Honeyhive - A Network Intrusion Detection System Framework Utilizing Distributed Internet of Things Honeypot Sensors

Zachary D. Madison

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Information Security Commons

Recommended Citation
https://scholar.afit.edu/etd/3179

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.
HONEYHIVE – A NETWORK INTRUSION DETECTION SYSTEM FRAMEWORK UTILIZING DISTRIBUTED INTERNET OF THINGS HONEYPOT SENSORS

THESIS

Zachary D. Madison, Capt, USAF
AFIT-ENG-MS-20-M-038

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.
HONEYHIVE – A NETWORK INTRUSION DETECTION SYSTEM FRAMEWORK UTILIZING DISTRIBUTED INTERNET OF THINGS HONEYPOT SENSORS

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science

Zachary D. Madison, B.S.C.S.
Capt, USAF

March 2020

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.
HONEYHIVE – A NETWORK INTRUSION DETECTION SYSTEM FRAMEWORK UTILIZING DISTRIBUTED INTERNET OF THINGS HONEYPOT SENSORS

THESIS

Zachary D. Madison, B.S.C.S.
Capt, USAF

Committee Membership:

Barry E. Mullins, Ph.D., P.E.
(Chairman)

Scott R. Graham, Ph.D.
(Member)

Stephen J. Dunlap, M.S., CISSP
(Member)
Abstract

With the ever increasing number of Internet-connected devices, the importance of cyber security also increases. Exploding over the past decade, the number of Internet of Things (IoT) devices connected to the Internet jumped from 3.8 billion in 2015 to 17.8 billion in 2018. A major concern with many IoT devices is that they contain vulnerabilities that are often left unpatched. To make matters worse, many of these IoT devices lack modern security measures found on traditional computing devices, due to their inherent hardware limitations and vendors focusing on functionality and time to market over security.

Honeypots are devices not part of routine network usage that are meant to alert of an attacker’s presence, capture an attacker’s tools, and record their Tactics, Techniques, and Procedures (TTPs). Honeyd is a framework capable of rapidly creating low-interaction honeypots by simulating the network stack.

HoneyHive is a framework that uses distributed IoT honeypots as Network Intrusion Detection System (NIDS) sensors that beacon back to a centralized Command and Control (C2) server. This research uses the three Honeyd IoT honeypots developed by Stafira, but HoneyHive is flexible enough to support any device capable of running the Python HoneyB Agent script. HoneyHive offers a method for network intrusion detection using the lure of vulnerable IoT devices as distributed honeypot intrusion detection sensors.

The HoneyHive framework consists of the C2 server, transfer server, Snort log parser, Database (DB), and the HoneyB Agent script. The C2 server interacts with all other components and displays honeypot interactions and Snort alerts to network operators in real time. The transfer server receives PCAPs and other Binary Large
Object (BLOB) from the HoneyB Agent. Upon receiving a PCAP, it is passed to
Snort and analyzed for matching signatures. Immediately after this, the Snort log
parser reads the log file and then sends any alert information back to the C2 server.
The DB stores alerts, alert metadata, and PCAPs received from the HoneyB Agent.
The HoneyB Agent script captures and reports any traffic interactions with Stafira’s
honeypots. Captured traffic is sent to the transfer server and honeypot interactions
are sent to the C2 server.

The tests in this experiment involve four types of scans and four levels of active
honeypots against the HoneyHive framework and a traditional NIDS on the simulated
test network. The scan types include No Scan (Control Group), TCP Connect scan,
Aggressive scan, and NIDS Avoidance scan. The levels for honeypots are 0, 3, 6, and
9 honeypots. Each of these was run in different combinations with one another for a
full factorial experiment resulting in 16 different combinations.

This research successfully created a framework of distributed network intrusion
detection IoT honeypot sensors that capture traffic, create alerts, and beacon back to
a central C2 server. The HoneyHive framework operated correctly by not alerting on
routine network traffic and alerting on non-routine network traffic. Additionally, the
HoneyHive framework successfully detected intrusions that traditional NIDSs cannot
through the use of distributed IoT honeypot sensors and packet capture aggregation.
To my Wife,
Thank you for all your love, support, and hand-crafted late night lattes. I am so proud of how diligent you work and all your accomplishments. I never want my career or goals to take priority over yours, “Love is taking turns riding shotgun.”

To my soon to be born baby girl,
You motivated your Daddy to work extra hard and finish early so that we could have your nursery all setup. I pray that you love the Lord, dream big, and never settle for what’s easiest, but fervently pursue justice.

To my Mom and Dad,
Thank you for all your love and encouragement through every step of my life. Thank you for instilling in me a work-ethic and a perseverance to overcome any adversity.

I love you both to the moon and back!

PLUS ULTRA!
Acknowledgements

“We all want progress. But progress means getting nearer to the place you want to be and if you have taken a wrong turning, then to go forward does not get you any nearer. If you are on the wrong road, progress means doing an about-turn and walking back to the right road; and in that case, the man who turns back soonest is the most progressive man.” - C.S. Lewis

To my advisor, Dr. Barry Mullins, thank you for your shared excitement, expertise, and keeping me from going down too many rabbit holes during my research.

Stephen, even though I grumbled at every design change to the HoneyHive framework and my experiment methodology, thank you. My thesis would not be near the product it is now without all your insight and wonderful suggestions.

Zachary D. Madison
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Dedication</td>
<td>vi</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>vii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiv</td>
</tr>
<tr>
<td>List of Acronyms</td>
<td>xv</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Research Goals</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Approach</td>
<td>4</td>
</tr>
<tr>
<td>1.4.1 HoneyHive Framework</td>
<td>4</td>
</tr>
<tr>
<td>1.4.2 Simulated Network</td>
<td>5</td>
</tr>
<tr>
<td>1.4.3 Experiment</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Assumptions and Limitations</td>
<td>6</td>
</tr>
<tr>
<td>1.5.1 Assumptions</td>
<td>6</td>
</tr>
<tr>
<td>1.5.2 Limitations</td>
<td>6</td>
</tr>
<tr>
<td>1.6 Research Contributions</td>
<td>7</td>
</tr>
<tr>
<td>1.7 Thesis Overview</td>
<td>7</td>
</tr>
<tr>
<td>II. Background and Related Research</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Overview</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Background</td>
<td>8</td>
</tr>
<tr>
<td>2.2.1 Internet of Things (IoT)</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2 IoT and Computer Network Security</td>
<td>9</td>
</tr>
<tr>
<td>2.2.3 Network Intrusion Detection System (NIDS)</td>
<td>12</td>
</tr>
<tr>
<td>2.2.4 Networking Monitoring</td>
<td>13</td>
</tr>
<tr>
<td>2.2.5 Honeypots</td>
<td>14</td>
</tr>
<tr>
<td>2.2.6 Honeytokens</td>
<td>15</td>
</tr>
<tr>
<td>2.2.7 Honeyd 1.5c</td>
<td>16</td>
</tr>
<tr>
<td>2.2.8 Cyber Deception</td>
<td>18</td>
</tr>
<tr>
<td>2.2.9 Programming and Languages</td>
<td>19</td>
</tr>
<tr>
<td>2.2.10 Tools</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Related Research</td>
<td>26</td>
</tr>
<tr>
<td>2.3.1 Conpot</td>
<td>26</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>2.3.2</td>
<td>IoT Web-Based Honeypots by Lukas Stafira</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Honeycomb by Christian Kreibich</td>
</tr>
<tr>
<td>2.3.4</td>
<td>Honeyd Syslog Solutions</td>
</tr>
<tr>
<td>2.3.5</td>
<td>IoTCandyJar</td>
</tr>
<tr>
<td>2.3.6</td>
<td>HoneyLab</td>
</tr>
<tr>
<td>2.3.7</td>
<td>SIPHON</td>
</tr>
<tr>
<td>2.3.8</td>
<td>HoneyIo4</td>
</tr>
<tr>
<td>2.3.9</td>
<td>IoTPOT and IoTBOX</td>
</tr>
<tr>
<td>2.3.10</td>
<td>Multi-Purpose IoT Honeypot</td>
</tr>
<tr>
<td>2.3.11</td>
<td>ThingPot</td>
</tr>
<tr>
<td>2.3.12</td>
<td>IoTSec</td>
</tr>
<tr>
<td>2.3.13</td>
<td>Honeycomb and MazeRunner by Cymmetria</td>
</tr>
<tr>
<td>2.3.14</td>
<td>Comparison of Related Frameworks</td>
</tr>
<tr>
<td>2.4</td>
<td>Chapter Summary</td>
</tr>
<tr>
<td>III</td>
<td>Framework Design</td>
</tr>
<tr>
<td>3.1</td>
<td>Overview</td>
</tr>
<tr>
<td>3.2</td>
<td>Motivation</td>
</tr>
<tr>
<td>3.3</td>
<td>Third-Party Software</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Honeyd 1.5c</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Stafira’s Honeypots</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Ubuntu 12</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Snort</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Suricata</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Wireshark</td>
</tr>
<tr>
<td>3.4</td>
<td>Programming Languages</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Node.js</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Python 2.7</td>
</tr>
<tr>
<td>3.4.3</td>
<td>SQLite</td>
</tr>
<tr>
<td>3.5</td>
<td>HoneyHive Framework Design</td>
</tr>
<tr>
<td>3.5.1</td>
<td>C2 Server, Transfer Server, and Snort Log Parser</td>
</tr>
<tr>
<td>3.5.2</td>
<td>HoneyB Agent</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Database Design</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Network Design</td>
</tr>
<tr>
<td>3.6</td>
<td>Summary</td>
</tr>
<tr>
<td>IV</td>
<td>Research Methodology</td>
</tr>
<tr>
<td>4.1</td>
<td>Goals</td>
</tr>
<tr>
<td>4.2</td>
<td>Approach</td>
</tr>
<tr>
<td>4.3</td>
<td>System Boundaries</td>
</tr>
<tr>
<td>4.4</td>
<td>Parameters, Factors, and Metrics</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Assumptions</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>4.4.2</td>
<td>System Parameters</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Factors</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Metrics</td>
</tr>
<tr>
<td>4.5</td>
<td>Methodology</td>
</tr>
<tr>
<td>4.5.1</td>
<td>runExperiment.py</td>
</tr>
<tr>
<td>4.6</td>
<td>Results</td>
</tr>
<tr>
<td>4.7</td>
<td>Chapter Summary</td>
</tr>
<tr>
<td>V.</td>
<td>Results and Analysis</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Number of Alerts Overview</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Number of Distinct Types of Alerts Overview</td>
</tr>
<tr>
<td>5.1.3</td>
<td>Packet Capture Percentage Overview</td>
</tr>
<tr>
<td>5.2</td>
<td>Scan Type</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Control Group</td>
</tr>
<tr>
<td>5.2.2</td>
<td>TCP Connect</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Aggressive</td>
</tr>
<tr>
<td>5.2.4</td>
<td>NIDS Avoidance</td>
</tr>
<tr>
<td>5.3</td>
<td>Number of Honeypots</td>
</tr>
<tr>
<td>5.3.1</td>
<td>0 Honeypots</td>
</tr>
<tr>
<td>5.3.2</td>
<td>3 Honeypots</td>
</tr>
<tr>
<td>5.3.3</td>
<td>6 Honeypots</td>
</tr>
<tr>
<td>5.3.4</td>
<td>9 Honeypots</td>
</tr>
<tr>
<td>5.4</td>
<td>Summary</td>
</tr>
<tr>
<td>VI.</td>
<td>Conclusions</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>6.2</td>
<td>Research Conclusions</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Number of Alerts</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Number of Distinct Types of Alerts</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Percentage of Packets Captured</td>
</tr>
<tr>
<td>6.3</td>
<td>Research Significance</td>
</tr>
<tr>
<td>6.4</td>
<td>Research Limitations</td>
</tr>
<tr>
<td>6.5</td>
<td>Future Work</td>
</tr>
<tr>
<td>Appendix A</td>
<td>HoneyHive Framework</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Honeyd Configuration File</td>
</tr>
<tr>
<td>Appendix C</td>
<td>suricataConnect.py</td>
</tr>
<tr>
<td>Appendix D</td>
<td>runExperiment.py</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Growth of IoT Devices from 2015-2025</td>
</tr>
<tr>
<td>2.</td>
<td>VMware Workstation Hardware Settings</td>
</tr>
<tr>
<td>3.</td>
<td>Virtual Machine Structure</td>
</tr>
<tr>
<td>4.</td>
<td>Container Structure</td>
</tr>
<tr>
<td>5.</td>
<td>Stafira’s Network Configuration</td>
</tr>
<tr>
<td>6.</td>
<td>IoT CandyJar Design</td>
</tr>
<tr>
<td>7.</td>
<td>HoneyLab Design</td>
</tr>
<tr>
<td>8.</td>
<td>SIPHON Overview</td>
</tr>
<tr>
<td>9.</td>
<td>Attacker’s Interaction with SIPHON</td>
</tr>
<tr>
<td>10.</td>
<td>IoTPOT Overview</td>
</tr>
<tr>
<td>11.</td>
<td>IoTBOX Overview</td>
</tr>
<tr>
<td>12.</td>
<td>ThingPot Overview</td>
</tr>
<tr>
<td>13.</td>
<td>HoneyHive Framework</td>
</tr>
<tr>
<td>14.</td>
<td>UML Program Design</td>
</tr>
<tr>
<td>15.</td>
<td>Database Schema</td>
</tr>
<tr>
<td>16.</td>
<td>Simulated Test Network - Network Layout</td>
</tr>
<tr>
<td>17.</td>
<td>HoneyHive Framework</td>
</tr>
<tr>
<td>18.</td>
<td>Overview - Mean Number of Alerts</td>
</tr>
<tr>
<td>19.</td>
<td>Overview - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>20.</td>
<td>Overview - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>21.</td>
<td>HoneyHive Framework Mean Packet Capture Percentage (% HHP) by Level</td>
</tr>
<tr>
<td>22.</td>
<td>Control Group - Mean Percentage of Packets Captured</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>23.</td>
<td>TCP Connect - Mean Number of Alerts</td>
</tr>
<tr>
<td>24.</td>
<td>TCP Connect - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>25.</td>
<td>TCP Connect - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>26.</td>
<td>Aggressive - Mean Number of Alerts</td>
</tr>
<tr>
<td>27.</td>
<td>Aggressive - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>28.</td>
<td>Aggressive - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>29.</td>
<td>NIDS Avoidance - Mean Number of Alerts</td>
</tr>
<tr>
<td>30.</td>
<td>NIDS Avoidance - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>31.</td>
<td>NIDS Avoidance - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>32.</td>
<td>0 Honeypots - Mean Number of Alerts</td>
</tr>
<tr>
<td>33.</td>
<td>0 Honeypots - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>34.</td>
<td>0 Honeypots - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>35.</td>
<td>3 Honeypots - Mean Number of Alerts</td>
</tr>
<tr>
<td>36.</td>
<td>3 Honeypots - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>37.</td>
<td>3 Honeypots - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>38.</td>
<td>6 Honeypots - Mean Number of Alerts</td>
</tr>
<tr>
<td>39.</td>
<td>6 Honeypots - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>40.</td>
<td>6 Honeypots - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>41.</td>
<td>9 Honeypots - Mean Number of Alerts</td>
</tr>
<tr>
<td>42.</td>
<td>9 Honeypots - Mean Number of Distinct Alerts</td>
</tr>
<tr>
<td>43.</td>
<td>9 Honeypots - Mean Packet Capture Percentage</td>
</tr>
<tr>
<td>44.</td>
<td>Proposed HoneyHive GUI</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Comparison of Honeypot Frameworks</td>
<td>42</td>
</tr>
<tr>
<td>2.</td>
<td>Factors and Levels</td>
<td>73</td>
</tr>
<tr>
<td>3.</td>
<td>Overview - Mean Alerts by Level</td>
<td>85</td>
</tr>
<tr>
<td>4.</td>
<td>Anderson-Darling Test - Number of Alerts</td>
<td>87</td>
</tr>
<tr>
<td>5.</td>
<td>Permutation Test - Number of Alerts</td>
<td>88</td>
</tr>
<tr>
<td>6.</td>
<td>Overview - Mean Number of Distinct Alerts by Level</td>
<td>90</td>
</tr>
<tr>
<td>7.</td>
<td>Anderson-Darling Test - Number of Distinct Alerts</td>
<td>92</td>
</tr>
<tr>
<td>8.</td>
<td>Permutation Test - Number of Distinct Alerts</td>
<td>93</td>
</tr>
<tr>
<td>9.</td>
<td>Percentage of Scanned Devices Monitored in Test Network</td>
<td>94</td>
</tr>
<tr>
<td>10.</td>
<td>Overview - Mean Packet Capture Percentage</td>
<td>96</td>
</tr>
<tr>
<td>11.</td>
<td>Anderson-Darling Test - Packet Capture Percentage</td>
<td>99</td>
</tr>
<tr>
<td>12.</td>
<td>Permutation Test - Packet Capture Percentage</td>
<td>100</td>
</tr>
<tr>
<td>13.</td>
<td>Experiment Results</td>
<td>208</td>
</tr>
</tbody>
</table>
List of Acronyms

AFNet  Air Force Network
BLOB   Binary Large Object
BPF    Berkeley Packet Filter
C2     Command and Control
CA     Certificate Authority
CEO    Chief Executive Officer
CIKR   Critical Infrastructure and Key Resources
CLI    Command Line Interface
CPU    Central Processing Unit
CSS    Cascading Style Sheets
CSV    Comma-Separated Values
CUT    Component Under Test
CWMP   CPE WAN Management Protocol
DB     Database
DDOS   Distributed Denial of Service
DHS    Department of Homeland Security
DMZ    Demilitarized Zone
DoD    Department of Defense
DODIN  DoD Information Network
DOM    Document Object Model
DOS    Denial of Service
GPS    Global Positioning System
GUI    Graphical User Interface
HTML   HyperText Markup Language
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>ICS</td>
<td>Industrial Control System</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPC</td>
<td>Inter-Process Communication</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>LCS</td>
<td>Longest Common Substring</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MQTT</td>
<td>Message Queue Telemetry Transport</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum Transmission Unit</td>
</tr>
<tr>
<td>NIDS</td>
<td>Network Intrusion Detection System</td>
</tr>
<tr>
<td>NPM</td>
<td>Node.js Package Manager</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio-Frequency Identification</td>
</tr>
<tr>
<td>SCADA</td>
<td>Supervisory Control and Data Acquisition</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SPAN</td>
<td>Switch Port Analyzer</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>TTPs</td>
<td>Tactics, Techniques, and Procedures</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>UWB</td>
<td>Ultra-Wideband</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>VNC</td>
<td>Virtual Network Computing</td>
</tr>
<tr>
<td>VPN</td>
<td>Virtual Private Network</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>Wireless Fidelity</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless LAN</td>
</tr>
<tr>
<td>WPAN</td>
<td>Wireless Personal Network</td>
</tr>
<tr>
<td>XMPP</td>
<td>Extensible Messaging and Presence Protocol</td>
</tr>
</tbody>
</table>
I. Introduction

1. Background

With the ever increasing number of Internet-connected devices, the importance of cyber security similarly increases. Exploding over the past decade, the number of Internet of Things (IoT) devices connected to the Internet jumped from 3.8 billion in 2015 to 17.8 billion in 2018 [1]. A major concern with many IoT devices is that they contain vulnerabilities that are often left unpatched [2][3]. To make matters worse, many of these IoT devices lack modern security measures found on traditional computing devices, due their inherent hardware limitations and vendors focusing on functionality and time to market over security [2]. While an insecure IoT device connected to a consumer’s probably-already insecure home network does not cause much worry, insecure IoT devices connected to previously-secure networks do. An attacker now has a vector into a previously locked down network and can use the device as a pivot to gain access into the internal network [4]. If these devices were to become connected to Critical Infrastructure and Key Resources (CIKR) networks, the results could be catastrophic.

