2 are more inline with what was previously observed with smaller $P_{\text{notch}}$ values, the global distribution contained statistically significant deviations but when the same test was applied to traffic originating from specific hosts the results were non-significant.

Data set 3 from Table 6.17 and illustrated in Figure 6.19 contains a particularly interesting result which required further investigation. The alignment of the packets being requested by host 3 resulted in a significant result from the Friedman ANOVA test. The null-hypothesis was rejected meaning that there is a statistically significant deviation in the packets originating from host 3. The packet traces were reviewed in an
Figure 6.18: AWS Oregon Receptor observed packet counts, with $P_{\text{notch}}=0.10$ (Data Set 1)
attempt to determine why such a strong signal was observed from host 3, but no clear sources of the uncharacteristically strong signal were observed.

As can be seen in Figure 6.19, the first three notches can be observed at the predicted locations. When the Wilcoxon rank-sum test (alpha=0.5) is performed on data set 3, three periods associated of the three notches are observed to have statistically significant deviations from the median values of the data set. The box plots associated with these tests are illustrated in Figure 6.20.

No Quorum Sensing Signal

Two data collections were performed in the AWS Oregon data center and the packet distributions were analyzed with the Friedman ANOVA tests. As expected, the p-values associated with the Friedman ANOVA tests (0.138210 and 0.285578) were non-significant when the significant level was set at 95% (alpha-0.05) and the Bonferroni correction was applied.

6.2.3 Northern Virginia AWS Virtual Environment

A virtual machine located in Northern Virginia AWS EC2 data center was tested in parallel with the Oregon data center. Two sets of emitters operating the parallel to allow receptors in both the Oregon and Northern Virginia data centers. The results of the individual experiments are presented in this section.

Before tests were performed traceroute was used to characterize the network between FloridaTech and the Amazon’s data center in Northern Virginia. The round trip time was on the order of 11 ms with 13 network hops between the emitter and the receptor. This configuration should result in less than 6 ms delay between the packets
Figure 6.19: AWS Oregon Receptor observed packet counts, with $P_{\text{notch}}=0.10$ (Data Set 3)
transmission at FloridaTech and its reception in the AWS data center. A delay on the order of 5 ms is similar less than 1 period, which means we would expect the observed notch period to be similar to those observed in the FloridaTech validation experiments (previously described in Section 6.2.1).

**Quorum Sensing Signal with $P_{\text{notch}}=1.0$ and 0.5**

A set of data collections were performed in the Amazon Northern Virginia data center, the results were similar to those observed when the experimental network was validated
in the FloridaTech virtual environment. Unlike the results observed in Amazon’s Oregon data center, the observed packet distribution in the Northern Virginia data center did not appear to have multiple notches. The observed packet distribution is illustrated in Figure 6.21.

A set of Friedman ANOVA tests were performed on the two pairs of data collections: two data sets with $P_{\text{notch}}=1.0$ and two data sets with $P_{\text{notch}}=0.5$. The significance level for the Friedman ANOVA test was set at 95% (alpha=0.05) and the Bonferroni correction was applied (n=99). The Friedman ANOVA tests for all 4 of the global distributions tested return statistically significant results (p-values were 0.000000 for all four data sets). The null-hypothesis is rejected for these data sets meaning that there are statistically significant differences in the period distributions. Additional Friedman ANOVA tests were performed on the traffic filtered by source and the results were similar between the data sets. For the $P_{\text{notch}}=1.0$ data sets the p-values were all 0.000000, while results for the $P_{\text{notch}}=0.5$ data sets had a maximum p-value of 0.000205. Follow-up testing was performed in accordance with the process previously described with the Wilcoxon rank-sum tests and a significance level of 95% (alpha=0.05) was performed on these data sets. In all four cases, the Wilcoxon rank-sum identified the notch in the periods 51-54.

The receptor in Amazon’s Norther Virginia data center was configured with the same options as the Oregon data center (e.g., a t2.micro instance with the Ubuntu 14.04 LTS Server x86_64 operating system). The packet distributions do not contain the multiple humps. This difference may be due to the lack of co-located tenants attempting to utilize the same resources. The Northern Virginia data center is newer than the Oregon data center and is not the default option when establishing new virtual machines, there may be less competition for the same resources or there may be significantly more resources
Figure 6.21: AWS Northern Virginia Receptor observed packet counts, with $P_{\text{notch}}=1.0$ (Data Set 1)
available in this data center. It is likely the second case as the Norther Virginia data center is one of Amazon’s older data centers. As the appearance of multiple notches was observed at the Oregon data center and not by the receptor located in the Northern Virginia data center and these two sets of experiments were conducted in parallel with all of the emitters residing the same physical host at FloridaTech, we can conclude that the notches are an artifact of the Oregon data center’s configuration and not induced by the border of the FloridaTech’s network and the Internet.

**Quorum Sensing Signal with $P_{notch}=0.25$**

Figure 6.22 illustrates the packet distributions observed by the receptor when $P_{notch}=0.25$ located in the AWS Northern Virginia data center. As expected notch is visible in this data set, but it is not at deep when compared to the notches observed when $P_{notch}=1.0$ or $P_{notch}=0.5$.

Table 6.18 lists the p-values that resulted from the series the Friedman ANOVA tests of the data collected during this part of the research. The significance level for these tests remained at 95% (alpha=0.05) with the Bonferroni correction (n=99). In general these results are inline with that has been previously observed when $P_{notch}=0.25$, the p-values are significance for the observed global packet distributions, but not significant for the observed packet distributions of the individual clients (except for host 4 in data set 1).

The transition to Daylight Saving Time occurred on March 13th, 2016 during the middle of the second AWS Northern Virginia data collection. The scheduler was not able to properly handle the transition from Eastern Standard Time (EST) to Eastern Daylight Savings Time (EDT). The emitter was reconfigured after the change and continued to function properly for the remainder of the experiments. The values presented in this data
Figure 6.22: AWS Northern Virginia Receptor observed packet counts, with $P_{\text{notch}}=0.25$ (Data Set 1)
Table 6.18: Friedman ANOVA p-Values results at AWS Northern Virginia with $P_{\text{notch}}=0.25$

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>0.001708</td>
<td>0.086331</td>
<td>0.003083</td>
<td>0.000268</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
</tr>
<tr>
<td>2</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
<td>- (-)</td>
</tr>
<tr>
<td>3</td>
<td>0.000042</td>
<td>0.040365</td>
<td>0.342822</td>
<td>0.106769</td>
<td>0.048096</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>4</td>
<td>0.000000</td>
<td>0.010667</td>
<td>0.000752</td>
<td>0.005605</td>
<td>0.098578</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

set are not correct and as such they are not presented in the table, nor were the results included in the characterization of the quorum sensing signal.

In this case, the null-hypothesis is rejected and the observers on the network can conclude that there is a statistically significance difference in the packet distributions observed by the server. If they attempt to narrow down the which hosts are potentially causing the disturbance by repeating the Friedman ANOVA tests on the traffic originating from the individual hosts, the null-hypothesis can only be rejected in one case: by host 1 in data set 1. If the defender did not perform a data collection during that period or the data was more similar to the other 3 data sets, the network defender would not be able to conclude that any of the hosts are causing the disturbance. Following the Friedman ANOVA test, the network defender should follow-up with a Wilcoxon rank-sum test to determine which periods are statistically significant (using alpha=0.05), the results of the Wilcoxon rank-sum test are illustrated in Figure 6.23. The notch at periods 52 and 53 is identified but there are also a large number of other anomalous periods between period 56-95.
When the emitters are configured to generate traffic with a lower signal to noise ratio, $P_{\text{notch}}=0.1$, the signal is less pronounced in the traffic observed by the receptor. An example of the traffic observed by the receptor located in the Northern Virginia data center is illustrated in Figure 6.24. If the receptor is aware of the distance between the receptor and the emitter and is expecting to observe the notch to be located at approximately period 52, then a notch candidate can be observed. The significance of the candidate notch will be reviewed during this section.

A summary of the Friedman ANOVA tests results for the tests performed on the four data collections with $P_{\text{notch}}=0.1$ with the receptor located in the AWS Northern Virginia data center is presented in Table 6.19. The Friedman ANOVA test results indicate that the null-hypothesis can be rejected for data sets 1 and 3, but not for 2 and 4. In cases 2 through 4, if the follow-up filtering is performed and the Friedman ANOVA tests are
Figure 6.24: AWS Northern Virginia Receptor observed packet counts, with $P_{notch}=0.1$ (Data Set 1)
Table 6.19: Friedman ANOVA p-Values results at AWS Northern Virginia with $P_{\text{notch}}=0.10$

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000023</td>
<td>0.079806</td>
<td>0.002510</td>
<td>0.231231</td>
<td>0.000005</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.003733</td>
<td>0.339795</td>
<td>0.068258</td>
<td>0.371766</td>
<td>0.206947</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.000140</td>
<td>0.362528</td>
<td>0.336809</td>
<td>0.034477</td>
<td>0.075822</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.013607</td>
<td>0.609712</td>
<td>0.017076</td>
<td>0.777983</td>
<td>0.160394</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td></td>
</tr>
</tbody>
</table>

again applied all of the individual hosts will have non-significant results meaning that the null-hypothesis cannot be rejected. These represent another set of cases in which the determined adversary has the advantage. The signal can be observed in the global signal by all of the observers but a potentially infected individual host cannot conclusively be identified by the network defender.

As an example of the follow-up Wilcoxon rank-sum process, data set 1 is further analyzed with the Wilcoxon rank-sum (alpha=0.05). The resultant box plots of the periods which were statistically significantly different from the media values of the data sets are presented in Figure 6.25. It is interesting to note that in the global traffic observed by the receptor, visually it appears that there would be a significant period at approximately period 52, characterized by the tighter box in the figure. When this set is compared against the median values for the data set, this is not the period associated with the notch that is highlighted as containing a statistically significant deviation. Period 55 has a significant result, meaning that the null-hypothesis is rejected.
Figure 6.25: Wilcoxon rank-sum statistically significant periods at AWS Northern Virginia with $P_{\text{notch}}=0.10$ (Data Set 1)

**No Quorum Sensing Signal**

A series of data collections were performed with the receptor located in the AWS Northern Virginia data center. The Friedman ANOVA test was performed on the overall global packet distributions and the results for the two collections were a p-value=$0.850193$ and p-value=$0.043204$. With a significance level of 95% (alpha=$0.05$) and the Bonferroni correction (n=99), both of these tests return non-significant results. The null-hypothesis cannot be rejected meaning that there are no statistically significant deviations in the packet distributions. This result is inline with the expectations as the quorum sensing signal is not being embedded in the traffic being transmitted by the emitters.
6.2.4 Ireland AWS Virtual Environment

Another large data center maintained by Amazon is the AWS data center in Dublin, Ireland. This research attempted to characterize the signal in a variety of different network environments and in order to expand the geographic distance between the emitters and receptor in Ireland was chosen as another hosting location for the receptor. The Ireland data center was expected to have a significant amount of traffic flowing into and out of the data center as it is offers one of the most complete service offerings in Europe and is the oldest Amazon data center in Europe (established in 2007).

Before experiments were conducted in the Ireland data center, the route between the FloridaTech virtual environment and the Amazon data center was characterized. With the Linux traceroute utility it was determined that the number of network hops between the receptor and the emitter was 19, and the round trip delay between packet request transmission and packet response reception was approximately 115 ms (or 11.5 periods). Based on the observed times and assuming that the one way trip time is half of the round trip time, it was calculated that the time between transmission by the emitter and reception at the receptor is approximately 67 ms or 6.7 periods.

**Quorum Sensing Signal with $P_{\text{notch}}=1.0$ and $P_{\text{notch}}=0.5$**

As set of data collections was performed with the $P_{\text{notch}}=1.0$ and $P_{\text{notch}}=0.5$ with the receptor located in the AWS Ireland data center. The p-value results of the Friedman ANOVA tests are more similar to the results collected from the AWS Oregon data center than the AWS Northern Virginia data center, but all of the results follow the same general pattern when the emitters are inducing a strong quorum sensing signal. An example of the global traffic observed by the quorum sensing receptor located in the Ireland
data center is illustrated in Figure 6.26. As can be seen by the figured, there appear to be multiple sets of notches observed by the receptor. The periods associated with the notches are as follows: the first notch occupies periods [55-59], the second notch occupies periods [66-70], the third notch occupies period [79-82] and the forth notch occupies periods [89-92]. When the differences between the middles of the notches is analyzed, it appears that the mean value is 11.3 periods (113 ms) and the median is 11.2 periods (112 ms). It appears that the spacing between the notches is somewhat delayed over what was observed in the Oregon data center but the delay is consistent.