Honeypots are devices not part of routine network usage that are meant to alert of an attacker’s presence, capture tools, and record Tactics, Techniques, and Procedures (TTPs) [5]. Honeypots come in varying levels of sophistication and are available in
multitudes of frameworks. Honeyd is one such framework that is capable of rapidly creating low-interaction honeypots by simulating the network stack. Lukas Stafira used Honeyd to develop three convincing web-based IoT honeypots which are used in this research. Stafira created IoT honeypots for the TITAThink camera, Proliphix thermostat, and an ezOutlet2 power outlet [6].

Network Intrusion Detection Systems (NIDSs) are devices that analyze network traffic and create alerts if they see traffic of malicious nature and or anomalous traffic. Intrusion detection is split into two categories, signature matching and anomaly detection. Signature matching uses known patterns of malicious traffic and creates alerts upon seeing the pattern. Anomaly detection on the other hand uses baselining and heuristics to create alerts when network traffic deviates from the network baseline. Signature matching is faster and easier to setup than anomaly detection but only alerts on traffic matching installed signatures. This means that signatures must be kept up to date and any previously unknown exploit (zero-day) will not generate an alert. Anomaly detection can detect zero-day exploits but requires much more setup and can create many false positives, or false negatives, if the heuristics are not fine-tuned. Modern NIDS often are a hybrid of the two detection techniques.

1.2 Motivation

Even networks with security measures in place are not immune to compromise; an example is the cyber attack against Ukranie in 2016 where attackers successfully gained internal network access through a phishing campaign. After initial access, attackers then conducted internal network scans and credential harvesting over a period of several months. Using gathered credentials, attackers gained access to Supervisory Control and Data Acquisition (SCADA) networks and took approximately 30 power stations offline, sending the country and more than 230,000 residents into darkness for
several hours [7]. If the network had contained convincing SCADA honeypots then it is possible that network administrators would have detected the attackers’ presence and been able to respond in time before the real SCADA systems were taken offline. While honeypots do not guarantee network security nor are they the solution to securing every network, their use is another viable tool for network defense.

Due to IoT devices’ lack of sophisticated hardware and vendor support for security updates, other methods must be implemented to secure the network. Because so many IoT devices remain unpatched, unmonitored, and left on, they have become a tantalizing target for attackers to gain network access or add another device to their botnet [8]. Due to IoT device popularity with attackers HoneyHive was developed. HoneyHive is a framework that uses distributed IoT honeypots as NIDS sensors that beacon back to a centralized Command and Control (C2) server. This research uses the IoT honeypots developed by Stafira, but HoneyHive is flexible enough to support any device capable of running the Python 2.7 HoneyB Agent script. Providing security for all IoT devices with their heterogeneous nature is a monumental task. HoneyHive instead offers another method for network intrusion detection using the lure of vulnerable IoT devices as distributed honeypot intrusion detection sensors.

Because traditional NIDSs typically only monitor Switch Port Analyzer (SPAN) traffic from the switch they are located on, they can miss attacks located on other parts of the network. Typical placement of a NIDS is just inside a network’s external firewall in the Demilitarized Zone (DMZ) [9]. If an attacker manages to infiltrate an internal network without tripping the NIDS, then internal attacks and or scans can be performed without raising an alert, as was the case with the Ukraine power network. The HoneyHive framework addresses this shortcoming of the traditional NIDS construct by using distributed IoT Honeypot NIDS sensors.
1.3 Research Goals

The goal of this research is to first develop the HoneyHive framework and then test its effectiveness in network intrusion detection compared to that of a traditional NIDS.

The hypotheses for this research are:

1. The HoneyHive framework operates correctly by not alerting on routine network traffic and alerting on non-routine network traffic.

2. The HoneyHive framework detects intrusions that traditional NIDSs cannot through the use of distributed IoT honeypot sensors and packet capture aggregation.

1.4 Approach

In order to determine HoneyHive’s effectiveness at network intrusion detection, several steps must be taken. These include the development of the HoneyHive framework, setting up the simulated network for experimentation, and then designing and running the experiment.

1.4.1 HoneyHive Framework

To develop the HoneyHive framework, Honeyd and Stafira’s IoT honeypots are first setup. Then the individual components of the framework are developed, including the C2 server, transfer server, Snort log parser, Database (DB), and HoneyB Agent script. Snort is also integrated into the framework for increased signature matching. The HoneyHive framework is explained in more depth in Chapter 3.
1.4.2 Simulated Network

A simulated network also needs to be setup in order to run the experiment. The network is composed of IoT devices, Stafira’s honeypots (duplicated several times), Windows 10 devices running Ubuntu Virtual Machines (VMs), Ubuntu VMs running the Honeyd honeypots and HoneyB Agent script, the HoneyHive C2 server, Suricata, an Ubuntu attacker machine, and networking devices. The network layout is described in Chapter 3, and the actual devices used on it and in the experiment are described in Chapter 4.

1.4.3 Experiment

After development, HoneyHive’s effectiveness at network intrusion detection is tested in a simulation where an attacker has gained access to the internal network, has narrowed down their list of targets through previous reconnaissance, and now is performing internal nmap network scans against the specific Internet Protocol (IP) addresses before launching exploits against them. The attacker launching exploits on scanned devices is not tested in this experiment. The exploitation and propagation stage would hopefully be prevented by network administrators through the use of alerts from the HoneyHive framework. The tests in this experiment involve four types of scans and four levels of active honeypots. The scan types include No Scan (Control Group), TCP Connect scan, Aggressive scan, and NIDS Avoidance scan. The levels for honeypots are 0, 3, 6, and 9 honeypots. Each of these are run in different combinations with one another for a full factorial experiment resulting in 16 different combinations. Each test is performed 30 times for a total of 480 runs. Because of the timing and coordination required to run the experiment, gather results, and reset devices to their initial state after each run, the runExperiment.py script automates this process. This script is discussed further in Chapter 4 and is found in Appendix
1.5 Assumptions and Limitations

1.5.1 Assumptions

The following assumptions are made in this research:

1. Routine network traffic on the simulated network does not contain any traffic a NIDS would treat as malicious.

2. Given the same set rules, NIDS create the same number of distinct alerts and the same number of total alerts when analyzing an identical sample of network traffic.

1.5.2 Limitations

1.5.2.1 HoneyHive

Several limitations currently exist in the HoneyHive framework. The HoneyB Agent script is written in Python 2.7 which is near the end of its life. Additionally, HoneyHive relies on a NIDS (Snort) to perform signature matching instead of being self-contained and possessing its own sophisticated intrusion detection system.

1.5.2.2 Honeypots

While the honeypots in this experiment are useful for testing hypotheses, implementing modern and more sophisticated honeypots would improve the HoneyHive framework. Honeyd is outdated and no longer regularly maintained [10][11]. Also, Stafira’s honeypots are low-interaction and are only convincing with web traffic.
1.6 Research Contributions

The HoneyHive framework offers increased network intrusion detection to all networks its deployed to. It can be used for integration in CIKR-based networks since IoT devices share some similarities with Industrial Control System (ICS). In addition, government organizations or commercial companies that work in cyber security could integrate HoneyHive into their existing network security architecture. The impact of this framework is a cross-platform, standalone, NIDS / Network Monitoring solution capable of improving the rate at which network intrusions are detected. While HoneyHive may not be the solution for every network, it is a viable tool for increasing network security through intrusion detection.

1.7 Thesis Overview

Chapter 2 provides background information and related research on the state of IoT devices, IoT and Computer Network Security, NIDS and Network Monitoring, and honeypots and honeytokens. It also provides details about software and programming languages used in this thesis. Chapter 3 describes the HoneyHive framework design and components in depth and explains the rationale behind design decisions. Chapter 4 describes the methodology for running the experiment and the research questions posed for this thesis. The methodology includes all parameters, factors, metrics, and a step-by-step procedure to replicate the experiment. Chapter 5 presents the experiment results and provides analysis. Finally, Chapter 6 provides a summary and conclusion for this thesis as well as future work to improve the HoneyHive framework, and hopefully, IoT and Computer Network Security.
II. Background and Related Research

2.1 Overview

This chapter provides background information on IoT, IoT and Computer Network Security, NIDS, and Network Monitoring, honeypots and honeytokens in Section 2.2. It also covers Honeyd 1.5c, Cyber Deception, and programming languages and tools used in this research. Section 2.3 explores related research and emerging technologies in the field of IoT and honeypots.

2.2 Background

2.2.1 Internet of Things (IoT)

The term IoT covers a myriad of devices and appliances with capabilities to sense the world around them, process information, and share this information with other devices on an internal network or the Internet at large [12]. Simple sensors and appliances now have the computing power for making intelligent decisions, as well as communication abilities for sharing perceived data and being remotely interacted with [13]. Suo et al. break IoT device functionality into four layers: the application layer, support layer, network layer, and perceptual layer. The application layer is the actual service displayed to the user such as a web page, application, or screen. The support layer acts as the intermediary between the application and network layers and involves cloud computing to bring increased performance. The network layer deals with transmitting data between devices through numerous different communication protocols. Finally, the perceptual layer is responsible for collecting data in the physical world and converting it to digital data through the use of sensors, cameras, Radio-Frequency Identification (RFID), Global Positioning System (GPS), transducers, thermostats, etc. [12].
The majority of IoT devices communicate on a Wireless Personal Network (WPAN) with a 10 meter range using one of several different Institute of Electrical and Electronics Engineers (IEEE) protocols. These protocols mainly include Bluetooth (IEEE 802.15.1), Ultra-Wideband (UWB) (IEEE 802.15.3), and Zigbee (IEEE 802.15.4) [13][14][15][16][17]. Some devices utilize Wireless Fidelity (Wi-Fi) (IEEE 802.11) instead for communication over a Wireless LAN (WLAN) with a range up to 100 meters. However, the devices used in this research, and that would be found as honeypots, communicate over Ethernet (IEEE 802.3), which is the focus of this research.

The added functionality of smart devices makes them very appealing and has caused the IoT market to explode over the past decade. As shown in Figure 1, the total number of devices connected to the Internet in 2018 was 17.8 billion, 7 billion of which were IoT devices. By 2025, the total number of devices connected to the Internet is expected to grow to 34.2 billion with IoT devices comprising 21.5 billion of the total devices. The IoT market is anticipated to grow to reach $1.6 trillion by 2025, making it a lucrative and competitive market [1].

With such a competitive market, vendors are scrambling to be the first to release the latest and greatest product, often cutting corners in areas like security to reduce cost and production time. IoT devices are often slapped together with inexpensive, outdated, and insecure third party components that are no longer supported or patchable [2]. Insecure IoT devices are an alarming problem in network security for consumers, corporations, the Department of Homeland Security (DHS), and the Department of Defense (DoD).

2.2.2 IoT and Computer Network Security

IoT creates new possibilities for technologies never before imagined but also opens up new vulnerabilities and attack vectors for malicious hackers [4]. These new attack
vectors arise from a lack of security in devices. Unfortunately, some vendors in this market are not primarily concerned with the security of their devices. Their main focus is to rapidly develop innovative and easy to use technology before competitors and turn a profit. This mindset often leaves many IoT devices ripe for exploitation.

Many IoT devices are riddled with vulnerabilities due to vendors focusing on cheap solutions and rapid development in a competitive market. HP performed a study and found that 70 percent of IoT devices contain vulnerabilities. When researching 10 of the most popular IoT devices, they found an average of 25 security vulnerabilities per device, with over 250 vulnerabilities in total [3]. Outdated software is loaded on devices that are then often never updated or very cumbersome to do so for the average user. This results in millions of IoT devices with known unpatched vulnerabilities connected to the Internet, just waiting for attackers to exploit them. Attackers can quickly discover devices with known vulnerabilities using websites such as Shodan.
While being able to lock a house, switch on or off lights, or adjust a thermostat remotely can be desirable, the inclusion of these IoT devices opens up significant vulnerabilities in networks. A once-secure network can now be accessed by exploiting a vulnerable IoT device and using it as a pivot into the otherwise unreachable network. Because these devices are not intended to be accessed by just anyone, unlike web servers, they are not placed inside DMZs, but are instead placed deeper within the network. Attackers are still able to reach these vulnerable IoT devices if they first compromise a DMZ or a different internal device and utilize that device as a pivot to the IoT device. Traffic is already allowed to DMZ devices, but a misconfigured router or router with port forwarding can allow internal devices to be compromised.

The threat of IoT devices being hacked is not just theoretical; in 2014, smart meters were hacked allowing attackers to spoof messages between nodes. With spoofed messages attackers could avoid paying their monthly utility bill or shut down energy from the utility company altogether, without the use of any kinetic effects. The attacker’s ability to shut down energy demonstrates the threat of cyber to CIKR networks; they are susceptible and can be disabled as well [21].

Many IoT devices have been found that allow logins with empty, default, or weak passwords. IoT devices are becoming increasingly common targets for use in botnets. While IoT devices may have limited computational power, their sheer number and ease of exploitation have made them an enticing target for attackers. In addition, these devices are not often updated and have limited user interaction allowing attackers to go unnoticed on a network for a prolonged period of time [22]. Take the Mirai (Japanese for “the future”) worm for example, hundreds of thousands of IoT devices have been compromised and assimilated as part of botnets since its release in 2016. Mutations of the Mirai worm are even prevalent today because of the lack of
security implemented in IoT devices [8]. These botnets, consisting of up to 400,000 devices, are available for purchase and have been used to execute a Distributed Denial of Service (DDOS) attack on a number of web servers successfully [23].

Because IoT devices lack sophisticated hardware and are so diverse, traditional methods for securing them like installing antivirus software or automatic updates are not possible typically. Kolias et al. argue that the vendor is responsible for implementing automatic updates and better security in device [8]. IoT devices rarely receive updates to fix vulnerabilities, and on the off chance they do, there is even a smaller percentage of users that take the time to manually install the updates [2]. The average user plugs the IoT device into their network without changing default passwords, and never manually checks or installs updates [20].

2.2.3 Network Intrusion Detection System (NIDS)

One common device to increase computer network security is a NIDS. A NIDS can be deployed on networks to detect malicious traffic and intrusions. They can be placed either inline, which can affect network latency as all traffic now passes through the NIDS and is then forwarded to its destination, or mirrored where copies of all traffic are sent to the NIDS as well as the original destination. NIDS can use multiple techniques for intrusion detection which include signature / pattern matching, and or baselining / anomaly detection. Signature-based detection searches network traffic for patterns defined in rule-sets and creates an alert if there is a match. Baselining involves taking a snapshot of normal traffic on a network, and then using heuristics; any behavior that is abnormal (an anomaly), generates an alert. Modern NIDS employ a combination of the techniques as they both have advantages and disadvantages. Signature-based detection is great at alerting on known exploits, but is unable to alert on zero-day exploits, which results in false negatives. Baselining on the
other hand requires creating a network traffic standard that if deviated from causes an alert. If the network changes or routine traffic changes, then a new baseline has to be performed. Anomaly detection can create numerous false alerts (false positives) and require significant setup time for learning the network baseline. However, anomaly detection is capable of detecting previously unknown vulnerabilities.

This research proposes the use of honeypots as NIDS. Using honeypots as a NIDS is not a novel idea. Spitzner argued that they make more effective NIDS than traditional ones since they reduce false positives because any traffic sent to them is suspicious [24].

### 2.2.4 Networking Monitoring

Network monitoring software works closely with intrusion detection sensors but is focused on overall network traffic patterns and determining whether or not devices are reachable. While NIDS inspect the content of traffic, network monitors record the volume and types of traffic on the network. Network monitoring quickly helps network operators identify overloaded network links and devices. Network monitoring is useful for bringing devices back online and can detect spikes in traffic, indicative of a Denial of Service (DOS) attack. Furthermore, if a device goes down, it can signal that an attacker launched an exploit that resulted in a crashed service. NIDS and network monitoring used together provide a better picture of network health while still closely investigating traffic for malintent. The HoneyHive framework developed in this research provides both network monitoring and network intrusion detection for networks it is deployed to.
2.2.5 Honeypots

One way to mitigate or detect exploited devices is the use of honeypots. Honeypots are used to increase the security of computer networks by emulating real devices attackers might be interested in compromising. A honeypot being interacted with can be one of the first signs of compromise in a network or of an impending attack, and can therefore act as a NIDS. By using honeypots, previously unknown vulnerabilities (zero-day vulnerabilities) may be discovered when an adversary targets and gains access to the device. In addition, honeypots leave known vulnerabilities unpatched so that TTPs of an adversary can be learned and or later used to fingerprint an actor that employs the specific TTPs.

Honeypots come in all kinds of shapes, sizes, and implementations. They range from simple scripts, virtual devices, to physical devices and support low to high-interaction.

**Low-Interaction Honeypots**  Low-Interaction honeypots can simulate common network services and the network stack. However, upon receiving a known exploit, the attacker does not receive full control of the device because the command terminal spawned is simulated. This also means that zero-day exploits are not captured since they are outside what the honeypot knows how to react to. Because the attacker cannot gain full control of the honeypot, this does make them safer for deployment in a network, but at the expense of being easier to detect as a honeypot by attackers.

**High-Interaction Honeypots**  High-Interaction honeypots in contrast do not emulate network services or the network stack but do allow the attacker to gain full control of the device. This not only allows for a more believable honeypot, but also supports gathering more information about the attack such as zero-day exploits,
tools and TTPs used by the hacker. Although there are many benefits to high-interaction honeypots, they also have disadvantages. While the honeypot looks more convincing to the attacker because of the full control allowed, it now presents an increased security risk to the network. In addition, high-interaction honeypots are costly to develop in both time and resources; they require more maintenance and oversight than low-interaction honeypots [24][25][26][27][28].

2.2.6 Honeytokens

Cymmetria breaks honeytokens into several sub categories: breadcrumbs, “beacons”, and tokens [29]. Breadcrumbs are data left intentionally for a hacker to find and use to allow them to move throughout the network. However, by using breadcrumbs an attacker only moves through a controlled path of devices monitored by network defenders. All the while, an attacker’s TTPs, which include commands, tools, and exploits are being recorded. Cymmetria’s “beacons” create alerts whenever they are interacted with. They are not part of routine usage (like honeypots) by the organization so any interaction is considered malicious and can even help to identify insider threats. Examples of “beacons” include decoy shares, documents with embedded macros, and websites that all beacon back to a C2 server when touched. The “beacons” defined by Cymmetria are essentially various types of intrusion detection sensors. Cymmetria’s last category, tokens, are Honeydocs (fake documents) that act as a beacon to alert that a file was exfiltrated out of the network. The main difference between beacons and tokens, as defined by Cymmetria, is that “beacons” reside on the organization’s internal network and tokens are meant to detect data leaving the network [30].
### 2.2.7 Honeyd 1.5c

One common framework to create virtual honeypots is Honeyd. The version of Honeyd used and described is 1.5c and was last updated in May of 2007. Honeyd 1.6d is available on Github with a last commit of December 2013, but as noted by Stafira, contains program stability issues [6].

Honeyd simulates the network stack to allow one physical device to act as numerous honeypots. All traffic for the honeypots is sent to Honeyd which makes it look like the devices are running independently on separate IPs. Honeypots can also be customized by using Nmap DB files to deceive scanning and fingerprinting software. The Nmap DB file defines how different Operating Systems (OSs) and their respective versions respond to messages as well as ports and services that are running by default [25][31]. While not identical, the Nmap DB file can be used to closely match network fingerprints of IoT devices. One deficiency Stafira noted about Honeyd was the outdated Nmap DB files [6].

Within the Honeyd configuration file, low-interaction honeypots can be quickly created. Each honeypot can be assigned a personality defined by the Nmap DB file, customize ports to open, filter, and close, and run custom shell scripts on open ports. Running customized scripts is one of the selling features of Honeyd; with sophisticated enough scripts, entire services can be mimicked. In theory, creating scripts to match every service would yield a convincing and very interactive honeypot, but the operating system itself, as well as Inter-Process Communication (IPC) would be painstakingly time consuming and better alternatives such as VMs exist for high-interaction honeypots. Honeyd is designed and better suited for quickly creating numerous honeypots and simulating a handful of services to provide a low-interaction honeypot framework.

The IP and Media Access Control (MAC) addresses of the honeypot are also
configurable, but the IP must be on the same network as Honeyd. MAC addresses can be used to identify the type of device and manufacturer, so allowing customization leads to more convincing honeypots. However, with these customization options, it is imperative that both the IP and MAC addresses be unique on a network in order to prevent collisions [31].

The Honeyd documentation states that with the flag “l” it logs packets and connections to a specified file. However, this logging option contains only time stamps, IP addresses, ports, protocols, and transmission byte counts; the actual packet contents captured by Honeyd are not included. While advanced methods did exist to receive the contents of the packet capture from Honeyd, none are viable now due to the out-of-date libraries and compilation errors. Furthermore, Provos stated that he expected a NIDS or other scripts to be run in tandem with Honeyd [31]. Implementing a full packet capture whose contents are accessible is one of the deficiencies this research plans to address in the Honeyd framework.

Honeyd can be compiled with internal Python services that allow interacting with Honeyd through either honeydctl (Honeyd Control) or Python scripts while it is still running. This allows for the creation of dynamic honeypots. Honeydctl connects to Honeyd and presents the user with a console for issuing commands. These commands allow listing running honeypots, modifying, or deleting them. One noteworthy command is “!” which allows sending Python commands directly to Honeyd. By simply importing the honeyd module in honeydctl (after issuing !) or a Python script, the user now has access to all the data received and transmitted by Honeyd. Having access to this data would allow creating packet captures and signatures for received data. Unfortunately, Honeyd has compilation issues because of out-of-date library dependencies and the version of Honeyd 1.5c that can be installed from the Ubuntu packages list is not built with internal Python services. Therefore, this research per-
forms its own packet capture for signature creation and forensic analysis [31].

**Honeydstats and Honeyview** Honeydstats and Honeyview are plugins that allow analyzing the log that Honeyd generates from received traffic. Honeydstats is a text-based representation of packet level data received (very similar to Honeyd’s ‘l’ option), while Honeyview is a web-based GUI representation. Both Honeydstats and Honeyview focus on the OS versions, destination ports, country codes, and IP addresses from attackers [31]. Although statistics can be aggregated from across the network, they both still rely on log files. The information they provide is not in depth enough for forensic analysis nor fast enough for today’s cyber attacks. This research hopes to provide a framework with real time alerts while capturing detailed evidence for forensic analysis.

### 2.2.8 Cyber Deception

In their 2017 Cyberthreat Defense Report, the CyberEdge Group recommended that cyber deception technology should include coverage for IoT devices [32]. While not IoT specific, Cymmetria is leading the way on Deception Campaigns. They define cyber deception as “baiting, studying, investigating, fingerprinting, and/or smoking out” attackers. Through the use of cyber deception, organizations can prevent attackers from moving freely throughout their network and impose an increased cost to attack the defended network. Cymmetria explains that cyber deception is more than just implementing everything honey (honeypots, honeynets, and honeytokens), as traditional honey technology is difficult to integrate into a network, expensive to develop and maintain, and easy for attackers to detect. It is the creation, management, and monitoring of these false devices in order to manipulate attackers through a predefined and monitored path of the network (or as Cymmetria defines, “orchestration and virtualization”). Even if the attacker realizes there are fake devices and
documents, the speed of their attack and propagation is hindered because now they have to spend extra time verifying if a target is real or a honeypot. All the while, the attacker’s tools and exploits are at risk of being captured. If they are captured, then a signature or patch can be created and propagated throughout the networks, or even worse for the attacker, reported to vendors and antiviruses and distributed throughout the world. This renders the attacker’s tools worthless and requires them to spend more time and money to create new ones. While traditional honeypots could still pose this threat to attackers, alerts and all collected information on the attacker are not available in real time [30].

2.2.9 Programming and Languages

JavaScript  JavaScript is one of the staples for web development today. It is an interpreted language and allows interacting and modifying the HyperText Markup Language (HTML) Document Object Model (DOM), which enables creating dynamic web pages without requiring a user to reload it. Through the use of Node.js and Electron, applications such as HoneyHive can be created with dynamic Graphical User Interfaces (GUIs) [33].

Node.js  Node.js is a standalone runtime environment for JavaScript built and maintained by Google. It utilizes Chrome’s V8 engine and possess the functionality to interact with the OS that JavaScript normally does not have since it is sandboxed in the browser. This extra functionality ranges from interacting with local files to networking modules for creating a full fledged server. The OS can be queried for information and concurrency can be implemented through the use of child processes. These are just a handful of built in modules, but using the Node.js Package Manager (NPM), modules can quickly be downloaded and installed for use in an application. Node.js is so versatile because it runs exactly the same across all platforms
Electron

Electron is a module for Node.js that allows creating cross-platform GUIs. It utilizes HTML, Cascading Style Sheets (CSS), and JavaScript to render the GUI and is essentially the same as coding a standalone web page for the application. This makes creating a GUI that works on any platform relatively fast and easy because of all the CSS frameworks available. Many applications have already been written in Electron, such as Atom, Visual Studio Code, Discord, and Slack, to name a few. Electron applications can even be bundled into executable files for ease of distribution using the Electron Packager module [35].

Python

Python is an interpreted language and currently supports two versions 2.7.X and 3.7.X [36]. The Python code written in this research uses 2.7.X because of the greater community support for Python modules available and because Lukas Stafira, whose work is built upon in this study, created his IoT honeypots with version 2.7.X. Modules can be quickly installed in Python by using its package manager, pip [37].

2.2.10 Tools

This section covers the essential tools used in this research for testing and results. These tools include Nmap, packet capturing software (Wireshark and TCPDump) and VMware Workstation. Docker is also discussed because several related researchers utilize it, and future work on this research references it.

Nmap

Nmap is an open source port scanning tool used by attackers and security professionals alike for reconnaissance and vulnerability analysis on networks and their devices [38]. It provides information on a device’s ports, the services running
on the ports, and the suspected OS. All gathered information from ports, running services, and the Time to Live (TTL) in packet responses are compared against the Nmap DB to make a best guess about the target’s OS. Nmap scans are customizable in the type of host discovery performed, to the way it scans ports for services, all the way to firewall and NIDS evasion. Using firewall and evasion flags, an attacker is able to spoof their MAC, port, checksum, TTL, and even modify their Maximum Transmission Unit (MTU), which results in smaller fragmented packets. The timing and performance flags allow adjusting timeouts and data transmission for faster or slower scans. If used in combination, these flags can result in scanning a network over a long period of time, without ever alerting a firewall or NIDS [38]. Varying the scan types and parameters of the scan and testing how effectively and quickly a scan is detected is one measure of performance for the HoneyHive framework.

Wireshark and TCPDump Packet capturing software is used to sniff traffic on a network and inspect it in further detail later. Both Wireshark [39] and TCPDump [40] utilize the libpcap library to capture network traffic, but Wireshark is GUI-based, while TCPDump is Command Line Interface (CLI)-based. Filters can be used to eliminate unwanted traffic either before or after capture to narrow in on specific hosts or protocols. Although both Wireshark and TCPDump allow quick filtering of data by specifying source / destination hosts and ports, TCPDump allows filtering on specific bytes in frames, packets, datagrams, and applications using the Berkeley Packet Filter (BPF) syntax [41]. Although Wireshark has a command line equivalent (tshark), TCPDump is used to collect traffic in finer granularity and then analyzed in Wireshark for this research [39][40][42].

VMware Workstation VMware Workstation is a hypervisor software solution that allows creating and running VMs [43]. Virtual Machines contain a separate
emulated CPU, OS, memory, and disk space. Hardware from the host system is shared between it and all running VMs and the amount of resources allocated to a VM is highly configurable, see Figure 2. Users can select the amount of memory and disk space a VM has access to, as well as the number of processor cores and peripherals it can use. Not only does VMware allow running multiple OSs on a single computer without having to reboot, as is the case with multi-booting, but it also provides a more secure, sandboxed environment to run applications. If an application becomes compromised in a VM, only that VM’s OS is affected; other VMs as well as the host OS are not affected, see Figure 3. An attacker would have to break out of the VM OS, and VMware hypervisor to get at the host OS, a lengthier and more complicated process than escaping docker containers [43] [44].
Figure 2. VMware Workstation Hardware Settings
**Docker**  Docker is a program that allows running container images. A container image is packaged software with all the dependencies included, i.e., libraries, code, and other tools needed for execution. Because the containerized software includes all dependencies, execution is the same across different infrastructures. This greatly increases the portability and stability of software across different devices. At runtime, container images become containers, which results in isolating the running software from other processes and giving it only user-level privileges. This improves the security of the applications because now if the application is compromised by an attacker, the attacker is limited to access of that container only, essentially a sandbox. Whereas VMs run a guest operating system on top of the hypervisor to sandbox each application, the docker engine runs right on the host operating system. With the elimination of the guest operating system layer from Figure 3, containers utilize less resources (memory, disk, CPU), which allows running more containerized applications than virtual machines, as shown in Figure 4 [44].

Docker claims to increase application and device security, however, Sever and Kišasondi demonstrate that if container images are misconfigured then attackers can compromise other containers, possibly escape the container and compromise the host operating system [45]. While there are configurations and security measures that can be put into place to prevent this, the prepared Dockerfiles and many GitHub images do not use them [45]. Any system that is improperly configured becomes susceptible to exploitation and therefore users of Docker should not assume their systems are secure just because they are running containers. In fact, there are known exploits to escape docker containers as listed in the Exploit Database website [46]. Users should properly configure their containers with security in mind, and then lock down the security of the host operating system as well.
Figure 3. Virtual Machine Structure [44]

Figure 4. Container Structure [44]
2.3 Related Research

Many different frameworks for building honeypots have been developed and are explored in this section. They range from generic honeypots, ICS / SCADA and IoT honeypots. There are numerous honeypots that serve one purpose such as a specific exploit or service, but the focus of this section is honeypot frameworks that allow the creation of many convincing IoT honeypots.

2.3.1 Conpot

Conpot is developed and maintained by the Honeynet Project and is used to create ICS honeypots. Because IoT devices are used to control things like thermostats, electrical components, and appliances, they bear an ever-increasing resemblance to ICS. Conpot provides a suite of protocols found on ICS networks and throttles their responses to mimic real system response time [47].

2.3.2 IoT Web-Based Honeypots by Lukas Stafira

Using the Honeyd framework and Python, Stafira emulated the web services for three IoT devices to create realistic and interactive web-based honeypots [6]. These devices included the TITAThink Camera, Proliphix Thermostat, and ezOutlet2 Power Outlet. In order to make the devices appear dynamic, Stafira accessed local data, such as time and weather, and used them to generate web pages when responding to HyperText Transfer Protocol (HTTP) requests. Stafira tested whether Honeyd could be used to create near duplicate honeypots that simulate the web traffic of several IoT devices. The honeypots Stafira created successfully mimicked the HTML data of web transmissions for the real devices. He also tested the Transmission Control Protocol (TCP)/IP and HTML header similarity, response time, and Nmap completion time for SYN, UDP, and FIN scans. Stafira compared his results to the physical IoT devices
using Wireshark, Nmap, and custom Python scripts. His test network configuration
is shown in Figure 5. Honeyd is shown being able to run all three IoT honeypots on
a single VM. Overall, Stafira’s results showed that it is possible to create convincing
IoT honeypots, and the honeypots he created are used in this research as convincing
IoT sensors for network intrusion detection [6].
Figure 5. Stafira’s Network Configuration [6]
2.3.3 Honeycomb by Christian Kreibich

Honeycomb is a tool for automatic signature generation from malicious network traffic captured with honeypots, specifically those part of the Honeyd framework [48]. Kreibich treats all traffic captured by honeypots as malicious because interaction with them is suspicious and not routine. The signatures generated are formatted for both the Zeek (formerly Bro) and the Snort NIDS [49] [50]. Honeycomb hooks into Honeyd and keeps track of network connections (IP and port combinations), while filtering out traffic received from being scanned, and generates signatures using the Longest Common Substring (LCS) algorithm. Using Honeycomb, Kreibich successfully generated signatures for both the Slammer Worm and the CodeRed II Worm [48].

2.3.4 Honeyd Syslog Solutions

Kiwi Syslog Server  Kloet demonstrated how using Kiwi, it is possible to filter Syslog messages generated by Honeyd. The Syslog messages were sent from the host machine running Honeyd to the machine running the Kiwi NIDS [51]. The Kiwi program then filtered Syslog messages and generated Simple Mail Transfer Protocol (SMTP) email alerts based on predefined rules, such as a connection being established to a honeypot. Kloet also mentioned remedies for false positives which include fine tuning the Kiwi alert threshold, creating a static route to null, and excluding the address that the Honeyd daemon listens on [51]. Kloet’s solution of forwarding the Honeyd generated Syslog to a program more capable of parsing and displaying the alerts in a readable format is useful for small networks. However, in larger networks, network administrators could easily be flooded by emails, whether the emails are actual alerts or false positives, and it could be difficult to piece together and visualize what is happening. While Kiwi can filter out the noise of false positives, it has no graphical overview of the network for easy real time interpretation by network
Honeycomb by Lavenya and Kaur  Honeycomb by Lavenya and Kaur is a Honeypot log management tool. It gathers all log files generated by the Honeyd framework, emails them in one file for download, and then allows importing and inspecting them in the web-based GUI [53]. Much like the Kiwi syslog server solution, it is a step towards making the Honeyd log files more manageable and collection of them automated. However, the alerts are still not conveyed to network operators fast enough and the data in the log files is not in-depth enough to capture exploits, tools, or an attacker’s TTPs [53].

2.3.5 IoTCandyJar

Ramirez et al. discuss the need for their framework since building custom IoT honeypots or buying the actual physical device to create honeypots are too costly [54]. The vast heterogeneity of IoT devices makes creating custom IoT honeypots time consuming and they often are not high-functioning enough. To combat this, their framework uses machine learning to replicate the behavior of IoT devices, dynamically creating realistic honeypots, and presenting them as convincing devices to attackers [54]. Figure 6 displays the IoTCandyJar framework. This framework consists of three dynamic honeypots that attackers interact with (left), a DB that records responses from scanned IoT devices on the Internet (middle), the IoTScanner which conducts the scanning of IoT devices on the Internet (top right), and the IoT Learner which uses heuristics and training to predict correct responses to attackers (bottom right). Requests from attackers are first sent to IoTCandyJar’s dynamic honeypots. These honeypots then query the DB for responses that could be correct. The result of the query is in-turn passed to the IoT Learner which uses heuristics to select what it believes to be the correct response. This response is finally forwarded to the attacker.
The IoTScanner constantly adds to the DB by using attacker requests to scan IoT devices on the Internet [54].

While the framework can quickly imitate any IoT device connected to the Internet, the methodology cannot precisely match responses for exploits without sending actual IoT devices the exploits, which is illegal. IoTCandyJar does use some extent of exploit filtering, but this only works for known exploits. For the known exploits, they either have to manually create the response or drop the connection altogether which means they are back to creating custom low-interaction honeypots. While for unknown exploits, their own system may very well become an attacker itself. In addition, IoT devices have specific ports open and services running on them, which would preclude a single device from responding to a Nmap scan with the exact IoT profile the attacker is targeting [54].