The Friedman ANOVA tests were repeated on the global traffic observed by the receptor in the Ireland data center. Two pairs of data collections were performed: two collections with $P_{notch}=1.0$ and two collections with $P_{notch}=0.5$. In both cases the Friedman ANOVA test returned a p-value=0.000000 when $P_{notch}=1.0$, meaning that the null-hypothesis was rejected. Likewise when both data sets with the $P_{notch}=0.5$ and p-values return by the Friedman ANOVA test were also 0.000000 and also results in the null-hypothesis being rejected. This result means that in both pairs of data sets the data sets contain statistically significant deviations in the packet counts in the period distributions. The Friedman ANOVA tests can be repeated on traffic which is filtered by the individual emitter’s hosts and all of the resultant p-values except one will return significant results. Data set 1, host 1 when $P_{notch}=0.5$ will return a borderline non-significant Friedman ANOVA result with a p-value=0.001983. For all of the other cases the significant result means that there are statistically significant deviations in the periods, and follow-up testing with the Wilcoxon rank-sum tests are required to determine which periods are anomalous.
Figure 6.26: AWS Ireland Receptor observed packet counts, with $p_{notch}=1.0$ (Data Set 1)
Table 6.20: Friedman ANOVA p-Values results at AWS Ireland with $P_{\text{notch}}=0.25$

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server (s)</th>
<th>host 1 (ns)</th>
<th>host 2 (ns)</th>
<th>host 3 (s)</th>
<th>host 4 (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>0.001905</td>
<td>0.002973</td>
<td>0.000133</td>
<td>0.085248</td>
</tr>
<tr>
<td>2</td>
<td>0.000000</td>
<td>0.000352</td>
<td>0.000257</td>
<td>0.065033</td>
<td>0.000982</td>
</tr>
<tr>
<td>3</td>
<td>0.000000</td>
<td>0.001489</td>
<td>0.002523</td>
<td>0.000000</td>
<td>0.048096</td>
</tr>
<tr>
<td>4</td>
<td>0.000000</td>
<td>0.000934</td>
<td>0.000000</td>
<td>0.000157</td>
<td>0.000008</td>
</tr>
</tbody>
</table>

**Quorum Sensing Signal with $P_{\text{notch}}=0.25$**

Another set of four data collections were performed with $P_{\text{notch}}=0.25$ with the receptor located in the AWS Ireland data center. Figure 6.27 illustrates the overall packet distribution observed by the receptor, for the first of the four data collections. The presence of the multiple notches can still be observed in the traffic the quantity of notches has been reduced as well as their depth.

A summary of the Friedman ANOVA test results is provided in Table 6.20. As can be seen by the resultant p-Values, the global traffic distributions observed by the receptor display statistically significant deviations in the period packet counts. With these p-values, the null-hypothesis can be rejected the deviations in the packet counts are statistically significant. Unlike previous experiments, in this case there frequently a number of Friedman ANOVA tests specific to individual hosts which also return statistically significant results. The minimum number of hosts displaying the anomalous traffic is 1, while the maximum is 3.

Following the process previously outline, in all four of these cases a follow-up analysis can be performed with the Wilcoxon rank-sum (alpha=0.05). When the Wilcoxon rank-sum tests are performed on the global packet counts from the first
Figure 6.27: AWS Ireland Receptor observed packet counts, with $P_{notch}=0.25$ (Data Set 1)
data set, it can be seen that two of the notches are clearly visible in the periods with statistically significant deviations in periods 57-78 and 68-70 (see Figure 6.28).

Quorum Sensing Signal with $P_{\text{notch}}=0.1$

One of the last sets of experiments performed within the AWS Ireland data center where the four collections with $P_{\text{notch}}=0.1$. A sample of the global packet distribution traffic observed by the receptor is illustrated in Figure 6.29. The first three notches previously identified in are visible in the figure, but this could be only because the periods of the
Table 6.21: Friedman ANOVA p-Values results at AWS Ireland with $P_{\text{notch}}=0.10$

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server</th>
<th>host 1</th>
<th>host 2</th>
<th>host 3</th>
<th>host 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000222</td>
<td>0.836375</td>
<td>0.005167</td>
<td>0.063361</td>
<td>0.355198</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>2</td>
<td>0.018288</td>
<td>0.230034</td>
<td>0.318776</td>
<td>0.375853</td>
<td>0.004249</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>3</td>
<td>0.044098</td>
<td>0.316222</td>
<td>0.097548</td>
<td>0.108109</td>
<td>0.967489</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>4</td>
<td>0.000001</td>
<td>0.131837</td>
<td>0.176560</td>
<td>0.086567</td>
<td>0.000121</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
</tr>
</tbody>
</table>

notches were known in advance. At this lower signal to noise ratio, there are a number of other spikes in the packet counts which could be mistaken for being a notch else where in the window.

The Friedman ANOVA tests were performed on all four data sets collected in the Ireland data center, and the resultant p-values are provided in Table 6.21. Only one data set indicated that there was a statistically significant deviation in the packet counts. The Friedman ANOVA test with 95% significance level (alpha=0.05) with the Bonferroni correction (n=99) only returned a significant result for data set 1. In this case, the Friedman ANOVA test can then be applied to the traffic originating from the individual hosts, and none of the test results return a statistically significant result. Data sets 2 and 3 return non-significant results for all tests, but data set 4 returns an interesting result. In data set 4, the Friedman ANOVA tests return a non-significant results for all tests except for the traffic originating from host 4.

The Wilcoxon rank-sum tests of the packet distributions on host 4 from data set 4 do not seem to indicate that the embedded quorum signal is the source of the Friedman ANOVA test’s significant result. The results of the Wilcoxon rank-sum test filtered on the packet distributions from host 4 are illustrated in Figure 6.30. Indeed most of the
Figure 6.29: AWS Ireland Receptor observed packet counts, with $P_{\text{notch}}=0.1$ (Data Set 1)
Figure 6.30: Wilcoxon rank-sum statistically significant periods at AWS Ireland with $P_{\text{notch}} = 0.10$ (Data Set 4)

periods which are statistically significant are not associated with the predicted location of the notch. Of the six periods which are statistically significant only two are associated with the notch (periods 57 and 58).

In this case, when the experiment was performed with $P_{\text{notch}} = 0.1$, most of the periods which were statistically significantly different were not located where the notch was expected to occur as a result of embedding the quorum sensing signal. A network defender attempting to locate a signal is in danger of making a type I error, and accepting a false-positive.
No Quorum Sensing Signal

In order to characterize the default state of the network without the quorum sensing emitters embedding the quorum sensing signal into the network traffic, two sets of data collections were performed with the receptor residing in the AWS Ireland data center. The Friedman ANOVA tests were performed on both data sets and the results were a p-value=0.328391 and a p-value=0.354148. With these p-values, the null-hypothesis cannot be rejected. This was expected as the quorum sensing signal was not embedded in the network traffic and it means that there are no statistically significant deviations in the packet count distributions of the individual periods.

6.2.5 Tokyo AWS Virtual Environment

In parallel to the data collections in Ireland, the AWS Tokyo data center was used to host another quorum sensing receptor. This section details the results collected when the receptor was located in the AWS Tokyo data center and the receptors were located at FloridaTech.

Quorum Sensing Signal with $P_{\text{notch}}=1.0$ and $P_{\text{notch}}=0.5$

As the round-trip time for the packets between the FloridaTech emitters and the receptors located in the AWS Tokyo data center was approximately 200 ms, it was expected that if the notch was observed it would be delayed from period 50 to approximately period 60 in the packet distributions observed by the receptor. An example of the traffic observed by the receptor is illustrated in Figure 6.31.

The packet distributions were similar to the traffic observed in the AWS Oregon and AWS Ireland data centers, in that multiple notches were observed. Two different
notches were observed, and consistently observed in the packet distributions recorded by the receptor in the AWS Tokyo data center. The notches are in periods 60-63 and 79-82 which are consistent with the prediction that the Tokyo data center is approximately 100 ms delayed (or 10 periods). The lack of the second notch at around periods 70-63 may mean that the vCPUs are scheduled to time share the CPU in 200 ms. This would only be able to be confirmed by gaining a better understanding about the time sharing processes that Amazon has implemented in their data centers.

The Friedman ANOVA tests were performed on both pairs of data collections, two $P_{\text{notch}}=1.0$ and two $P_{\text{notch}}=0.5$ data collections with a significance level of 95% (alpha=0.05) and the Bonferroni correction was applied. The resultant p-values were 0.000000 in all for cases for the global packet count distributions, which resulted in the rejection of the null-hypothesis. In addition when the packet distributions were filtered by their origin and the Friedman ANOVA tests were re-run, all of the tests again returned statistically significant results. This was the expected result and is consistent with the previous data collections when $P_{\text{notch}}=1.0$ or $P_{\text{notch}}=0.5$.

**Quorum Sensing Signal with $P_{\text{notch}}=0.25$**

Another set of four data collections was performed at the AWS Tokyo data center, during these collections $P_{\text{notch}}$ was set to 0.25. Previous when data collections were performed with $P_{\text{notch}}=0.5$, the global distribution displayed statistically significant deviations while the individual host’s packet count distributions did not display statistically significant deviations. Figure 6.32 illustrates the packet counts that were observed by the receptor for the first data collection. Two of the notches are easily observed round periods 60-62 and 79-81.
Figure 6.31: AWS Tokyo Receptor observed packet counts, with $P_{\text{notch}}=1.0$ (Data Set 1)
Figure 6.32: AWS Tokyo Receptor observed packet counts, with $P_{notch} = 0.25$ (Data Set 1)
A series of Friedman ANOVA tests were performed on the packet distributions recorded during the four data collections, the results are summarized in Table 6.22. Like all of the other Friedman ANOVA tests performed during this phase of the research, the significance level was set at 95% (alpha=0.05) and the Bonferroni correction (n=99) was applied to adjust for the repeated measurements. All of the global packet distributions observed by the receptor contained statistically significant deviations in the periods. As a result the null-hypothesis was rejected and follow-up testing with the Wilcoxon rank-sum tests should be performed.

Friedman ANOVA tests were repeated after the packet distribution were limited so only packets originating from a single source were present in the distribution to be tested. Half of the data sets indicated that the individual host’s traffic did not exhibit any statistically significant deviations (data collections 1 and 4). While the data collections 2 and 3 had one or more hosts which had statistically significant deviations as the null-hypothesis was rejected.

The Wilcoxon rank-sum tests were performed on the packet distributions with a significance level of 95% (alpha=0.05). Figure 6.33 contains the box plots for the period which contained statistically significant results from the Wilcoxon rank-sum tests. Most

---

Table 6.22: Friedman ANOVA p-Values results at AWS Tokyo with \( P_{notch} = 0.25 \)

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server</th>
<th>host 5</th>
<th>host 6</th>
<th>host 7</th>
<th>host 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>0.024135</td>
<td>0.003405</td>
<td>0.001474</td>
<td>0.004297</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>2</td>
<td>0.000001</td>
<td>0.009400</td>
<td>0.002375</td>
<td>0.019332</td>
<td>0.000131</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
</tr>
<tr>
<td>3</td>
<td>0.000000</td>
<td>0.000126</td>
<td>0.246393</td>
<td>0.465483</td>
<td>0.000000</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(s)</td>
</tr>
<tr>
<td>4</td>
<td>0.000044</td>
<td>0.004675</td>
<td>0.624417</td>
<td>0.050651</td>
<td>0.516399</td>
</tr>
<tr>
<td></td>
<td>(s)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>
Figure 6.33: Wilcoxon rank-sum statistically significant periods at AWS Tokyo with $P_{\text{notch}}=0.25$ (Data Set 1)

of the periods displayed in the Figure are below the median set, including two sets of periods which contain the two notches.

**Quorum Sensing Signal with $P_{\text{notch}}=0.1$**

The set of tests with an active quorum sensing emitter to be collected utilized the scheduler with $P_{\text{notch}}=0.1$ and the receptor in the AWS Tokyo data center. Again, 4 data collections were performed and the global packet counts observed by the receptor for the first data collection are illustrated in Figure 6.34. At this signal to noise level, the notches are not clearly visible in the traffic observed by the receptor.

Although Figure 6.34 does not visually appear to have any notches in the expected periods (i.e. periods 60-62 and 79-81) another data collection (illustrated in Figure 6.35) appears to have a set of notches in the expected locations. The Friedman ANOVA test was used to determine if the notches are statistically significant.