2.3.6 HoneyLab

HoneyLab is a distributed framework for deploying and sharing honeypots between cyber-security researchers that seeks to address the shortcomings Chin et al. described as infrastructure fragmentation, flexibility for deploying devices, and the limited IP address space [55]. HoneyLab runs honeypots in a virtualized environment for high level interaction and attack containment. The framework, shown in Figure 7, is composed of a web interface to register / login and control honeypots, the C2 node called HoneyLab Central, and sensor nodes distributed worldwide that run on Xen servers. The Xen servers deploy honeypot VMs alongside VMs with sensing software. Users can upload custom VM images which allows for maximum flexibility and custom honeypot support. Commands can be issued to honeypots through HoneyLab’s web interface but users also have the option of interacting with their honeypots through Virtual Network Computing (VNC) or a remote shell after establishing a Virtual
Private Network (VPN) connection to the network. All sensor nodes run the Honeylab daemon software to communicate with the C2 HoneyLab Central device to report alerts and receive commands [55].

Limitations with the framework include IP-only level traffic (no Ethernet), all traffic must go through the HoneyLab Central device which could become overburdened, and all outgoing connections (reverse connections) are blocked. These limitations affect the convincingness of the actual honeypots and may not fool attackers. Also, once an attacker is in the target’s internal network, they can see that the honeypot is not part of the network and all traffic is forwarded to it. Additionally, it is not apparent how propagation throughout the HoneyLab honeynet from a compromised honeypot is prevented. Finally, the research appears to be discontinued because the website was not found to be up and operational [55]. Like HoneyLab, the IoT honeypot sensors in this research beacon back to a central command and control server for real time alerts.
Figure 7. HoneyLab Design [55]
2.3.7 SIPHON

Like HoneyLab, SIPHON is a globally distributed honeynet intended to be a “Scalable, high-Interaction Physical HONeypot” framework [28]. As shown in Figure 8, the framework uses IP addresses distributed around the world from servers rented from cloud providers (Amazon, Digital Ocean, and Linode) that act as “wormholes” – interconnecting the honeynet through SSH tunnels. By using this design, certain geographically located devices that are more desirable to attackers can be simulated. As Figure 9 illustrates, the wormholes send the attacker’s traffic to SIPHON’s “forwarder” devices that change IP address and perform man-in-the-middle attacks before finally sending the traffic to actual physical IoT devices. This setup is very similar to IoT CandyJar’s method of sending traffic to physical devices, but instead of merely recording the devices’ responses for replay, SIPHON’s network owns the IoT devices and, can, therefore, allow high-interaction and record advanced attacker methodology [28].

2.3.8 HoneyIo4

HoneyIo4 by Alejandro Guerra Manzanares is a low-interaction honeypot with four Python scripts to match the expected Nmap DB scan responses for the following IoT devices: GoPro Hero3 camera, Casio QT6600 cash register, Nintendo Wii video game console, and Oki B4545 printer [23]. HoneyIo4 also includes a web-based GUI that allows starting or stopping each honeypot by simply executing the associated Python script. While HoneyIo4 successfully matched target Nmap DB OS profiles, it appears to be trying to re-invent the wheel as the Python scripts attempt to do what Honeyd does already. Honeyd allows for quickly customizing ports, responses, and specifying an OS profile from a Nmap DB for response traffic to match. Manzanares claims that Honeyd cannot match IoT OS fingerprints, but as long as the OS is in
Figure 8. SIPHON Overview \[28\]

Figure 9. Attacker’s Interaction with SIPHON \[28\]
the supplied Nmap DB file to Honeyd, the traffic can be matched.

Honeyd also has more advanced capabilities than HoneyIo4 such as running multiple honeypots at once on the same physical device, routing of network traffic, and keeping state for each honeypot [23].

2.3.9 IoTPOT and IoTBOX

IoTPOT and IoTBOX is a two-part honeypot system consisting of a Telnet service “frontend” (IoTPOT) and sandboxed “backend” (IoTBOX) [22]. IoTPOT changes its responses to match different IoT devices that an attacker is targeting based on their initial Telnet requests as illustrated in Figure 10. By using this method, IoTPOT can appear to be a vast number of different IoT devices. IoTPOT also logs all traffic which includes login attempts and credentials. Login settings can also be customized to allow authentication on the first attempt, a specific username and password combination, or authenticate only after a set number of attempts. After an attacker successfully authenticates, IoTPOT checks if the command issued has a known, stored response. If it is a known command, IoTPOT responds to the attacker directly. If the command is not known, IoTPOT forwards the command to IoTBOX, stores IoTBOX’s response so it can quickly respond to the same command in the future, and then forwards it to the attacker [22].

The design of IoTBOX is shown in Figure 11. Because some commands can be to download malware, IoTBOX is ran in a controlled environment with frequent image resets. IoTBOX uses QEMU to emulate eight different Central Processing Unit (CPU) architectures which are then run on the OS OpenWRT. The benefit of this is that malware executables are compiled to run on a specific CPU architecture, and through CPU emulation, the captured malware can be run and analyzed in depth [22].

It is not apparent how IoTPOT would know the correct banner response an at-
tacker is looking for from a specific IoT device. Also, because IoTPOT uses one IP address, an attacker or scanning tool that documents device analysis would notice this single IP responds like multiple different devices and may become suspicious of it being a honeypot.

2.3.10 Multi-Purpose IoT Honeypot

Inspired by IoTPOT, Krishnaprasad created the “Multi-Purpose IoT Honeypot” to handle four protocols commonly used by IoT devices: Secure Shell (SSH), Telnet, HTTP, and CPE WAN Management Protocol (CWMP) [56]. Multi-Purpose IoT Honeypot utilizes a “frontend” proxy that is running a Python script for each of the supported protocols. The frontend logs data about the attack and then forwards it to the corresponding service “backend” which are only two docker machines running the services. While Multi-Purpose IoT Honeypot is running common services for IoT devices, it does not tailor its responses to deceive Nmap scans performed by attackers that it is in fact an IoT device and not a honeypot. Furthermore, if an attacker does connect to a service, they realize it is not an IoT device and not connected to any real network. Krishnaprasad’s use of docker to containerize honeypot machines is a concept also used by Cymmetria’s Honeycomb framework. This technique allows for high-interaction honeypots that are easier to develop, deploy, and maintain than physical devices or traditional virtual machines [56].

2.3.11 ThingPot

Like IoTPOT and Multi-Purpose IoT Honeypot, ThingPot also has a frontend and backend design [57]. ThingPot classifies itself as a “Medium Interaction Honeypot” simulating Extensible Messaging and Presence Protocol (XMPP) and Message Queue Telemetry Transport (MQTT) and low interaction for HTTP REST traffic. Each of
Figure 10. IoTPOT Overview [22]

Figure 11. IoTBOX Overview [22]
these services is also run in a virtual environment using docker. An overview of ThingPot’s design is shown in Figure 12. The XMPP and REST nodes implement that respective protocol while the controller node logs and stores data. Using this design, ThingPot imitated a Phillips Hue smart light and had an actual attacker try and take control of it [57].

2.3.12 IoTSec

One proposed solution for securing IoT devices is the use of interceding devices called µmboxes as an intermediary between IoT devices, that dynamically configure firewall rules to allow for the specific traffic of IoT devices on the network, essentially acting as a personal firewall or blue coat proxy for the IoT devices [20]. µmboxes work together and alert the centralized IoTSec Control Platform (C2) if an intrusion or anomaly is detected. They utilize several different methods for detecting intrusions: signature matching, network baseline generation, and cross-device policies. Signature matching and network baselining are not new concepts, but cross-device policies are interesting because using the functionality of other IoT devices, a safety check can be performed. The example Yu et al. give is an IoT camera checking that a person is home before a smart oven is allowed to be issued the “on” command [20].

2.3.13 Honeycomb and MazeRunner by Cymmetria

Cymmetria is a cyber-security solutions company based out of Tel Aviv, Israel and was founded in 2014 by Gadi Evron [58]. The Chief Executive Officer (CEO), Gadi Evron, has over 15 years of experience in cyber security and was the former vice president of cyber security strategy at Kaspersky Lab. Cymmetria’s flagship product is MazeRunner which utilizes their Honeycomb framework [59].

Honeycomb by Cymmetria, not to be confused with Honeycomb by Kreibich (a
Figure 12. ThingPot Overview [57]
plugin for Honeyd), or Honeycomb by Lavenya and Kaur (a Honeyd log manager), allows for rapid and customized honeypot creation using containers and Python plugins. Honeypots are spawned off through the use of these containers [60].

MazeRunner allows users to create honeypots, add services, and change configurations all rapidly and with a GUI. MazeRunner is the container that manages all the honeypots and acts as the command and control for intrusion detection. It provides an overview of real time alerts of interaction with the honeypots. It implements packet capture, memory dump, shows what commands an attacker ran and allows downloading the tools an attacker used. The tool’s hash can then be propagated as a signature to flag on throughout the network. Custom scripts such as Stafira’s can be run with the Enterprise edition for even more customized honeypots. The backend uses their Honeycomb framework [61] [62] [63].

MazeRunner contains a component called ActiveSOC which automatically investigates an incident using rules and heuristics to determine if the incident needs further investigation by an analyst. Using ActiveSOC, false positives can be reduced and analysts can focus on investigating actual intrusions [64].

MazeRunner has been successful in catching red teams in NATO exercises [29] as well as APTs such as APT3 (pirpi - a Chinese Threat Actor) in European government networks, defense contractor networks, and several other customer’s networks [65]. Additionally, MazeRunner successfully captured the tools and TTPs of the cyberespionage group Patchwork, as Patchwork moved throughout the MazeRunner network. Patchwork, aptly named from the copy-paste code used from online forums, is a targeted attack against government agencies and has infected several thousand machines since 2015 [66].
2.3.14 Comparison of Related Frameworks

This research builds off of the IoT honeypots created by Lukas Stafira. The closest research to the HoneyHive framework is MazeRunner by Cymmetria. However, MazeRunner does not use nor allow creating custom IoT honeypots as network intrusion detection sensors. Table 1 provides a summary and comparison of all the frameworks mentioned. The specific categories compared include the honeypot level of interaction (Honeypot Level), whether or not the framework focuses on IoT honeypots (IoT), whether or not the framework implements full packet capture (PCAP), whether or not the framework is distributed (Distributed), whether or not the framework was developed with the intent for it to be used as a NIDS (NIDS), whether or not the framework reports alerts and receives commands from a C2 server (C2), and the year of the last update on the framework (Last Update). The various levels of honeypot interaction include low, medium, and high. An “X” in a category denotes the framework possess that trait. Neither Conpot nor Honeyd were made to create IoT honeypots specifically, perform full packet capture, have a distributed framework, be implemented as a NIDs, or have a Command and Control structure.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Honeypot Level</th>
<th>IoT</th>
<th>PCAP</th>
<th>Distributed</th>
<th>NIDS</th>
<th>C2</th>
<th>Last Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conpot</td>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2019</td>
</tr>
<tr>
<td>Honeyd</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2013</td>
</tr>
<tr>
<td>HoneyHive</td>
<td>Low</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>2020</td>
</tr>
<tr>
<td>HoneyIo4</td>
<td>Low</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>2017</td>
</tr>
<tr>
<td>HoneyLab</td>
<td>High</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>2009</td>
</tr>
<tr>
<td>IoTCakeJar</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2017</td>
</tr>
<tr>
<td>IoTPO &amp; IoTBOX</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>2017</td>
</tr>
<tr>
<td>MazeRunner</td>
<td>High</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>2020</td>
</tr>
<tr>
<td>Multi-Purpose IoT Honeypot</td>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2017</td>
</tr>
<tr>
<td>SIPHON</td>
<td>High</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>2017</td>
</tr>
<tr>
<td>ThingPot</td>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2017</td>
</tr>
</tbody>
</table>

Table 1. Comparison of Honeypot Frameworks adapted from [19]
2.4 Chapter Summary

This chapter discusses the exponential growth of IoT devices and the need for improved IoT and computer network security. Several methods for improving network security include the use of a NIDS, network monitoring, and honeypots. The HoneyHive framework utilizes a combination of all these aspects for increased network security. Languages and tools for the development of the HoneyHive framework are also covered in detail. Finally, related research in the field of IoT honeypots is explored to understand existing solutions, their shortcomings, and their inspiration in the development of this research.
III. Framework Design

3.1 Overview

This chapter describes the design decisions involved in the creation of the HoneyHive framework. The HoneyHive framework is comprised of IoT honeypots simulated by Honeyd, a C2 server, and a HoneyB Agent script that beacons back to the C2 server. Using the IoT honeypots as NIDS Sensors, the system can recognize scanning patterns and known IoT exploits to create real time alerts. Full packet capture of all network traffic received by the honeypots is also a feature. Additionally, Snort, Suri-cata, Nmap, VMware, and Wireshark are tools used for augmentation in the testing of this framework.

3.2 Motivation

While there are numerous NIDS and network monitoring solutions available, none meet all the objectives of this research. For one, most NIDS and network monitoring software are not tailored to honeypots or IoT. Without a convincing, high-interaction honeypot, an attacker will not launch advanced exploits (zero-days), deploy tools, or employ their TTPs [28]. Additionally, many NIDS log to files or send an email to an administrator when alerts are generated. This is neither real time nor manageable for large-scale networks. The closest solution found to this proposed research is MazeRunner by Cymmetria as discussed in Chapter 2. However, the honeypots used in MazeRunner are not IoT devices, and MazeRunner does not currently support running custom scripts for honeypot emulation.

NIDS may also only offer a limited view of the entire network they are defending. On network switches, traffic is sent only to the port of the intended recipient. Some switches contain mirror ports that send a copy of all traffic received out another port.
typically for analysis by a NIDS. In addition, for every switch another mirror port and NIDS would be needed, and then only a specific segment of the network would be analyzed individually by each NIDS. This would not provide a global picture of all network traffic across all switches for analysis. The proposed solution in this research scatters IoT honeypot sensors across the network and has them beacon back to a C2 server, where traffic is then combined for analysis. Packets are captured using Scapy on the honeypots, sent to the C2 server, and then analyzed by Snort IDS. In this experiment, Nmap is used to conduct various scans against the HoneyHive framework to test how effective it is at detecting network scans.

One question focused on in the design of the HoneyHive framework was, “What are the qualities of an effective NIDS?” Roesch describes some of the qualities of the Snort NIDS that have made it so effective, even still today. It is lightweight, cross-platform, has a small network footprint, and can be easily configured by an administrator in a short amount of time. Cross-platform means the NIDS should run on a variety of different enterprise systems and OSs. A small network footprint means a device does not generate large amounts of network traffic [67]. Furthermore, an ideal NIDS should have a low rate of false positives and virtually no false negatives (for known exploits), provide meaningful alerts in real time, and not introduce further vulnerabilities into the network.

3.3 Third-Party Software

This section describes the rationale behind software solutions used in the HoneyHive framework that were developed by other individuals. These solutions includes Honeyd, Stafira’s three IoT honeypots, Ubuntu, Snort, Suricata, Nmap, VMware, and Wireshark.
3.3.1 Honeyd 1.5c

This section describes the design decision of using Honeyd 1.5c. Because Lukas Stafira reported stability issues with Honeyd 1.6d and built IoT honeypots using 1.5c, the latter is used in this research [6]. Honeyd 1.5c is installed using apt-get install because compiling the source code generates errors even after following guides and downloading and installing older versions of library dependencies. Unfortunately, the version of Honeyd installed with command “sudo apt-get install honeyd 1.5c” was compiled without Python support. The inability to compile Honeyd prevented compiling it with the Python support as well as the Honeycomb plugin, which requires Honeyd being recompiled with its source files included. This severely impacts the level of control and interaction with the Honeyd program, prevents use of the Honeycomb automatic signature generation plugin, and influenced other HoneyHive framework design decisions.

3.3.2 Stafira’s Honeypots

The three honeypots used in the HoneyHive framework are the TITAThink Camera, Proliphix Thermostat, and ezOutlet2 Power Outlet. They simulate the web interface of real IoT devices and were created by Lukas Stafira using Honeyd 1.5c, bash, and Python 2.7. These honeypots were selected because of the high level of web-based authenticity they shared with their real counterparts and the in-depth analysis already performed on them [6].

3.3.3 Ubuntu 12

Ubuntu version 12 was used for two reasons. First, it was used by Lukas Stafira in his research with Honeyd, and second, newer versions of Ubuntu did not have Honeyd available in repositories for installation. Because of the previously mentioned
compilation problems, this ruled out the newer versions of Ubuntu.

3.3.4 Snort

Rather than create a NIDS when there are many solutions already available, Snort was selected because it is lightweight and open source. It is also widely used and has a vast community of support. Community Snort signature rules, that are periodically updated, are readily available for download [50] [67]. Snort is used in the HoneyHive framework to parse PCAPs and create alerts, but was not used as a NIDS in the DMZ. Suricata would have been used as the NIDS to perform signature matching in the HoneyHive framework instead of Snort, but there were issues with getting it to run on Windows 10.

3.3.5 Suricata

Suricata is used in the test network of this experiment but is not part of the HoneyHive framework. In the simulated network, Suricata was selected as the NIDS to monitor SPAN traffic in the DMZ. Suricata was selected as the DMZ NIDS in this experiment because it uses the same rule structure as Snort which allowed the exact same rule-set to be shared between the two. In addition, Suricata employs modern NIDS features such as multi-threading and is regularly updated [68]. In this experiment, it was important to have highly capable NIDS to compare results against, otherwise results would not be as meaningful.

3.3.6 Wireshark

Wireshark itself was used primarily for troubleshooting in pilot studies. However, several tools included in the Wireshark package were used in the actual experiment, Mergecap and Capinfos. Mergecap was used to combine multiple PCAPs into a single
PCAP. Capinfos was used to count the number of packets in the PCAP.

3.4 Programming Languages

This section describes the design decisions behind chosen languages in the development of this framework. Node.js was used for coding the C2 server whereas Python 2.7 was used for the HoneyB Agent that communicates with the C2 server. SQLite was the language selected to create the DB.

3.4.1 Node.js

Node.js was selected over other languages such as C++, Go, and Java for several reasons. Node.js uses Chrome’s V8 JavaScript engine to achieve a standalone runtime environment. Any computer with Chrome will be able to run the program with the same execution and output [34]. Another selling feature of Node.js is the Electron package, which allows rapidly creating GUIs using HTML, CSS, and JavaScript [35].

3.4.2 Python 2.7

Python 2.7 was selected instead of 3.X because the IoT honeypots selected were written in 2.7. Originally, the design was to modify the IoT honeypots to beacon back to a server. However, without Python support in Honeyd, there was very little interaction and control provided for the honeypots. Additionally, Ubuntu 12 allowed installation of Python 2.7 but for Python 3.X, only 3.2.X could be installed and the current version is 3.7.X. Python 3.2 also lacked support for modules available in 2.7 and 3.7, such as pyshark and Scapy [69].
3.4.3 SQLite

Because the DB is stored on the same file system as the C2 server, there was no need to authenticate to it over the network like in some languages such as MySQL. This allows for much faster querying and DB modification since the limiting factor is now disk read / write speed and not network latency [70].

3.5 HoneyHive Framework Design

This section describes the individual components that make up the HoneyHive framework. Shown in Figure 13, these components include the C2 server, the HoneyB Agent script, and the DB. After these components are described, the network layout for experimentation and testing is explained. The network layout includes the HoneyHive framework, Honeyd, Stafira’s honeypots, Snort, Suricata, the simulated test network, and an attacker machine.
Figure 13. HoneyHive Framework
In Figure 14, the Unified Modeling Language (UML) diagram displays the interaction of all the components that make up the HoneyHive framework. The HoneyB Agent script monitors the Honeyd honeypots for traffic, captures any traffic, alerts the C2 Server, and transfers captured traffic to the C2 server. Credentials, commands, and attempted binary uploads are all captured in network traffic. The HoneyB Agent also periodically reaches out to the C2 server to report it is still functioning, known as a heartbeat.

The C2 server receives alerts and captured traffic from the HoneyB Agent script and then relays all of this to the DB, GUI, and Snort. Although the GUI development is for future work, the overall system was designed with it in mind for ease of use by network operators. The HoneyHive framework functions fully without a GUI and displays relevant information via command line. Additionally, the C2 server provides a kill switch capability to terminate the HoneyB Agent script, Honeyd and the honeypots, and the rest of the framework.

The DB stores alert information and captured traffic. The DB also supports exporting all data, and clearing parts or all data in the DB.
Figure 14. UML Program Design
3.5.1 C2 Server, Transfer Server, and Snort Log Parser

After setting up Honeyd and Stafira’s honeypots, the C2 server was the first component developed because every other component interacts with it. The networking features were among the first to be developed. For handling higher loads and lower network latency, the server uses both multithreading and IPC. The C2 server scripts (main.js) are located in Appendix A.

Originally designed with a single server (the main server) that accepted alerts and heartbeats from honeypots, a secondary server (the transfer server) became necessary for handling the transfer of PCAPs. Without a separate transfer server, binary-encoded data would have to be converted to base64 and then back again to send over a channel expecting JavaScript Object Notation (JSON) data. This decreased the server’s robustness because additional checks would need to determine if the data was in JSON format or binary data. The conversion to base64 also created a large overhead in transferring files because it increased the number of bytes that needed to be sent in addition to the CPU cycles required for the conversion. These factors ultimately led to the separate transfer server, which listens on a different port for receiving large files.

C2 Server  Messages are sent in JSON format from the HoneyB Agent script to the C2 server with one of the following four commands: AUTHENTICATE, ALERT, HEARTBEAT, and PCAP. The AUTHENTICATE command allows Honeyd machines and their respective honeypots to be authenticated and then added to the list of IPs that are tracked by the C2 server. The ALERT command notifies the server of suspicious activity, which is then displayed to the network operator via command line and sent to the DB to store the incident. This allows for real time intrusion alerts and notification of honeypot interaction. HEARTBEAT is the pulse
function used by the HoneyB Agent script to periodically report that Honeyd and the honeypots are all functioning correctly. If the HoneyB Agent misses too many heartbeats, then an alert is generated, displayed, and stored in the DB. This heartbeat alert threshold is a user configurable parameter. Finally, the PCAP signals that the HoneyB Agent script is ready to transfer captured network traffic. To facilitate this, the transfer server is spun up and listens on the port number specified by the HoneyB Agent to download the PCAP. After a PCAP is successfully downloaded, it is stored on disk and in the DB. It is then sent to Snort for analysis, which supports the objective of full packet capture for later forensic investigation.

Additionally, several commands are added to the C2 server for the purpose of running the experiment. These include SNORT, SURICATA, REBOOT, and RESET. While packets were originally processed by Snort immediately after being received, Snort processing on packets was changed on the C2 server to wait until after the SNORT command was received from the runExperiment.py script. This made it so that Mergecap only ran once to merge all PCAPs together, and Snort only ran once on the final merged PCAP, as opposed to subsets of the merged PCAP and creating an incorrect number of alerts in the log file. Capinfos is run immediately after Mergecap in order to gather the number of packets captured. For network deployment, this entire process would be replaced with a user-defined parameter to run Mergecap and Snort on captured traffic after a set amount of time. However, in this experiment, when the SNORT command is received, the C2 server creates a merged PCAP and calls the Snort log parser on each of the individual PCAPs and on the merged PCAP. The alerts from PCAPs individually processed are stored in one log file (“alert.ids”), and the alerts from the merged PCAP are stored in a separate log file. These log files are then parsed and the results are reported back to the C2 server.

SURICATA is a command sent from the suricataConnect.py script to the C2
server and is used to record all transferred metrics from the Suricata machine - Suricata’s number of alerts (SuA), Suricata’s number of distinct alerts (SuT), and the number of packets Suricata captured (SuP). Upon receiving the REBOOT command from runExperiment.py, the C2 server reboots the Windows VM it is running on to reset everything to the initial state. When the RESET command is received from runExperiment.py, the C2 server transfers all metrics to the runExperiment.py script and then resets all metrics to zero.

**Transfer Server** When the C2 server receives the PCAP command from the HoneyB Agent, the transfer server is launched as a child process and listens on the port specified by the HoneyB Agent. The transfer server then downloads the PCAP sent to it by the HoneyB Agent. After the download is complete, the transfer server notifies the C2 server through IPC, and then the transfer server shuts down. The transfer server’s code is shown in Appendix A.

**Snort Log Parser** The Snort log parser is invoked as a child process after the C2 server receives the SNORT command. The Snort log parser executes Snort on a PCAP specified by the C2 server. After Snort finishes processing the PCAP, the alert information in the log file (“alert.ids”) is parsed and then reported back to the C2 server through IPC. The Snort log parser code is shown in Appendix A.

### 3.5.2 HoneyB Agent

Originally, each honeypot was modified to beacon back to the C2 server itself, but this caused the honeypots and Honeyd to slow down. Additionally, with this design the honeypots beaconed back to the C2 server with every request to the honeypot, instead of once for an entire session. This structure also did not afford a level of control that the HoneyB Agent script does, which is found in Appendix A.
The HoneyB Agent script is multithreaded, performing network packet capture with Scapy, periodically sending a heartbeat, alerts, and packet captures to the C2. The plugin maintains a list of honeypot IP addresses with ongoing network traffic. After a set amount of time without additional traffic to or from an IP address, it ages out; the address is removed from the connections list and its corresponding PCAP file is transferred to the C2 server. The age out time and time between heartbeats are user-defined parameters. In this experiment they are set to a 9000 seconds, a time that exceeded all but two outlier runs in this experiment. This was done so that each honeypot produced a single PCAP to make the test between Snort parsing the PCAP individually versus a merged PCAP fair.

The HoneyB Agent only sends an alert message for honeypots without tracked network activity. Examples of this are honeypots receiving traffic for the first time, or receiving traffic again after it previously aged out. There is a separate thread whose sole function is to monitor the honeypot connection list and detect and remove any IP addresses that have aged out. The traffic for that IP address is then transferred to the C2 server. If a honeypot receives traffic after the age out time, but before it is detected by the watcher thread, then this causes a transfer of the packet capture as well. The capture files are named with the format of “IP.Y-m-d.HM.pcap.” IP is the honeypot’s IP address, Y is the current year, m is the current month, d is the current day, H is the current hour, and M is the current minute.

After trial and error with other packet capturing software, Scapy was the clear winner. Scapy allowed capturing, analyzing, and running custom functions on packets received. Honeyd.py would have been the ideal solution since Honeyd already performs packet capture and would have also allowed for dynamic control of Honeyd. However, Honeyd on Ubuntu 12 was not compiled with the Python plugin, and re-compiling it generated errors.
The TCPDump and tshark programs supported capturing packets but did not allow processing and running functions on those packets. Although these programs could still have been used, the capture files would need to be manually parsed while still being modified. tshark did have a Python module called pyshark, but only the legacy version could be installed for Python 2.7, which did not work correctly and said to upgrade to the newer version.

The HoneyB Agent also had several commands added for the purpose of testing which include TRANSFER, RESET, KILL, START, STOP, and REBOOT. TRANSFER signals the HoneyB Agent to transfer and then remove all PCAP files and to remove all tracked IP addresses from the connections list. RESET only removes all the IP addresses from the connection list. KILL terminates Honeyd and then the HoneyB Agent script. START launches Honeyd and registers the honeypots by authenticating to the C2 server. STOP kills only Honeyd. Finally, upon receiving the REBOOT command, the HoneyB Agent script reboots the Ubuntu VM that Honeyd and itself are running on. Originally, this was the chosen method of returning the machine to its initial state between each experimental run. However, Stafira’s honeypots did not run correctly when Honeyd was launched on startup from rc.local.

### 3.5.3 Database Design

The DB schema is shown in Figure 15. The main table that connects all the other tables is the Alert table. Each alert has a unique ID, which serves as the primary key of this table, a timestamp, mandatory layer 3 information (source and destination IP addresses), and associated honeypot (HoneyID which is used as foreign key for the Honeypot table). Optional information includes layer 4 information (source and destination ports), and hashes for the PCAPs and memory dump files. The hashes are used as foreign keys to look up the entries in their respective tables and down-
load their Binary Large Objects (BLOBs). Each alert will only have one associated honeypot, PCAP, and memory dump. However, multiple commands, credentials, and binaries, can be found across multiple alerts and an alert can have multiple of them, creating a many-to-many relationship. To support this design, an intermediary table is used to break the table into a one-to-many relationship for both sides. AlertID is used to lookup all the associated commands, credentials, and binaries for a single alert, whereas the SHA256 is used to lookup all the alerts a specific signature is found in. Additionally, the SHA256 can be used to look up the plain-text command, credentials, and binary BLOB. Note that these tables have MD5 hashes as well to support looking up and distributing signatures generated in this format. The final table is the Honeypot table, which stores information of honeypots that have successfully authenticated. It includes a unique ID (HoneyID), their host name, MAC and IP addresses, OS, and an optional description detailing the type of honeypot and other relevant information.
Figure 15. Database Schema
3.5.4 Network Design

Figure 16 shows the overall network design with devices on separate networked hosts to ensure all traffic is not seen by all hosts, which is unrealistic. Each Ubuntu VM housing Honeyd, Stafira’s honeypots, and the HoneyB Agent script reside on separate physical machines from that of the attacker. Each Honeyd daemon runs all three of Stafira’s honeypots, with only the IP addresses and MAC addresses modified in the Honeyd configuration file and the HoneyB Agent script. Stafira’s modified Honeyd configuration file can be found in Appendix B. The IP address scheme for the Honey devices is:

1. TitaThink Camera honeypot 192.168.1.1[5-7]0
2. Prolophix Thermostat honeypot 192.168.1.1[5-7]1
3. ezOutlet2 Power Outlet honeypot 192.168.1.1[5-7]2

Because of the constant reboots from running the experiment, the C2 server was moved to a Windows 10 VM with an IP address of 192.168.1.233. The Desktop Windows 10 machine, addressed with 192.168.1.235, runs this Windows 10 VM and one of the Ubuntu VMs. The attacker machine (Desktop Ubuntu Machine), was moved from inside a VM to running natively on a physical machine and has an IP address of 192.168.1.230.
Figure 16. Simulated Test Network - Network Layout
With physical separation of test components complete, the network was designed and developed. Each set of honeypots was placed on separate switches to distribute them throughout the simulated test network. In addition, the IoT devices were spread out over all switches. The IoT devices match their honeypot counterparts IP addressing scheme in the last digit. The TitaThink Camera is 192.168.1.190, the Prolophix Thermostat is 192.168.1.191, and the ezOutlet2 Power Outlet is 192.168.1.192. A NIDS (Suricata), running on the Raspberry Pi with the IP address of 192.168.1.231, was placed in the DMZ on the Netgear ProSafe switch’s SPAN port to match their traditional network deployment location [9]. This allowed it to receive a copy of all traffic that traverses the switch. Additionally, the Raspberry Pi was retrofitted with the suricataConnect.py script in order for it receive commands and report results in the experiment. This script is found in Appendix C.

To simulate a real world attack like that of the cyber attack against Ukraine in 2016, the attacker machine was placed in the internal network on the EdgeRouter X switch [7]. The assumption is that the attacker gained internal access through a phishing attack without tripping the NIDS (Suricata) and without alerting network administrators. They are now using the compromised box as a pivot to scan the rest of the internal network. The attacker is using an encrypted channel (SSH) to communicate with the compromised box. Through preliminary scanning and reconnaissance, the attacker has narrowed down their target list to the IP addresses 192.168.1.150-192.168.192 and is now conducting further scanning prior to launching exploits. With this network design, the simulated test network is representative to that of real-world scenarios.
3.6 Summary

In this chapter, the design and justification of decisions for the HoneyHive framework are explained as well as the motivation behind its development. Design decisions explored include the third-party software used, the programming languages selected for the framework’s development, the C2 server, the HoneyB Agent script, the DB, and the simulated test network design.
IV. Research Methodology

4.1 Goals

In this research the HoneyHive framework’s effectiveness of network intrusion detection is tested and compared against that of traditional NIDS (Snort and Suricata). These tests are conducted using Stafira’s IoT honeypots, their physical device counterparts, the HoneyHive framework, Snort, and Suricata. The results from this experiment address the research hypotheses:

1. The HoneyHive framework operates correctly by not alerting on routine network traffic and alerting on non-routine network traffic.

2. The HoneyHive framework detects intrusions that traditional NIDS’s cannot through the use of distributed IoT honeypot sensors and packet capture aggregation.

4.2 Approach

In this experiment, the total number of alerts as well as the number of distinct alert types is compared across three categories in order to determine network intrusion detection effectiveness: PCAPs captured by the HoneyB Agent are parsed by Snort individually, PCAPS captured by the HoneyB Agent merged into a single PCAP and parsed by Snort, and Suricata running on a SPAN port. Additionally the number of packets captured to and from the attacking machine by the HoneyHive framework and Suricata are also compared. These metrics are further explained in Sections 4.3 and 4.4.

Different types of scans are conducted against the internal network with varying levels of active honeypots. The type of scans include no scan (control group), TCP
Connect scan to simulate an attacker attempting to connect to devices, an Aggressive scan which portrays an attacker attempting to gather as much device information as possible, and a NIDS Avoidance scan to emulate an attacker who wishes to remain undetected on the network while gathering information. Scans are conducted using Nmap, and PCAPs from the scans are created using Scapy. These PCAPs are then ultimately transferred to the C2 server and analyzed by Snort. The number of Snort alerts and types generated by the captured traffic (both individually and merged) and total packet count are then compared to the number of Suricata alerts and types and total packet count.

After initial setup of the physical devices and network, the HoneyB Agent Python script is started on each of the Ubuntu VMs. After the HoneyB Agent scripts are launched, the experiment is conducted by launching the runExperiment.py script on Desktop Ubuntu (Attacker’s machine). This script handles running all scan types and honeypot test combinations, device orchestration (starting HoneyB Agents, transferring PCAPs and metrics, analyzing PCAPs via Snort, and resetting devices / HoneyB Agents) between each run, and recording results for later analysis. The runExperiment.py is located in Appendix D.

This experiment simulates an attacker compromising an internal device, unknown to network administrators and the NIDS. The attacker then uses an encrypted channel to the compromised device and uses it as a pivot point to conduct an internal network scan. This is similar to the 2016 cyber attack against Ukraine mentioned in Chapter 1 and Chapter 3 [7].

4.3 System Boundaries

In Figure 17, the System Under Test (SUT), the HoneyHive framework, is shown compromised of the HoneyB Agent, C2 server, and Snort. The specific part under
test, the Component Under Test (CUT), is the C2 server. The C2 server aggregates all captured traffic from the HoneyB Agents, run PCAPs through Snort, and displays alerts.
Figure 17. HoneyHive Framework
4.4 Parameters, Factors, and Metrics

4.4.1 Assumptions

During this experiment, several assumptions are made which are as follows:

1. The attacker has already compromised a machine in the internal network and through an encrypted channel, is using it as a pivot to scan the internal network

2. The attacker has performed rudimentary investigation already to narrow down the IP range to those scanned in the experiment

3. The honeypots are convincing enough for the attacker to perform further investigation, i.e., scanning

4.4.2 System Parameters

This section describes all the system parameters used to conduct the experiment. This includes the computing parameters, the programs and languages with their respective versions, and the configurations Honeyd, Snort, and Suricata.