215
AWS Tokyo Network with Active DQS Scheduler $p(\text{notch})=10\%$ (Host aws-tokyo-t2.micro)

Figure 6.34: AWS Tokyo Receptor observed packet counts, with $P_{\text{notch}}=0.1$ (Data Set 1)
Figure 6.35: AWS Tokyo Receptor observed packet counts, with $P_{notch}=0.1$ (Data Set 3)
Table 6.23: Friedman ANOVA p-Values results at AWS Tokyo with $P_{\text{notch}}=0.10$

<table>
<thead>
<tr>
<th>Data Set</th>
<th>server</th>
<th>host 5</th>
<th>host 6</th>
<th>host 7</th>
<th>host 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.581010</td>
<td>0.159043</td>
<td>0.810335</td>
<td>0.229621</td>
<td>0.491682</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>2</td>
<td>0.781683</td>
<td>0.085629</td>
<td>0.679896</td>
<td>0.953328</td>
<td>0.601072</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>3</td>
<td>0.042019</td>
<td>0.034366</td>
<td>0.626823</td>
<td>0.797206</td>
<td>0.048096</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
<tr>
<td>4</td>
<td>0.089885</td>
<td>0.281733</td>
<td>0.127946</td>
<td>0.221644</td>
<td>0.165242</td>
</tr>
<tr>
<td></td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
<td>(ns)</td>
</tr>
</tbody>
</table>

Friedman ANOVA tests were performed on the packet distributions observed by the receptor in the AWS Tokyo data center and the resulting p-values are listed in Table 6.23. As can be seen from the p-Values listed in the table, none of the Friedman ANOVA test results indicate that there is a statistically significant deviation in the traffic observed by the receptor.

Although the Friedman ANOVA tests indicated that there were not statistically significant deviations in the observed packet distributions, a network defender may be tempted to perform the Wilcoxon rank-sum tests to see if anything significant is reported. Figure 6.36 illustrates the results of the Wilcoxon rank-sum test with a significance level of 95% ($\alpha=0.05$). None of the periods associated with the predicted location of the notch (i.e. periods 60-62 and 79-81) as listed as being statistically significant.

Unlike the Wilcoxon rank-sum test results for data set 1, data set 3 has some results which align with the predicted notch locations. Figure 6.37 illustrates the box plot results for the periods which are statistically significantly different from the data sets’ median values. In this case the first period is identified, periods 60-62 are all listed as being statistically significantly different. While most of the periods associated with
The second notch are also identified. Periods 80 and 81 were identified as containing statistically significant deviations while 79 did not have a statistically significant result.

**No Quorum Sensing Signal**

Finally a set of data collections was performed with the receptor in the AWS Tokyo data center in which the hosts were transmitting packets without any embedded signal. The based on the results of similar experiments performed as part of this research, it was predicted that there would be no significant findings as a result of the statistical
Figure 6.37: Wilcoxon rank-sum statistically significant periods at AWS Tokyo with $P_{\text{notch}}=0.10$ (Data Set 3)

The Friedman ANOVA tests were performed on the global packet distributions observed by the receptor and the resultant p-values were 0.810061 and 0.972423. With a significance level of 95% (alpha=0.05) and the Bonferroni correction (n=99) the p-values indicate that the results are non-significant and the null-hypothesis cannot be rejected. This was expected as the quorum sensing signal is not been embedded in the packets being transmitted to the receptor.
6.3 Discussion

In total this phase of the research conducted 67 data collections representing 4,020 hours of experimental data collections for 5 hosts (1 server host the receptor and 4 clients hosting the emitters). In its entirely this phase of the experiments resulted in collection and analysis of approximately 20,000 hours of data. Based on the data that was observed and the results of the statistical testing several different conclusions can be drawn.

As a result of the experiments performed it can be determined that the quorum sensing signal can be detected in a number of different types of environments. Previously it was demonstrated that the signal can be observed in an isolated environment with physical machines, the results from this phase indicate that the quorum sensing signal can be used in some virtual environments. This difference exposed a weakness which could be exploited by network defenders to potentially disrupt C2 channels using the quorum signal discussed in this work. The signal failed in two different situations; 1) when the signal was used on machines without time synchronization (demonstrated in Section 5.2.1) and 2) in situations when the clock synchronization is not stable in virtual environments, such as those encountered with GENI (see Section 6.1.2). A network defender could disrupt the quorum signal’s coordination by allowing the clocks on the local network to drift or intentionally causing a divergence between the clocks on the network.

The next significant result is that there are specific $P_{\text{notch}}$ values which asymmetrically allow the determined adversary to act but not the network defender. In some cases, typically when $P_{\text{notch}}=0.25$ there are cases when the Friedman ANOVA test results indicate that there are statistically significant deviations occurring in the network traffic at the global level, but when follow up tests are conducted on specific hosts non-
significant results are returned. This means that a signal was observed at the network level which anyone on the network could observed but when filtering is put in place non signal entity on the network can be determined to be a likely source of the signal. The exact $P_{\text{notch}}$ value that is used to achieve this result varies by network. In the case of the AWS Tokyo receptor the value is likely between [0.25,0.1] but significantly closer to a value of 0.25. Likewise for AWS Ireland, the value is also in the range [0.25,0.1] but probably closer to the middle of this range as the 0.25 cases has a large number of results which the individual contributions were still significant.

Another interesting observation is that if the results of the Friedman ANOVA test returns a non-significant result, the network defender still has the opportunity to attempt to perform a set of Wilcoxon rank-sum tests. In doing the Wilcoxon rank-sum tests when the Friedman ANOVA test returns a non-significant result, the network defender risks dealing with sets of false-positives. In Figure 6.36 not one of the periods listed as being statistically significant was associated with the notch’s predicted periods, but on the other hand Figure 6.37 contained almost all of the periods associated with the notches.

In conclusion this phase of the research was extremely productive in gaining an understand of the behavior of the quorum sensing signal in more realistic network conditions. It demonstrated that the quorum signal can be created in a situations only relying upon existing network traffic and that under certain conditions the signal is only beneficial to the determined adversary. This work does not only benefit the determined adversary, an important result is that this signal can be trivially disrupted by allowing clocks to drift normally. It seems that clock stability is more of an issue for certain types of virtualization systems. The Ubuntu servers running at FloridaTech and the AWS t2.micro instances running NTP were able to maintain sufficient clock accuracy to allow
the signal to be generated and detected. Other virtualization platforms were unable to
provide for sufficient clock accuracy but the clocks were accurate to allow other services
to be provided (i.e. experiments running in GENI and AWS t2.medium instances).
Chapter 7

Potential Applications and Defenses

The goal of this research was to characterize a potential C2 channel which could be exploited by determined adversaries allowing the security community time to begin identifying potential mitigation strategies and defenses before this C2 channel is seen in the wild. The characterization of the C2 channel was successful and the next portion of the research is focused on understanding the architecture, determining different methods which can be used to defend against this type of communication scheme.

7.1 Biologically Inspired Defenses

Given that the underlying approach was inspired by quorum sensing, it would be of interest to see how natural systems detect and disrupt quorum sensing systems. Two different methods for dealing with quorum sensing systems were uncovered during the literature review.

A proposed treatment for *P. aeruginosa* [40] is to infuse the environment with quorum sensing signals which cause pathogenic bacterial cells to transition to their
social states. In this state *P. aeruginosa* begins production of virulence factors. If an infection has not been established, the bacteria will begin the production of virulence factors and trigger a response from the host immune system. The immune system will be able to identify the presence of relatively few pathogenic bacterial cells and clear the infection. A similar strategy could work with digital quorum sensing. If a piece of targeted malware is identified and/or a quorum sensing signal has been identified on the network, the signal could be enhanced to cause a state transition in the malware. Prematurely triggering the C2 channel would disrupt the determined adversary’s mission. If the population of infected hosts is relatively small this could help identify infected hosts by anomalous behavior when they start interacting with the environment, but this would need to be identified before the malware has achieved a quorum.

Another strategy that has been used to prevent bacteria from using quorum sensing is the use of Quorum Sensing Interference and Quorum Sensing Inhibition [148, 74, 64]. Some types of eukaryotes such as algae and types of plants produce and release a number of different QSI compounds. Some chemicals interact with the autoinducers produced by the bacterial cells and prevent the autoinduction feedback loop from occurring as the autoinducers are degraded before the quorum receptors activate. These types of compounds are also referred to as quorum quenching compounds [47, 147, 14, 135]. Implementing a defense based on quorum quenching in a network is more difficult than simply enhancing the potential signal and hoping for the best. One possible strategy would be to flood the network with a large amount of traffic that has no effect on the network’s overall mission but has a well defined pattern. The network can be characterized and monitored by the network defender. When potential deviations in the network have been identified, the extra network traffic can be returned to previous
distribution. The network traffic patterns such that minor signals which could have previously been observed due to the strong consistency of the traffic are now lost due to the extra noise in the traffic.

7.2 Exploiting Weaknesses in the Architecture

Another method for designing possible defenses against malware employing a quorum sensing C2 channel is to review the architectural requirements of the C2 channel and design defenses that exploit potential weaknesses. Digital quorum sensing provides a strategy that is robust and resistant to directed remediation by the defender.

In Gray III’s discussion of counting covert channels [66], two possible countermeasures were proposed; probabilistic partitioning and fuzzy time. Both of these methods have the potential to disrupt a quorum sensing signal. The autoinducers described in the quorum sensing architecture are based on inserting a pattern into the distribution of existing network traffic. As described by Gray III, using probabilistic partitioning it might be possible to ensure that during a given interval the distribution of packets are shaped into a known pattern. A host emitting packets that correspond to a different distribution could be identified as anomalous and possibility infected. As Gray III recommended, this distribution would change over time so the malware would need to adapt to a changing signal. The probabilistic partitioning method might not to be practical without substantial modifications to the networking stack. The second method relying on fuzzing time has the potential to disrupt the signal was been confirmed as a potential defense against observing the signal on the network during this research. The fuzzy time method of disrupting a covert channel works by disrupting the time source on the host. Covert timing channels require an accurate time source to schedule and
coordinate the encoding and decoding of signals. Erratically shifting the time source could disrupt the synchronized encoding and decoding of messages. During the initial validation stages of the prototype, the effect of normal clock drift was sufficient to disrupt the quorum sensing signal (see section 5.2.1). Furthermore the effects were again replicated during the preliminary deployments within the GENI environment (see section 6.1.2).

The current architecture relies on determining the time that a packet was scheduled to be transmitted and then periodically injecting a slightly delay into the scheduler. Monitoring a periodic reconfiguration of the scheduler could be a reliable method for detecting quorum sensing emitters on the host, but it is not likely that a determined adversary would employ the scheduler that was used to conduct the experiments done in this work. They are more likely to develop a custom library that is either injected into the kernel or networking libraries to cause these periodic delays. Regardless of the ultimate implementation that is used, the packet schedule will need to be periodically delayed, and this delay will require a series of time checks against the local time source. The packet scheduler used in this research required two time checks every 10 ms while the scheduler was running. This resulted in a single process checking the system time 200 times every minute or approximately 72,000 times in six hours. A process implementing the quorum sensing scheduler will obsess about the current time. Checking for processes and libraries that are constantly checking the current system time could be used as an indicator that the process is behaving as a quorum sensing scheduler. Alternate methods can be implemented without relying on the external scheduler which could be used that do not cause as many interactions with the local system’s clock.
7.3 Applications for Digital Quorum Sensing

Another important aspect of this research was to attempt to determine other possible malicious applications of digital quorum sensing. It is possible that a determined adversary may attempt to leverage digital quorum sensing in situations outside the scope of this research.

There are a few differences between bacterial quorum sensing and digital quorum sensing; one important aspect is that bacteria exist in a three-dimensional world. Computers are not limited to sharing information across a wired NIC. Information can be exchanged via WiFi and acoustics [72, 73, 93]. It is possible that a determined adversary may use a different mechanism for exchanging signals.

Bacterial quorum sensing works in a three-dimensional environment. Adjusting the perspective to account for a more three-dimensional approach might indicate a few different methods in which determined adversaries may attempt to apply digital quorum sensing. As previously mentioned WiFi and Acoustics could also be used as they allow communication in three-dimensions. If a WiFi or Acoustical signal is used, the overall infected population density would be estimated by the overall signal strength. At first glance it may not be a useful application of digital quorum sensing due to the network isolation of high-valued networks. If the determined adversary is not interested in attacking a high-value network that is isolated, then digital quorum sensing may become a viable solution for estimating the infected population density.

Another avenue of potential research is use of digital quorum sensing with mobile devices. Work has already been presented on exfiltrating information to mobile devices using induced WiFi [69] and acoustical channels [42]. AirHopper is an example of a system which attempts to exfiltrate information from an air-gapped network by
displaying specific patterns on a screen to create a WiFi signal which can be intercepted by a roaming mobile device [69]. The difference between the C2 channel proposed in this research and AirHopper is that only a single host is exfiltrating information. Deshotels presented another method for covertly exchanging information between infected mobile hosts [42] by using portions of the audio spectrum which are inaudible to humans.