Computing Parameters The following is a list of physical devices and their hardware specifications used in this experiment:

- Desktop Windows 10 64 bit
  - ASUS ATX DDR4 LGA 1151 Motherboard Z170-E
  - Intel Core i7 6700K 4.00 GHz Unlocked Quad Core Skylake Desktop Processor, Socket LGA 1151 [BX80662I76700K]
  - 4 X Crucial Ballistix Sport LT 2400 MHz DDR4 DRAM Desktop Gaming Memory Single 16 GB CL16 BLS16G4D240FSE (Red) - Total 64 GB
- ASUS GeForce GTX 1080 8 GB Turbo Graphic Card TURBO-GTX1080-8G
- Samsung 850 EVO 500 GB 2.5-Inch SATA III Internal SSD (MZ-75E500B/AM)
- Seagate Barracuda ST2000DM001 2 TB 3.5 Internal Hard Drive (ST2000DM008)
- Seagate BarraCuda 4 TB Internal Hard Drive HDD – 3.5 Inch Sata 6 Gb/s 5400 RPM 256 MB Cache for Computer Desktop PC Laptop (ST4000DM04)
- Cooler Master GeminII S524 Version 2 CPU Air Cooler with 5 Direct Contact Heat Pipes (RR-G5V2-20PK-R1)
- NZXT Phantom 410 Mid Tower Computer Case, White/Blue (CA-PH410-W2)
- Corsair CX Series 750 Watt 80 Plus Bronze Certified Modular Power Supply

- **Desktop Ubuntu 18.04.2 64 bit**
  - ASUS ROG Crosshair V Formula-Z AM3+ AMD 990FX + SB950 SATA 6 Gb/s USB 3.0 ATX AMD Gaming Motherboard with 3-Way SLI/CrossFireX Support and UEFI BIOS
  - AMD FX-8350 Black Edition Vishera 8-Core 4.0 GHz (4.2 GHz Turbo) Socket AM3+ 125W FD8350FRHKBOX Desktop Processor
  - Crucial BX500 240 GB 3D NAND SATA 2.5-Inch Internal SSD - CT240BX500SSD1Z
  - G.SKILL Ripjaws X Series 16 GB (4 x 4 GB) 240-Pin DDR3 SDRAM DDR3 1866 (PC3 14900) Desktop Memory Model F3-14900CL9Q-16GBXL
  - XFX Radeon RS RX 480 DirectX 12 RX-480P836BM 8 GB 256-Bit GDDR5 PCI Express 3.0 CrossFireX Support Video Card
- CORSAIR HX Series HX850 850W ATX12V 2.3 / EPS12V 2.91 SLI Ready CrossFire Ready 80 PLUS GOLD Certified Modular Active PFC Power Supply New 4th Gen CPU Certified Haswell Ready
- Cooler Master Hyper 212 Plus - CPU Cooler with 4 Direct Contact Heat-pipes
- Cooler Master HAF 922 - High Air Flow Mid Tower Computer Case with USB 3.0 and All-Black Interior

- **Lenovo Laptop Windows 10 64 bit** - Lenovo Thinkpad W541 Intel i7-4910MQ processor 2.9 GHz running Windows 10 and 32 GB of RAM. HITACHI HTS725050A7E635 Travelstar Z7K500 Opal 500 GB Hard drive

- **ASUS Laptop Windows 10 64 bit** - ASUS VivoBook F556UA-AB32 Laptop (Windows 10, Intel i3-6100U 2.3 GHz, 15.6” LED-Lit Screen, Storage: 1000 GB, RAM: 4 GB) Black/Silver *Upgraded to 8 GB memory (added second Crucial 4 GB Single DDR4 2133 MT/s (PC4-17000) SR x8 SODIMM 260-Pin Laptop Memory - CT4G4SFS8213) and replaced HDD with Samsung 850 EVO 500 GB 2.5-Inch SATA III Internal SSD (MZ-75E500B/AM)*

- **Raspberry Pi 3 Model B+ Rev 1.3** - Raspbian 4.19.75-v7. Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz, 1 GB LPDDR2 SDRAM, 16 GB Class 4 SanDisk Edge microSDHC memory card, running Suri-cata Version 4.1.2

- **R7500-200NAS Nighthawk AC2350 4X4 MU-MIMO Dual Band WiFi Gigabit Router** - Router Firmware Version V1.0.3.16

- **Netgear ProSafe GS108Ev2 Switch** - Firmware version v1.00.12
- Ubiquiti EdgeRouter X Model ER-X - Firmware version v1.9.0
- TitaThink Camera TT520PW - Firmware Version 6.10
- Prolophix Thermostat NT130h - SW 3.0.3 / HW H.03
- ezOutlet2 Power Outlet EZ-22b - Version EZT.8824 (04)

Virtual Machines  Two virtual machine images are used in the experiment. The Windows 10 VM is used as the C2 server, and the Ubuntu VM is used to run Stafira’s HTTP IoT Honeyd Framework and the HoneyB Agent script. The Ubuntu VM is cloned and then distributed to the Windows Desktop, Lenovo Laptop, and ASUS Laptop in order add physical separation and network distribution between honeypots. Only IP and MAC addresses are modified in settings, scripts, and configs for the cloned VMs. The VMware specifications and versions of installed software are as follows:

- Ubuntu 12.04 VMs (Honeyd Machines) - 2 processors, 4.3 GB memory, 20 GB disk space. Honeyd 1.5c installed, bash version 4.2.25, Python version 2.7.3

- Windows 10 64 bit VM (C2 Server) - 4 logical processors, 8 GB memory, 40 GB disk space. Git, TortoiseGit, Node.js Version v12.13.0, V8 Engine Version 7.7.299.13-node.12, Snort Version 2.9.15-WIN32 GRE (Build 7), and Wireshark Version 3.0.6 are all installed

Program and Language Versions  This section documents the combined programs and programming languages across all devices:

- VMware Workstation 15 Pro Version 15.5.0
- Node.js Version v12.13.0
- V8 Engine Version 7.7.299.13-node.12
- Python 2.7.15+ on Ubuntu Desktop, Python 2.7.3 on Ubuntu VMs, Python 2.7.16 on Raspberry Pi
- Bash 4.4.20 on Ubuntu Desktop, 4.2.25 on Ubuntu VMs, and 5.0.3 on Raspberry Pi
- Honeyd 1.5c
- Nmap version 7.60
- Wireshark Windows Version 3.0.6
- Snort Version 2.9.15-WIN32 GRE (Build 7)
- Suricata Version 4.1.2

**Honeyd Configuration** The configuration file from Stafira’s HTTP IoT Honeyd framework is used as the configuration file for Honeyd 1.5c in this experiment and is shown in Appendix B. Only IP and MAC addresses are modified in this configuration file.

**Snort and Suricata Configuration** Snort Version 2.9.15-WIN32 GRE (Build 7) is installed on the C2 server and the default snort.conf file is used with some modifications in order to run it on windows with only the emerging-scan.rules rule set. Additionally, Suricata Version 4.1.2 is installed on the Raspberry Pi and the default configuration file, suricata.yaml, is modified to only use the emerging-scan.rules rule set. The snort.conf, suricat.yaml, and emerging-scan.rules are developed and vetted by the Snort community and available for download on their website.
Suricata was built to use the same rule syntax and can successfully implement these community rules as well.

### 4.4.3 Factors

The following factors were used in this experiment:

- **Scan Type (ST)** - The type of scan performed in the test. These include the control group (no scan), TCP Connect scan, Aggressive scan (OS detection, version detection, script scanning, and traceroute), and NIDS Avoidance scan (scan delay, host randomization, and packet fragmentation). The Nmap parameters of these commands are:
  
  1. No Scan (Control Group) - Wait 1321 seconds
  
  2. TCP Connect scan - “sudo nmap -iL ipLst.txt -sT -Pn -oX runNum.xml”
  
  
  4. NIDS Avoidance scan - “sudo nmap -iL ipLst.txt -scan-delay 1075ms –randomize-hosts -f 8 -Pn -oX runNum.xml”

- **Number of Honeypots (HP)** - The total number of honeypots spun up in the test. Honeypots are launched in multiples of three with each of the following types: TitaThink Camera, Prolophix Thermostat, and ez-Outlet2 Power Outlet. The level of honeypots are 0, 3, 6, and 9 honeypots.

Factors with their corresponding levels are shown in Table 2.

<table>
<thead>
<tr>
<th>Factors</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan Type (ST)</td>
<td>No scan (Control Group)</td>
</tr>
<tr>
<td></td>
<td>TCP Connect</td>
</tr>
<tr>
<td></td>
<td>Aggressive</td>
</tr>
<tr>
<td></td>
<td>NIDS Avoidance</td>
</tr>
<tr>
<td>Honeypots (HP)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>

**Table 2. Factors and Levels**
4.4.4 Metrics

The metrics from the experiment include:

- **Number of Snort Alerts Individually (SAI)** - Numeric value representing the total number of Snort alerts generated by parsing each PCAP file transferred to the C2 server, individually, and resetting Snort after each run.

- **Number of Distinct Types of Snort Alerts Individually (STI)** - Numeric value representing the number of distinct types of Snort rules triggered by parsing each PCAP file transferred to the C2 server, individually, and resetting Snort after each PCAP is parsed.

- **Number of Snort Alerts Merged (SAM)** - Numeric value representing the total number of Snort alerts generated by parsing each PCAP file transferred to the C2 server as a merged PCAP file.

- **Number of Distinct Types of Snort Alerts Merged (STM)** - Numeric value representing the number of distinct types of Snort rules triggered by parsing each PCAP file transferred to the C2 server as a merged PCAP file.

- **Number of Suricata Alerts (SuA)** - Numeric value representing the total number Suricata alerts generated receiving from SPAN traffic.

- **Number of Distinct Types of Suricata Alerts (SuT)** - Numeric value representing the number of distinct types of Suricata rules triggered from SPAN traffic.

- **Percentage of Packets HoneyHive Captured (% HHP)** - The percentage of packets the HoneyHive framework captured to and from the attacker, measured by the combined total packet count of all transferred PCAPs to the C2 server.
server and then divided by the number of received and transferred packets on the attacker’s machine

- **Percentage of Packets Suricata Captured (% SuP)** - The percentage of packets the Raspberry Pi captured to and from the attacker, measured by counting the number of packets sent and received to and from the attacker, divided by the number of received and transferred packets on the attacker’s machine.

The number of HoneyHive Interactions (HHI) is a numeric value that represents the number of times HoneyB Agents collectively reported an interaction with one of their monitored honeypots. Each honeypot should only create one interaction per test in this experiment for a total of three interactions per HoneyB Agent. For this reason, it was not used in statistical analysis against Snort and Suricata but is instead a verification that the honeypots and HoneyHive framework are operating correctly. Additionally, the attacker’s total number of packets sent and received during each test (AP) and the Elapsed Time of the test in seconds (ET) are useful for understanding each trial, but not used in analysis. Metrics are bolded in Appendix E.

## 4.5 Methodology

The procedural steps taken to run the experiment are as follows:

1. Ensure all devices are connected to the network with the correct static IP and that the network setup matches the network diagram.

2. Ensure that all program code/scripts are up to date and current with the current Git repository. Perform a Git pull for scripts that are not.

3. Ensure the C2 server launches on startup - Windows+R and then type shell:startup.
4. Ensure the suricataConnect.py script launches on startup in rc.local.

5. Ensure no remnants remain from previous experimental runs.

   (a) Remove experiment.txt and all .csv and .xml files from the attacker machine.

   (b) Remove /var/log/suricata/fast.log from the Suricata machine.

6. Ensure emerging-scan.rules is located in “C:\Snort\rules” and that it is the only active rule in the snort.conf file “C:\Snort\etc\snort.conf”.

7. Ensure emerging-scan.rules is located in “/etc/suricata/rules” and that it is the only active rule in the suricata.yaml file “/etc/suricata/suricata.yaml”.

8. Reboot all physical machines and virtual machines.

9. On each Honeyd VM, cd to honeyhive/scripts and then run “sudo python connectToServer.py”.

10. Ensure the Windows 10 VM is up and the C2 server is running.

11. On the attacker machine, run “sudo python runExperiment.py”.

12. Wait for experiment to complete (about 10 to 12 days).

13. Download nmap.csv file and perform analysis on results.

4.5.1 runExperiment.py

This section describes the functions the runExperiment.py executes.

1. Check if experiment.txt exists and is not empty.

   (a) Read in each line as a separate test run (Test type, Number of Honeyd Machines, and Test Type Level) and append it to an array for execution.
2. If experiment.txt does not exist.

   (a) Create an array with all combinations of user-specified test types (nmap-Scan), number of Honeyd machines (0-3), and Test Type Level (0-3) for each user-specified test. Each Honeyd machine controls 3 honeypots, creating the honeypot levels of 0, 3, 6, and 9. This is all then multiplied by the number of user-specified replicants (30).

   The nmapScan Test Type Levels:

   i. Level 0 - No Scan (Control Group): Wait 1321 seconds for routine network traffic (median amount of time required by the median nmapScan test in pilot studies). The mean was 1341 seconds.

   ii. Level 1 - Nmap TCP Connect scan ("sudo nmap -iL ipLst.txt -sT -Pn -oX runNum.xml").

   iii. Level 2 - Nmap TCP Syn scan with OS detection, version detection, script scanning, and traceroute. (Aggressive scan) ("sudo nmap -iL ipLst.txt -A -Pn -oX runNum.xml").

   iv. Level 3 - Nmap TCP Syn scan with scan delay, host randomization, and packet fragmentation (NIDS Avoidance scan) ("sudo nmap -iL ipLst.txt --scan-delay 1075ms --randomize-hosts -f 8 -Pn -oX runNum.xml"). These parameters were used because Nmap’s website listed them as ways to avoid a NIDS [38].

   (b) Randomize the order of the array.

   (c) Create experiment.txt and write contents of the recently created array to the file.

3. Execute all test runs in the array (loop).
(a) Send UDP “START” command to corresponding number of Honeyd machines and to Suricata over command and control port 9830, three times. The following devices are sent “START” based on the number of Honeyd Machines:

i. 0 - no Honeyd machines are started.

ii. 1 - 192.168.1.150-154 Honeyd machines are started.

iii. 2 - 192.168.1.150-154 and 192.168.1.170-174 Honeyd machines are started.


Suricata is always started.

(b) Wait 40 seconds for the Honeyd and Suricata programs to finish starting (based on pilot-studies, Honeyd takes 10-15 seconds and Suricata takes 30 seconds).

(c) Record the transmitted (tx_packets) and received packet (rx_packets) count and start time of the attacker machine before executing the test.

(d) Execute the next test run in the execution array and wait for completion.

(e) Record the transmitted (tx_packets) and received packet (rx_packets) count for the attacker’s interface and end time after executing the test. Subtract the end counts from the start counts to get the total packet counts sent and the elapsed time of the test.

(f) Send UDP “RESET” command to all Honeyd Machines and Suricata over command and control port 9830, three times.

(g) Wait 45 seconds for all the PCAPs and Suricata results to be transferred to the C2 server (based on pilot-studies, sending all PCAPs takes 20-30 seconds and sending Suricata results take 3-5.
(h) Send the “SNORT” command to the C2 server over TCP port 9830 to signal it to parse all received PCAPs.

(i) Wait 90 seconds for Snort to finish (Snort takes 30-50 seconds to complete, based on pilot studies).

(j) Send the “RESET” command to the C2 server over TCP port 9830 to signal for the C2 Server to send results over the socket and then clear all data. runExperiment.py waits until the results are received.

(k) Write the results received from the C2 server, attacker machine packet count and experiment elapsed time to the corresponding test result file in csv format (nmap.csv).

(l) Remove completed experiment from the experiment.txt file (first line).

(m) Send UDP “REBOOT” command to all Honeyd Machines and Suricata over command and control port 9830, three times (Honeyd Machines set to ignore command).

(n) Send the “REBOOT” command to the C2 server over TCP port 9830 to signal it to reboot the machine.

(o) Wait 45 seconds for all machines to finish rebooting (Windows takes 20-30 seconds, Ubuntu 15-20 seconds, Raspberry Pi 20-25 seconds).

(p) Ensure the C2 Server has booted by continuously trying to connect to it until successful.

4.6 Results

The results from this experiment are saved in Comma-Separated Values (CSV) format in the file nmap.csv. This file/data is imported into Excel and used to create bar graphs and tables to show the differences between the HoneyHive framework...
and that of a traditional NIDS (Suricata). Additionally, Anderson-Darling tests for normality are run in Excel [72]. Finally, using the permutation_test.py script in Appendix F, Permutation tests are run for SAI versus SAM, SAI / SAM versus SuA, STI versus STM, STM versus SuT, and % HHP versus % SuP.

The Anderson-Darling test is used to determine how well a set of data fits a specified distribution. In the analysis of results for this experiment this is used to test whether the given data is normal. This is done across all combinations of factors and is important to do because some statistical tests, such as t-tests, assume normal distribution. Suffice it to say that if the p-value from the Anderson-Darling test is less than 0.05 then the data does not follow a normal distribution.

Permutation tests do not require a normal distribution to perform statistical analysis and are used to determine what the probability is of obtaining results that exceed those in two competing data sets, given their initial means and difference. To perform a Permutation test, first the arithmetic mean for each of the two data sets is computed and then the difference between their arithmetic means is recorded as the “test statistic”. After that, the data from both sets is combined and shuffled into two new random subsets. The difference between the newly formed sets’ means is calculated and compared to the test statistic. This is repeated 900,000 times in this experiment, and then number of permutations whose absolute value exceeds that of the test statistic is divided by the total number of permutations performed to calculate the p-value. If this p-value is less than the chosen significance value, 10% in this experiment, then there is a statistical significance between the two data sets [73] [74] [75]. The code developed to perform Permutation tests on this experiment’s results is found in Appendix F.
4.7 Chapter Summary

This chapter outlined the research goals and questions this experiment seeks to answer. The experiment itself was broken down into the SUT, all system parameters, factors, and metrics, the step by step procedure of executing the experiment, and finally how the results of the experiment will be analyzed and presented.
V. Results and Analysis

5.1 Overview

This chapter provides an analysis of experiment results which can be found in Appendix E. Results are presented first with an overall picture and then grouped by each factor to distinguish patterns and the effect of each factor. Results are also grouped by all combinations of factors (scan type with number of honeypots). First tables and graphs present a comparison of each different metric. The Control Group Scan and 0 Honeypot levels are not included in the means of other factors as they create skewed results. This is followed by a table with the results of the Anderson-Darling test for normality with a significance level ($\alpha$) of 5%. Both metrics that are being compared against one another are required to be normally distributed in order to perform a t-test. While there are a handful of individual metrics that accept the null hypothesis of normal distribution, only the Aggressive scan with 9 honeypots accepted the null hypothesis for SAI and SAM to be compared against one another.

Because only one test set out of eighty is normally distributed, a Permutation test is selected for testing whether results are statistically significant with a significance level of 10% . 900,000 permutations are performed for each test. For number of alerts, SAI and SAM are tested against one another as well as the SAI/SAM (whichever has the higher average mean) against SuA. It appears that Snort consolidates some alerts when PCAPs are analyzed together which is why SAI is often the more fair test against SuA. However, it appears that Snort is able to detect and create more alerts when analyzing packets together for the NIDS Avoidance Scans and when only 3 honeypots are active. When this happens, SAM is compared against SuA. For the number of distinct alerts, STI is compared against STM, and STM is compared against SuT. Snort created more distinct alerts on average when analyzing PCAPs
merged together which is why STM was used over STI. Finally, % HHP is compared against % SuP to compare the percentage of captured packets.

5.1.1 Number of Alerts Overview

In Table 3 and Figure 18, the mean number of alerts across all groupings and combination of factors is shown. Suricata generated more alerts in all factor groupings except for when 9 honeypots were active in an Aggressive scan. Additionally, in the TCP Connect scan Snort and Suricata created a similar amount of alerts when 9 honeypots were active. However, Suricata created a significant number more alerts on average in the NIDS Avoidance scan when 9 honeypots were active. Table 4 displays the Anderson-Darling test for normality results conducted on the experiment results. Only the Aggressive Scan with 9 honeypots was normal for both SAI and SAM. Additionally, SAI was normal for the TCP Connect scan with 6 honeypots. In Table 5 the Permutation test results are shown. In the SAI versus SUA column, cells italicized and denoted with a “*” indicate that SAM was used for comparison instead because it had a higher mean for alerts than SAI. SAI and SAM were not statistically significant in test. However, Snort versus SuA was significantly different in all tests except the TCP Connect Scan with 9 honeypots and 9 honeypots overall. This indicates that the null hypothesis (Snort and Suricata create the same number of alerts) is rejected. This suggests Suricata outperforms Snort in all alert tests where its mean is higher and statistically significant. This is true for all but the Aggressive Scan with 9 honeypots, where Snort outperforms Suricata and is statistically significant.

While HoneyHive created alerts in the majority of runs, there were 3/270 runs (runs 270, 291, and 465) that it did not create alerts (HHI) when it should have. Additionally, HoneyHive is currently using Snort for a higher level of signature matching and alert creation. However, Snort did not create alerts for 32/270 runs that it should
have. This means roughly 10% of intrusions did not have successful signature matching performed on packet captures. This is either from Snort crashing, not finishing in a timely manner, simply not creating alerts, or an error in the HoneyHive framework.
<table>
<thead>
<tr>
<th></th>
<th>HHI</th>
<th>SAI</th>
<th>SAM</th>
<th>SuA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CG 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CG 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CG 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CG 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TCP Connect</td>
<td>5.933333333</td>
<td>24.14444444</td>
<td>22.07777778</td>
<td>33.95555556</td>
</tr>
<tr>
<td>TCP 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TCP 3</td>
<td>2.8</td>
<td>12.86666667</td>
<td>14.03333333</td>
<td>23</td>
</tr>
<tr>
<td>TCP 6</td>
<td>6</td>
<td>23.53333333</td>
<td>22.5</td>
<td>39.2</td>
</tr>
<tr>
<td>TCP 9</td>
<td>9</td>
<td>36.03333333</td>
<td>29.7</td>
<td>39.66666667</td>
</tr>
<tr>
<td>Aggressive</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agg 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Agg 3</td>
<td>2.833333333</td>
<td>40.56666667</td>
<td>42.63333333</td>
<td>103.5333333</td>
</tr>
<tr>
<td>Agg 6</td>
<td>5</td>
<td>74.06666667</td>
<td>71.83333333</td>
<td>126.5666667</td>
</tr>
<tr>
<td>Agg 9</td>
<td>7.933333333</td>
<td>133.43333333</td>
<td>116.9666667</td>
<td>107.1</td>
</tr>
<tr>
<td>NIDS Avoidance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIDS 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>54.55555556</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>3</td>
<td>13.86666667</td>
<td>17.23333333</td>
<td>28</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>6</td>
<td>22.6</td>
<td>25.83333333</td>
<td>58.1666667</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>8.9</td>
<td>32.3</td>
<td>39.3</td>
<td>61.93333333</td>
</tr>
<tr>
<td>0 Honeypots</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>54.55555556</td>
</tr>
<tr>
<td>3 Honeypots</td>
<td>2.877777778</td>
<td>22.43333333</td>
<td>24.63333333</td>
<td>51.51111111</td>
</tr>
<tr>
<td>6 Honeypots</td>
<td>5.666666667</td>
<td>40.06666667</td>
<td>40.05555556</td>
<td>74.64444444</td>
</tr>
<tr>
<td>9 Honeypots</td>
<td>8.611111111</td>
<td>67.25555556</td>
<td>61.98888889</td>
<td>69.56666667</td>
</tr>
<tr>
<td>Overall</td>
<td>5.718518519</td>
<td>43.25185185</td>
<td>42.22592593</td>
<td>65.24074074</td>
</tr>
</tbody>
</table>

Table 3. Overview - Mean Alerts by Level
Figure 18. Overview - Mean Number of Alerts
<table>
<thead>
<tr>
<th></th>
<th>SAI</th>
<th>SAM</th>
<th>SuA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Connect</td>
<td>8.4179E-10</td>
<td>2.10219E-06</td>
<td>8.45677E-05</td>
</tr>
<tr>
<td>TCP 3</td>
<td>1.86436E-09</td>
<td>1.80995E-09</td>
<td>0.001764216</td>
</tr>
<tr>
<td>TCP 6</td>
<td>0.07823086</td>
<td>0.018595137</td>
<td>3.75442E-05</td>
</tr>
<tr>
<td>TCP 9</td>
<td>0.003989696</td>
<td>0.04930214</td>
<td>0.001424823</td>
</tr>
<tr>
<td>Aggressive</td>
<td>0.00231741</td>
<td>0.009967393</td>
<td>4.53772E-10</td>
</tr>
<tr>
<td>Agg 3</td>
<td>0.001050125</td>
<td>0.000460888</td>
<td>4.02293E-10</td>
</tr>
<tr>
<td>Agg 6</td>
<td>0.01229756</td>
<td>0.045563122</td>
<td>0.000516952</td>
</tr>
<tr>
<td>Agg 9</td>
<td>0.118392405</td>
<td>0.136474624</td>
<td>0.002484829</td>
</tr>
<tr>
<td>NIDS Avoidance</td>
<td>0</td>
<td>7.24625E-25</td>
<td>4.65401E-19</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>1.03809E-09</td>
<td>1.02987E-07</td>
<td>4.93212E-11</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>9.41431E-15</td>
<td>4.06702E-13</td>
<td>3.23667E-12</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>1.18093E-13</td>
<td>4.58503E-11</td>
<td>5.75098E-11</td>
</tr>
<tr>
<td>3 Honeypots</td>
<td>1.90455E-12</td>
<td>1.2306E-10</td>
<td>7.56855E-23</td>
</tr>
<tr>
<td>6 Honeypots</td>
<td>1.14242E-13</td>
<td>3.13753E-13</td>
<td>7.32928E-21</td>
</tr>
<tr>
<td>9 Honeypots</td>
<td>3.86714E-12</td>
<td>1.15597E-11</td>
<td>1.65326E-15</td>
</tr>
<tr>
<td>Overall</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4. Anderson-Darling Test - Number of Alerts
<table>
<thead>
<tr>
<th></th>
<th>SAI v. SAM</th>
<th>SAI v. SuA</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Connect</td>
<td>0.489953333</td>
<td>0.00048</td>
</tr>
<tr>
<td>TCP 3</td>
<td>0.74778</td>
<td>*0.0034967</td>
</tr>
<tr>
<td>TCP 6</td>
<td>0.762092222</td>
<td>0.0001189</td>
</tr>
<tr>
<td>TCP 9</td>
<td>0.339487778</td>
<td>0.49631</td>
</tr>
<tr>
<td>Aggressive</td>
<td>0.497806667</td>
<td>*0.0006667</td>
</tr>
<tr>
<td>Agg 3</td>
<td>0.76365</td>
<td>0</td>
</tr>
<tr>
<td>Agg 6</td>
<td>0.817308889</td>
<td>0.0016511</td>
</tr>
<tr>
<td>Agg 9</td>
<td>0.275418889</td>
<td>0.0702367</td>
</tr>
<tr>
<td>NIDS Avoidance</td>
<td>0.310293333</td>
<td>0 *</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>0.462257778</td>
<td>*0.0015556</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>0.687624444</td>
<td>*3.33E-06</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>0.453642222</td>
<td>*0.0007711</td>
</tr>
<tr>
<td>3 Honeypots</td>
<td>0.532037778</td>
<td>0 *</td>
</tr>
<tr>
<td>6 Honeypots</td>
<td>0.996777778</td>
<td>3.33E-06</td>
</tr>
<tr>
<td>9 Honeypots</td>
<td>0.555283333</td>
<td>0.7710211</td>
</tr>
<tr>
<td>Overall</td>
<td>0.795945556</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5. Permutation Test - Number of Alerts

5.1.2 Number of Distinct Types of Alerts Overview

Table 6 and Figure 19 show the mean number of distinct alerts across all tests. Suricata created more distinct number of alerts (SuT) on average than Snort. PCAPs merged together and then analyzed by Snort (STM) created more distinct alerts on average as compared with each packet being parsed individually by Snort (STI). Furthermore, not only did the average number of distinct alerts grow for STM and
STI as the number of honeypots increased but the average difference in the number of distinct alerts between STM and STI grew as well. In addition, the difference between STM and SuT shrank. This supports that distributed data / PCAP aggregation exhibits merit for NIDS. Table 7 shows the results of the Anderson-Darling test. No data sets fit the normal distribution. Table 8 displays the results of Permutation tests for STI compared with STM and STM compared with SuT. STI versus STM was statistically significant in all tests except the TCP Connect Scan and NIDS Avoidance Scan each with 3 honeypots. These tests were both very close to 10% but did not meet the required significance level. Because STM created more distinct alerts on average for each test over STI this means that it performed better for all tests except the two aforementioned. This supports the hypothesis that the HoneyHive framework can detect intrusions that traditional NIDS cannot through distributed sensors and packet aggregation. However, Suricata created more distinct alerts on average than SAM in all categories and was statistically significant in all tests except the Aggressive Scan with 9 Honeypots. This indicates there was not an advantage of combining packets over that of just listening on a SPAN port.
<table>
<thead>
<tr>
<th></th>
<th>STI</th>
<th>STM</th>
<th>SuT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control Group</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CG 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CG 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CG 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CG 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>TCP Connect</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP 0</td>
<td>4.477777778</td>
<td>5.5444444444</td>
<td>7.4111111111</td>
</tr>
<tr>
<td>TCP 3</td>
<td>4.3333333333</td>
<td>5.0333333333</td>
<td>6.7333333333</td>
</tr>
<tr>
<td>TCP 6</td>
<td>4.6333333333</td>
<td>5.8666666667</td>
<td>7.5</td>
</tr>
<tr>
<td>TCP 9</td>
<td>4.4666666667</td>
<td>5.7333333333</td>
<td>8</td>
</tr>
<tr>
<td><strong>Aggressive</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agg 0</td>
<td>6.3</td>
<td>7.8222222222</td>
<td>10.13333333</td>
</tr>
<tr>
<td>Agg 3</td>
<td>5.7</td>
<td>7.1</td>
<td>11.36666667</td>
</tr>
<tr>
<td>Agg 6</td>
<td>6.2333333333</td>
<td>7.7333333333</td>
<td>9.6</td>
</tr>
<tr>
<td>Agg 9</td>
<td>6.9666666667</td>
<td>8.6333333333</td>
<td>9.4333333333</td>
</tr>
<tr>
<td><strong>NIDS Avoidance</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIDS 0</td>
<td>3.6333333333</td>
<td>4.9111111111</td>
<td>7.2333333333</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>3.3333333333</td>
<td>4.2333333333</td>
<td>5.8</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>3.7</td>
<td>4.4666666667</td>
<td>7.9666666667</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>3.8666666667</td>
<td>6.0333333333</td>
<td>7.9333333333</td>
</tr>
<tr>
<td><strong>0 Honeypots</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Honeypots</td>
<td>0</td>
<td>0</td>
<td>8.388888889</td>
</tr>
<tr>
<td><strong>3 Honeypots</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Honeypots</td>
<td>4.4555555556</td>
<td>5.4555555556</td>
<td>7.9666666667</td>
</tr>
<tr>
<td><strong>6 Honeypots</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Honeypots</td>
<td>4.8555555556</td>
<td>6.0222222222</td>
<td>8.3555555556</td>
</tr>
<tr>
<td><strong>9 Honeypots</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 Honeypots</td>
<td>5.1</td>
<td>6.8</td>
<td>8.4555555556</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>4.803703704</td>
<td>6.092592593</td>
<td>8.259259259</td>
</tr>
</tbody>
</table>

Table 6. Overview - Mean Number of Distinct Alerts by Level
Figure 19. Overview - Mean Number of Distinct Alerts
<table>
<thead>
<tr>
<th></th>
<th>STI</th>
<th>STM</th>
<th>SuT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TCP Connect</strong></td>
<td>1.42E-30</td>
<td>2.26E-18</td>
<td>0</td>
</tr>
<tr>
<td><strong>TCP 3</strong></td>
<td>2.26E-10</td>
<td>1.04E-06</td>
<td>3.82E-08</td>
</tr>
<tr>
<td><strong>TCP 6</strong></td>
<td>4.44E-12</td>
<td>2.19E-13</td>
<td>3.78E-16</td>
</tr>
<tr>
<td><strong>TCP 9</strong></td>
<td>1.47E-10</td>
<td>2.38E-08</td>
<td>3.96E-15</td>
</tr>
<tr>
<td><strong>Aggressive</strong></td>
<td>0</td>
<td>0</td>
<td>1.1E-12</td>
</tr>
<tr>
<td><strong>Agg 3</strong></td>
<td>9.08E-12</td>
<td>3.92E-14</td>
<td>5.21E-20</td>
</tr>
<tr>
<td><strong>Agg 6</strong></td>
<td>2.25E-14</td>
<td>2.56E-16</td>
<td>0.005961</td>
</tr>
<tr>
<td><strong>Agg 9</strong></td>
<td>4.67E-08</td>
<td>6.4E-13</td>
<td>7.33E-05</td>
</tr>
<tr>
<td><strong>NIDS Avoidance</strong></td>
<td>9.44E-29</td>
<td>1.93E-20</td>
<td>0</td>
</tr>
<tr>
<td><strong>NIDS 3</strong></td>
<td>6.37E-09</td>
<td>2.95E-10</td>
<td>2.66E-08</td>
</tr>
<tr>
<td><strong>NIDS 6</strong></td>
<td>1.61E-08</td>
<td>9.65E-12</td>
<td>8.99E-24</td>
</tr>
<tr>
<td><strong>NIDS 9</strong></td>
<td>4.02E-13</td>
<td>1.5E-13</td>
<td>2.28E-27</td>
</tr>
<tr>
<td><strong>3 Honeypots</strong></td>
<td>2.01E-10</td>
<td>4.23E-11</td>
<td>2.33E-09</td>
</tr>
<tr>
<td><strong>6 Honeypots</strong></td>
<td>2.16E-10</td>
<td>1.93E-10</td>
<td>1.72E-20</td>
</tr>
<tr>
<td><strong>9 Honeypots</strong></td>
<td>1.34E-08</td>
<td>7.63E-15</td>
<td>4.72E-22</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>3.26E-27</td>
<td>1.58E-28</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7. Anderson-Darling Test - Number of Distinct Alerts
<table>
<thead>
<tr>
<th></th>
<th>STI v. STM</th>
<th>STM v. SuT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Connect</td>
<td>0.00062</td>
<td>0</td>
</tr>
<tr>
<td>TCP 3</td>
<td>0.1014511</td>
<td>0.0011956</td>
</tr>
<tr>
<td>TCP 6</td>
<td>0.0273578</td>
<td>0.0015211</td>
</tr>
<tr>
<td>TCP 9</td>
<td>0.0310778</td>
<td>0</td>
</tr>
<tr>
<td>Aggressive</td>
<td>0.0003433</td>
<td>0</td>
</tr>
<tr>
<td>Agg 3</td>
<td>0.0726156</td>
<td>0</td>
</tr>
<tr>
<td>Agg 6</td>
<td>0.0417956</td>
<td>0.0032133</td>
</tr>
<tr>
<td>Agg 9</td>
<td>0.0042444</td>
<td>0.1583944</td>
</tr>
<tr>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>0.1012789</td>
<td>0.0982622</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>0.0897022</td>
<td>0</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>1.22E-05</td>
<td>0</td>
</tr>
<tr>
<td>3 Honeypots</td>
<td>0.0084811</td>
<td>0</td>
</tr>
<tr>
<td>6 Honeypots</td>
<td>0.0026778</td>
<td>0</td>
</tr>
<tr>
<td>9 Honeypots</td>
<td>1.00E-05</td>
<td>0</td>
</tr>
<tr>
<td>Overall</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 8. Permutation Test - Number of Distinct Alerts

5.1.3 Packet Capture Percentage Overview

The percentages of scanned devices that are monitored for each level of honeypots in the test network for the HoneyHive framework and Suricata, are shown in Table 9. As the number of honeypots increases, the total number of active devices on the network that are scanned also increases. HoneyHive’s percentage of monitored devices increases as the number of honeypots increases. However, the percentage of monitored
devices for Suricata decreases from 0 to 3 honeypots, increase from 3 to 6 honeypots, and then decreases from 6 to 9 honeypots. This is because the honeypots activated for 6 honeypots reside on Suricata’s switch (the DMZ switch).

The percentage of packets captured by the HoneyHive framework versus Suricata is shown in Table 10 and Figure 20. Overall, HoneyHive and Suricata captured roughly the same average percentage of packets. However, when 9 honeypots are active the HoneyHive framework significantly outperforms Suricata in all categories except the NIDS Scan. At the level of three honeypots, packets captured are within several percentage points of another which is to be expected as Suricata does not have active honeypots on its switch. Similarly, at 6 honeypots HoneyHive and Suricata stayed within a couple percentage points of one another since they both increase by three honeypots. The one category this does not hold for is the NIDS Avoidance scan in which Suricata captures significantly more traffic than HoneyHive. Part of the reason for this could be how packets are counted in Snort versus Suricata. Capinfos counts all the packets in the PCAP and could be combining the fragmented parts together, reducing the actual number. Suricata reports by using Scapy to increment its packet for every packet sent or received with the attacker’s IP address. Table 11 shows the Anderson-Darling normality test for the percentage of packets captured but only % HPP for the TCP Connect Scan with 6 honeypots and % SuP for the Aggressive Scan with 9 honeypots are normal. Table 12 shows the Permutation test results for percentage of packets captured. All but TCP Connect overall, TCP Connect Scan

<table>
<thead>
<tr>
<th>HoneyHive</th>
<th>Suricata</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 / 6 = 0.00%</td>
</tr>
<tr>
<td>3</td>
<td>3 / 9 = 33.33%</td>
</tr>
<tr>
<td>6</td>
<td>6 / 12 = 50.00%</td>
</tr>
<tr>
<td>9</td>
<td>9 / 15 = 60.00%</td>
</tr>
</tbody>
</table>

Table 9. Percentage of Scanned Devices Monitored in Test Network
with 3 and 6 Honeypots (although very close), Aggressive Scan with 6 honeypots, and overall results are statistically significant. % HHP performed significantly better than % SuP in the Aggressive Scan overall and all tests with 9 honeypots, minus the NIDS Avoidance Scan. % SuP performed significantly better in all NIDS Avoidance Scans, Aggressive Scan with 3 honeypots, and 3 and 6 Honeypots overall. Even though the HoneyHive framework does not at this moment create more alerts or more distinct alerts than Suricata, the fact that it significantly outperforms Suricata in capturing network traffic when all 9 honeypots are active supports it will be able to detect more than a traditional NIDS could. Also of note, Suricata did not significantly outperform the HoneyHive framework in any of the TCP Connect scans.