7.4 Applications based on Information Asymmetry

All of the previous discussions on applications and defenses have focused on applying quorum sensing as a C2 channel. The digital quorum sensing channel represents a situation in which both the adversary and defender have access to the same information on which only the adversary can act. This observation opens a potential new path for future research and was proposed at the New Security Paradigms Workshop 2016 conference [53] and this section discusses the work presented at the NSPW conference.

Traditional computer defenses, such as Address Space Layout Randomization (ASLR), rely on the defender having access to more complete information than the adversary. In ASLR, the memory layout is randomized from the perspective of the adversary. The adversary does not know the memory addresses of functions that it wishes to utilize. The defender is aware of the memory layout and thus is able to function without interruption. This information asymmetry can be expressed as follows:

\[ H(X|P = a) > 0 \text{ and } H(X|P = d) = 0. \]

\( P \) is a random variable selected from the \{attacker, defender\} set. The variable \( X \) represents information in the system that has been encoded and is of interest to both the attacker and defender. Finally \( H(x) \) represents the amount of uncertainty associated
with the variable $x$. Traditional defenses, such as ASLR, assume that the attacker has a higher level of uncertainty than the defender. Unfortunately with ASLR, the attacker can exploit information that the defender is leaking and transition to a state in which they have the same level of uncertainty.

An important aspect of the model is that it results in a set of key requirements for systems relying on uncertainty. These requirements are as follows:

- **Global availability:** The objects represented by $X$ must be available to all parties involved (i.e. the attacker and the defender).

- **Stochastic processes:** The specific values associated with $X$ cannot be deterministic. The possible values of $X$ can only be statistically described before an actual value of $X$ is produced. If the value could be predicted by either party then the uncertainty for that party with respect to the variable is 0.

- **Synchronization:** The process is structured such that a synchronization element is present, it must be assigned before it can be accessed.

In the NSPW paper, two different applications of this model were discussed. The first application was to enhance ASLR such that both the attacker and defender had a non-zero uncertainty when attempting to access memory. The application of the uncertainty model to digital quorum sensing was the second discussion.

### 7.4.1 The Magic Memory Management Unit

The first application of the uncertainty model discussed in the NSPW paper focus on replacing traditional Memory Management Units (MMUs) with a Magic Memory Management Units (MMMs). MMUs are specialized pieces of hardware which
converts virtual addresses into physical addresses. The proposed MMMU; works by generating a list of potential virtual addresses and providing these to the client process. All memory access is restricted such all memory access attempts must pass through the MMMU, either the adversary nor the defender has direct access to physical memory. Because of this restriction, both defender and the adversary have uncertainty in the layout of memory.

The MMMU provides the process with a set of virtual addresses which can be used to access memory and the desired resource. There are two types of virtual addresses returned in this set: valid and tracked. Valid virtual addresses will result in the MMMU providing access to the desired memory location, while Tracked virtual addresses will not allow access and result in an exception. Furthermore, when a Valid virtual address is accessed, the MMMU reissues the set of virtual addresses to the calling process. When an address is submitted to the MMMU that is not either a valid or tracked virtual address, the MMMU will signal the CPU to terminate the calling process.

The MMMU provides for the three key requirements identified by the model in section 7.4. More explicitly, the requirements identified by the model are addressed as follows:

- **Global availability**: The MMMU mediates all memory access as such it is available to both the attacker and defender.

- **Stochastic processes**: The virtual memory addresses are generated randomly as such they are only quantifiable stochastically.

- **Synchronization**: The value of the virtual memory address is assigned by the MMMU and must be assigned before either the attacker or defender can attempt to access the memory location.
When the application is compiled, the application’s functions are wrapped in a trampoline. The trampoline works on the set of virtual addresses, it tries a random virtual address from the set of virtual addresses provided by the MMMU. If the address is unsuccessful, the trampoline tries another address in the set. The function call is guaranteed to succeed as it is a given that at least one of the virtual addresses is valid and will permit access to the desired function.

It is important to highlight that both the attacker and defender do not have information about the true location of the function in memory. Furthermore, if the attacker manages to extract the address set from the defender’s process and call the function, the defender will know when their virtual address set is replaced by the MMMU. As the defender is guaranteed to have either a valid or tracked virtual address, their process will not be terminated during execution. While a defender who is attempting to guess virtual addresses will routinely have their process terminated by the MMMU signaling the CPU.

### 7.4.2 Uncertainty in Digital Quorum Sensing

The model discussed in Section 7.4 applies to quorum sensing. In Sections 2.1.3 and 2.4 it was discussed that the adversary and the defender are operating under different constrains which has the potential to lead to situations in which only one party can act.

Digital quorum sensing satisfies the three requirements which are needed for the uncertainty model discussed in Section 7.4. The relationship between the key requirements in the model and digital quorum sensing are as follows:

- Global availability: The global packet distributions is globally available on flat networks. Furthermore, the distribution is the cumulative result of all of the
individual nodes on the network. The distribution is also a function of directed and broadcast traffic meaning that all communications on the network have the potential to influence the global packet distribution.

- **Stochastic processes:** Although the individual protocols in use on the networks are deterministic in that all of the requests and responses are coordinated, the exact time and application of the protocol varies based on use and implementation.

- **Synchronization:** All of the entities on the network have their own time sources and modern networks often require the use of time synchronization protocols such as NTP.

This descriptive model is useful in that it allows the critical characteristics of digital quorum sensing system to be identified. The model allows the network defender to identify possible weaknesses which can be exploited to disrupt or disable the C2 channel. The synchronization component was demonstrated to be a potential weakness during the validation of the prototype in the isolated network (see Section 5.2.1) and during the attempted deployment to GENI (see Section 6.1.2).
Chapter 8

Conclusions

Malware developed by determined adversaries is a threat to computer networks. Determined adversaries are constantly developing novel methods for compromising and maintaining a presence on target networks. Determined adversaries and network defenders have different tolerances for uncertainty. Network defenders rely on and often require little or no uncertainty to perform their duties. Adversaries may be able to tolerate a higher level of uncertainty in the operational environment and be able to accomplish their objective in the presence of those uncertainties.

A biologically-inspired C2 channel, digital quorum sensing, was discussed and two implementations of this approach were created during this research. It is important to note that no malicious software was utilized during any of this research. The software that was created for this research only simulated host traffic which was used to characterize the C2 channel. The emitter and receptor developed for the proof of concept allowed the basic concept to be tested in a small number of different environments. The initial proof of concept generated novel traffic which could be used to induce a pattern in the traffic generated by a host. A small number of tests were performed to determine
if the C2 channel was viable. The results indicated that the C2 channel could likely be used in a normal networking environment and the results of this work were published at the MALCON 2014 conference[54].

Based on the results from the proof of concept, a new prototype was designed and implemented. During the design phase of the research, it was anticipated that the prototype would be validated in a small isolated network and then attempts to characterize the C2 channel would be performed in a large interconnected experimental testbed such as GENI.

The emitter was significantly redesigned for the prototype phase of the study. The design was changed from one in which the emitter generated novel packets to force the transmitted traffic into a new packet distribution to a design which relied on existing packets being generated by the host. The new design entirely relied on changing the transmission schedule of the packets on the host (i.e. the transmission times), which means that the network defender could not identify the quorum sensing traffic based on the characteristics of the packet’s contents. During the first tests with the redesigned emitter, it was determined that the statistical tests for the detection of the quorum sensing signal were inappropriate as the packet distributions were non-parameteric. The statistical tests for the receptor were changed so that the tests applied were more statistically sound.

The prototype produced two significant results during the experiments. The first successful result was a demonstration that a signal could be produced by a small network of emitters which was statistically-significant when the global packet distribution was tested, while the same statistical tests applied to the observed packet distribution from a single host did not produce statistically significant results. This observation meant that both the adversary and defender could observe a statistically significant anomaly in
the global packet distribution on the network but neither the adversary nor the defender could determine which specific hosts were responsible for the anomaly. This result is not a problem for the adversary who is only interested in determining if the network is sufficiently infected to begin the next phase of its operations. Conversely it is a problem for the network defender who is unable to determine which hosts are infected so they can be removed from the network and the system can be cleared of malware. The second result from the tests in the isolated network was that a possible defense against the implemented quorum signal was identified. During the initial validation of the emitter, it was noticed that the emitter was clearly transmitting a quorum sensing signal but the receptor was unable to observe the signal. The emitter’s host was reconfigured and NTP synchronization services were installed. After the NTP services were installed and the validation test was performed again, the receptor was able to observe the signal. The quorum sensing signal is sensitive to the natural clock drift of the host. It was so sensitive that the natural clock drift of the host was enough to prevent the receptor from observing the signal. Either allowing the clock to naturally drift or by the defender inducing a drift, it may be possible for network defenders to disrupt these signals on their network.

Unfortunately, the experiments in GENI could not be performed as planned. The clocks on the virtual hosts were not stable enough to allow even an obvious quorum sensing signal, such as the $p_{\text{notch}}=1.0$, to be observed. The planned experiments were altered to be conducted in various Amazon EC2 data centers. The receptor was placed in different AWS data centers in different geographic locations: northern Virginia, Oregon, Ireland and Tokyo. This arrangement allowed the signal to be characterized in the presence of different network distances and network noise. Despite the receptor being located on the other side of the Atlantic or Pacific Ocean with network delays on the order of 100 ms, the receptor was still able to observe the quorum sensing signal.
During the final phases of the experiments a formalized uncertainty model was developed and published at the NSPW 2016 conference[53]. The descriptive model identified three key requirements which could be used to identify possible situations in which both network defenders and adversaries could attempt to leverage to their advantage. This model also highlights potential weaknesses which could be used to develop countermeasures and defenses; specifically the model highlighted that a method of synchronization must be used in order for the signal to be used. This weakness was demonstrated during the prototype validation on the small network and the problems observed with GENI’s time synchronization.

Another important aspect of the research that was highlighted in the paper submitted to the NSPW 2016 conference is the idea that both the network defender and the determined adversary are operating with different tolerances towards uncertainty in information. Not only are there different tolerances but the tolerance changes depending upon the situation. The experiments in this research focused on characterizing a C2 channel, which both the defender and the adversary can observe but only the adversary can act upon. This scenario is not the only situation in which the different levels of uncertainty can be exploited. The NSPW paper discussed a hypothetical Magic Memory Management Unit, which altered how processes interact with memory. In this situation, either the defender or the adversary had complete information about the system but the defender’s processes were not impacted. Contrary to traditional defenses such as ALSR, the MMMU cannot be bypassed by leaking information. The adversary achieves little by gaining access to the information that the defender has about memory, “the adversary cannot steal something that the defender does not have”. The defender only has a set of possible memory addresses and they do not know which key is the real key.
This work to characterize the C2 channel resulted from the analysis of approximately 20,000 hours of network packet captures. Despite the large number of data collections performed there are still areas in which future studies can be conducted. Possible future work includes characterizing the C2 channel while the number of hosts and their distance is varied during the same data collection. Another potential avenue for expanding upon this research is to implement a feedback loop between the emitter and receptor to allow it to dynamically modifying the values of \( p_{\text{notch}} \) as a quorum is established. Finally, another potential area is to apply advanced machine learning techniques to the receptor to determine if there are other methods which can be reliably used to detect the quorum sensing signal.

In conclusion, this work has demonstrated that there are situations in which both the adversary and network defender can observe the same signal on the network, yet because the network defender is constrained by their uncertainty requirements only the adversary is able to act. During the experiments, a statistically significant signal in the global packet distribution was observed but the same statistical tests could not reliably indicate which hosts were contributing to the signal. As a network defender may be reluctant to take an entire network down, it may not be able to act in these situations. Fortunately it was discovered that simple defenses such as allowing clocks to naturally drift can disrupt the quorum sensing signal. The techniques described in this research have the potential to be developed into applications in which the defender relies on uncertainty as defense against attacks.
Chapter 9

List of Publications

The following are the list of publications that have been produced during the course of this Ph.D. research related to Digital Quorum Sensing.

Papers published


The following are the list of publications that have been produced during the course of this Ph.D. research which are not related to Digital Quorum Sensing.
**Other papers published**


Bibliography


[33] Eric Chien, Liam OMurchu, and Nicolas Falliere. W32.Duqu: The precursor to the next stuxnet. 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), 4:2, 2012.


[126] Taeshik Sohn, Jongsub Moon, Sangjin Lee, Dong Hoon Lee, and Jongin Lim. Covert channel detection in the ICMP payload using support vector machine. *Computer and Information 


256