Figure 21 shows the trend of an increase in mean capture packet percentage (% HHP) for the HoneyHive framework as the ratio of honeypots to devices scanned on a network also increases. In the TCP Connect scan each addition of 3 honeypots increases % HHP by 9%. For the Aggressive scan there is a 36% increase for % HHP from 3 to 6 honeypots and then a 39% increase from 6 to 9 honeypots. The NIDS Avoidance scan only has a 5% increase from 3 to 6 honeypots and an 8% increase from 6 to 9 honeypots. Finally, % HHP for each level of honeypot overall increase 17% from 3 to 6 and 18% from 6 to 9. Assuming this trendline continues, at 12 and 15 honeypots % HHP would be 62.5% and 80% respectively. This well exceeds % SuP which would continue to decrease with the addition of honeypots not connected to the switch it monitors.
<table>
<thead>
<tr>
<th></th>
<th>% HHP</th>
<th>% SuP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control Group</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CG 0</td>
<td>0%</td>
<td>21%</td>
</tr>
<tr>
<td>CG 3</td>
<td>0%</td>
<td>5%</td>
</tr>
<tr>
<td>CG 6</td>
<td>0%</td>
<td>6%</td>
</tr>
<tr>
<td>CG 9</td>
<td>0%</td>
<td>49%</td>
</tr>
<tr>
<td><strong>TCP Connect</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCP 0</td>
<td>0%</td>
<td>19%</td>
</tr>
<tr>
<td>TCP 3</td>
<td>13%</td>
<td>10%</td>
</tr>
<tr>
<td>TCP 6</td>
<td>22%</td>
<td>28%</td>
</tr>
<tr>
<td>TCP 9</td>
<td>31%</td>
<td>21%</td>
</tr>
<tr>
<td><strong>Aggressive</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agg 0</td>
<td>0%</td>
<td>4%</td>
</tr>
<tr>
<td>Agg 3</td>
<td>2%</td>
<td>4%</td>
</tr>
<tr>
<td>Agg 6</td>
<td>38%</td>
<td>35%</td>
</tr>
<tr>
<td>Agg 9</td>
<td>77%</td>
<td>41%</td>
</tr>
<tr>
<td><strong>NIDS Avoidance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NIDS 0</td>
<td>0%</td>
<td>32%</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>15%</td>
<td>23%</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>20%</td>
<td>40%</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>28%</td>
<td>34%</td>
</tr>
<tr>
<td><strong>0 Honeypots</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Honeypots</td>
<td>0%</td>
<td>18%</td>
</tr>
<tr>
<td><strong>3 Honeypots</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Honeypots</td>
<td>10%</td>
<td>12%</td>
</tr>
<tr>
<td><strong>6 Honeypots</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 Honeypots</td>
<td>27%</td>
<td>34%</td>
</tr>
<tr>
<td><strong>9 Honeypots</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 Honeypots</td>
<td>45%</td>
<td>32%</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>27%</td>
<td>26%</td>
</tr>
</tbody>
</table>

Table 10. Overview - Mean Packet Capture Percentage
Figure 20. Overview - Mean Packet Capture Percentage
Figure 21. HoneyHive Framework Mean Packet Capture Percentage (% HHP) by Level
<table>
<thead>
<tr>
<th></th>
<th>% HHP</th>
<th>% SuP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Connect</td>
<td>2.4739E-05</td>
<td>1.18E-08</td>
</tr>
<tr>
<td>TCP 3</td>
<td>1.7612E-08</td>
<td>0.001415</td>
</tr>
<tr>
<td>TCP 6</td>
<td>0.08450075</td>
<td>7.71E-10</td>
</tr>
<tr>
<td>TCP 9</td>
<td>0.00013778</td>
<td>0.013687</td>
</tr>
<tr>
<td>Aggressive</td>
<td>1.2062E-18</td>
<td>2.86E-14</td>
</tr>
<tr>
<td>Agg 3</td>
<td>1.2349E-07</td>
<td>3.13E-08</td>
</tr>
<tr>
<td>Agg 6</td>
<td>6.8951E-05</td>
<td>2.95E-05</td>
</tr>
<tr>
<td>Agg 9</td>
<td>5.1989E-06</td>
<td>0.474596</td>
</tr>
<tr>
<td>NIDS Avoidance</td>
<td>0.00022313</td>
<td>1.95E-05</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>9.268E-09</td>
<td>1.77E-08</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>3.2239E-06</td>
<td>1.11E-10</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>2.6557E-08</td>
<td>0.00294</td>
</tr>
<tr>
<td><strong>3 Honeypots</strong></td>
<td>2.018E-08</td>
<td>2.23E-13</td>
</tr>
<tr>
<td><strong>6 Honeypots</strong></td>
<td>1.2178E-15</td>
<td>3.65E-05</td>
</tr>
<tr>
<td><strong>9 Honeypots</strong></td>
<td>8.2817E-10</td>
<td>4.19E-05</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>0</td>
<td>9.39E-07</td>
</tr>
</tbody>
</table>

Table 11. Anderson-Darling Test - Packet Capture Percentage
Table 12. Permutation Test - Packet Capture Percentage

<table>
<thead>
<tr>
<th>Scan Type</th>
<th>% HHP v % SuP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Connect</td>
<td>0.20446</td>
</tr>
<tr>
<td>TCP 3</td>
<td>0.109532222</td>
</tr>
<tr>
<td>TCP 6</td>
<td>0.10659</td>
</tr>
<tr>
<td>TCP 9</td>
<td>0.003485556</td>
</tr>
<tr>
<td>Aggressive</td>
<td>0.010246667</td>
</tr>
<tr>
<td>Agg 3</td>
<td>0</td>
</tr>
<tr>
<td>Agg 6</td>
<td>0.667023333</td>
</tr>
<tr>
<td>Agg 9</td>
<td>0.003394444</td>
</tr>
<tr>
<td>NIDS Avoidance</td>
<td>0</td>
</tr>
<tr>
<td>NIDS 3</td>
<td>0</td>
</tr>
<tr>
<td>NIDS 6</td>
<td>0</td>
</tr>
<tr>
<td>NIDS 9</td>
<td>0</td>
</tr>
<tr>
<td>3 Honeypots</td>
<td>0.081776667</td>
</tr>
<tr>
<td>6 Honeypots</td>
<td>0.019293333</td>
</tr>
<tr>
<td>9 Honeypots</td>
<td>3.78E-05</td>
</tr>
<tr>
<td>Overall</td>
<td>0.537306667</td>
</tr>
</tbody>
</table>

5.2 Scan Type

This section groups experiment results by each of the different scan types for analysis. These scan types are No Scan (Control Group), TCP Connect, Aggressive, and NIDS Avoidance.
5.2.1 Control Group

For the control group, where no scan was performed, no alerts were generated for any of the metrics (HHI, SAI, SAM, and SuA). Similarly, without any alerts there were also no distinct types of alerts (STI, STM, and SuT). This supports that the HoneyHive framework operates correctly without any malicious traffic and does not create false positives. The packets captured by the HoneyHive framework were zero for all different levels of honeypots as is expected since no packets were sent to the honeypots. However, Suricata still captured traffic as shown in Figure 22. Although no malicious traffic was sent by the attacker, routine network traffic still takes place and this is the traffic that is captured by Suricata. One interesting aspect about the data is the sharp jump in percentage when 9 honeypots are active as compared to the other levels of honeypots. Roughly seven times the traffic captured in other levels is captured in this level. This is indicative that more traffic traverses the switch Suricata is on. Upon further investigation, one trial had a massive outlier of 1247% captured. When removing this outlier and reanalyzing, the mean falls in line with the other honeypot levels. Because Suricata’s packet count only contains traffic to or from the attacker and the percentage was much higher than what the attacker sent, some device must have been sending traffic to the attacker’s IP. It is speculated that this is attributed to Honeyd expiring all scan connections made by the attacker in the previous trial (234) aggressive scan.
Figure 22. Control Group - Mean Percentage of Packets Captured
5.2.2 TCP Connect

In Figure 23 Suricata generates more alerts on average in all levels of honeypots for the TCP Connect scan. When 9 honeypots are active, SAI is only a few alerts away from SuA. One aspect of note is that after 3 honeypots, SAI becomes and remains higher than SAM. It is speculated that this is partially due to Snort aggregating some alerts it sees relatively close to one another to reduce the number to parse through. Additionally, by parsing combined packets, it appears that Snort is able to create more alerts at levels of honeypots with a lower amount of traffic.

In Figure 24, STM significantly outperforms STI in honeypot levels 6, 9, and overall. However, Suricata significantly outperforms STM in all TCP connect tests. One interesting aspect to note is that the average at 6 honeypots is higher than that of 9 for STI and STM. Upon looking at the individual data, there were four runs where Snort reports 0 for the number of alerts and the distinct number of alerts in the 9 honeypot level and three runs for the 6 honeypot level. Upon removing these, the new averages are STI:5.15, STM: 6.62 for 9 honeypots and STI:5.15, STM: 6.52, for 6 honeypots, supporting the trend of more honeypots creating more distinct alerts with diminishing returns. The 0 runs are most likely from Snort crashing or not finishing before results were collected.

Figure 25 shows the average percentage of packets captured by the HoneyHive framework compared to that of Suricata. The HoneyHive framework begins with a higher capture percentage at the 3 honeypots level but then Suricata captures more at 6 honeypots. This is interesting because the number of honeypots increases equally for the HoneyHive framework and Suricata. However, at 9 honeypots and for TCP Connect scan overall the HoneyHive framework captures more traffic on average than Suricata. The capture percentage of HoneyHive at 9 honeypots is statistically higher than that of Suricata. No other levels were statistically significant. Another trend
expected to repeat is a drop in Suricata packet capture percentage from 6 to 9 due to an increase in attacker traffic that the SPAN port does not receive.
Figure 23. TCP Connect - Mean Number of Alerts
Figure 24. TCP Connect - Mean Number of Distinct Alerts
Figure 25. TCP Connect - Mean Packet Capture Percentage
5.2.3 Aggressive

In Figure 26 the trend holds for SAM creating more alerts than SAI at 3 honeypots and then SAI creating more alerts than SAM at 6 and 9 honeypots. Suricata performs significantly better overall and when 3 or 6 honeypots are active. SAI performs significantly better than SuA when 9 honeypots are active, supporting the hypothesis that distributed sensors can detect intrusions that traditional NIDS cannot. This is the only test in which the HoneyHive framework creates more alerts than Suricata although at 9 honeypots in the TCP connect scan, it is also very close to that of Suricata.

Figure 27 also follows the trend of the corrected data in the TCP connect scan, an increase in distinct number of alerts in STI and STM with STM being significantly greater than STI. SuT is significantly greater overall and for 3 and 6 honeypots. Even though it is larger, it is not significantly greater than STM at 9 honeypots. One interesting trend is that the number of distinct alerts for SuT actually decreases as the number of honeypots increases. This could be caused by the increased time spent scanning each host as more honeypots are activated and not tripping rules that are triggered by consecutive scanning in a certain amount of time.

The average packet percentage for each honeypot level in the Aggressive Scan is shown in Figure 28. The HoneyHive framework captures a higher percentage on average overall and for 6 honeypots and significantly more for 9 honeypots. Even though Suricata does not drop in packet percentage captured as it did in the TCP Connect scan, it does not increase nearly as much as it did from 3 to 6 honeypots.
Figure 26. Aggressive - Mean Number of Alerts
Figure 27. Aggressive - Mean Number of Distinct Alerts
Figure 28. Aggressive - Mean Packet Capture Percentage
5.2.4 NIDS Avoidance

The average number of alerts for the NIDS Avoidance Scan with each level of honeypot is shown in Figure 29. While not statistically significant, SAM performs better than STI in all levels of honeypots. This is most likely due to the packet fragmentation and scan delay not tripping as many Snort alerts, although more alerts are created as more honeypots are active and more packets are aggregated. Because SAM creates more alerts on average over SAI, it is compared against SuA. Unlike the TCP Connect scan with 9 honeypots and Aggressive scan with 9 honeypots, Suricata significantly outperforms Snort in the 9 honeypots level and all other tests.

Figure 30 shows the average distinct number of alerts for the NIDS Avoidance scan, broken down by each honeypot level and overall. All metrics increase as the number of honeypots they monitor increase as well. STM outperforms STI in all tests and is statistically significant in all tests except with 3 honeypots (just over 10%). SuT is statistically superior in performance in all tests for the NIDS Avoidance scan.

The trend of the percentage of packets captured by Suricata declining from 6 to 9 honeypots holds and is shown in Figure 31. However, in the NIDS Avoidance Scan % SuP significantly outperforms % HHP in all tests, even when 9 honeypots are active. % HHP still increases with each honeypot level though. As mentioned in the overview, the packet count may be different if Capinfos is combining fragmented packets together, resulting in a reduced packet count.
Figure 29. NIDS Avoidance - Mean Number of Alerts
Figure 30. NIDS Avoidance - Mean Number of Distinct Alerts
Figure 31. NIDS Avoidance - Mean Packet Capture Percentage
5.3 Number of Honeypots

This section groups experiment results by each of the different levels of honeypots for analysis. These levels are 0, 3, 6, and 9.

5.3.1 0 Honeypots

Without any active honeypots, no alerts should be created and no packets captured. This section is useful in understanding the baseline performance of Suricata across different scan types. The average number of alerts is shown in Figure 32. Suricata creates alerts in all scan types with the most in the Aggressive scan, followed by the NIDS Avoidance scan and then the TCP Connect scan. The number of alerts increases as the number of honeypots increases for TCP Connect and NIDS but actually decreases for all honeypots not monitored by it (all but 6). This is most likely due to a decrease in alerts being triggered as hosts being scanned in sequence takes longer since each scan takes longer. Suricata performed correctly by not creating any alerts when no scan was performed.

Figure 30 displays the average number of distinct alerts created by Suricata for each different type of scan. The Aggressive scan creates the most number of distinct alerts followed by TCP Connect scan and then the NIDS Avoidance scan. The distinct number of alerts actually decreases for the Avoidance scan as each honeypot level increases. The TCP Connect scan decreases for the 3 honeypots but then increases for 6 and 9 honeypots. The NIDS Avoidance scan similarly decreases for 3 honeypots, increases for 6, but then decreases slightly again for 9 honeypots.

The average percentage of packets captured by Suricata for each test is shown in Figure 34. The NIDS Avoidance scan captured the largest percentage of packets on average, followed by the TCP Connect scan, the Control Group, and then the Aggressive Scan. The NIDS Avoidance scan capture percentage being so high with
0 honeypots supports that it is being skewed in other tests. The TCP Connect scan results are within the expected amount for only one monitored device out of three. With the attacker sending no scan packets in the control group, one might wonder why packets are being captured. These packets are command and control packets for running tests. It is surprising that the Aggressive scan packet capture percentage was so low. This could be because the actual IoT devices cannot keep up with the scan so ones not on this switch generate lots of re-transmissions that are not captured or Nmap quickly dismisses the IoT device on the switch and moves on.
Figure 32. 0 Honeypots - Mean Number of Alerts
Figure 33. 0 Honeypots - Mean Number of Distinct Alerts
Figure 34. 0 Honeypots - Mean Packet Capture Percentage
5.3.2 3 Honeypots

Figure 35 shows the average number of alerts with 3 honeypots active grouped by different scan types. All three metrics create more alerts in the Aggressive scan than the other scan types. This is followed by the NIDS Avoidance Scan, and the TCP Connect scan. Suricata creates the most alerts in all tests followed by SAM and then SAI. Even though the NIDS Avoidance scan was designed to trigger the least number of alerts, it appears that the NIDS have the most difficulty in creating alerts for the TCP Connect scan. This is probably because the TCP Connect scan is difficult to distinguish from normal connections while the NIDS Avoidance scan sends fragmented packets, which might be treated as suspicious. The NIDS clearly have no trouble detecting Aggressive Scans which makes sense because the scan generates much more traffic by requesting OS and service information, giving it a very identifiable signature.

Figure 36 displays the number of distinct alerts for each scan with 3 honeypots active. Based on the sheer number of alerts generated by the Aggressive scan, it makes sense that the distinct number would also be higher than the other scan types. However, even though the NIDS Avoidance scan generated more alerts than the TCP Connect scan, the TCP Connect scan had a larger number of distinct alerts than that of the NIDS Avoidance scan for STI, STM, and SuT. This suggest that the NIDS Avoidance scan is repeatedly generating the same alerts while TCP connect has a broader spectrum.

In Figure 37, % SuP captures more traffic than % HHP in all scans except the TCP Connect scan. The NIDS Avoidance scan has the highest percentage of captured for both % SuP and % HHP followed by the TCP Connect and then the Aggressive Scan (although the Control Group is higher for % SuP). Because packets are fragmented and sent at a much slower rate, it appears that Suricata and HoneyHive are able to better capture them in the NIDS Avoidance scan with three honeypots active. With
0 honeypots active on its switch, it would appear that a single IoT device scanned creates more traffic than three scanned honeypots. In the TCP Connect scan, % HHP exceeds % SuP suggesting that IoT devices handle normal connections better than ones with long delays and fragmented packets. The aggressive scan being the lowest makes sense with the large amount of traffic that is generated and with only a small fraction of the devices being scanned being active honeypots.
Figure 35. 3 Honeypots - Mean Number of Alerts
Figure 36. 3 Honeypots - Mean Number of Distinct Alerts
Figure 37. 3 Honeypots - Mean Packet Capture Percentage
5.3.3 6 Honeypots

The same pattern for average number of alerts holds for Figure 38 for 6 honeypots as it did with 3. The main differences are more alerts are generated in each scan and now SAI create more alerts than SAM in the TCP Connect and Aggressive scans. SAM still creates more alerts than SAI in the NIDS Avoidance scan.

With 6 honeypots the trend for the average number of distinct alerts is mostly the same as 3 honeypots and is shown in Figure 39. SuT is still the highest in all scan types but the average distinct number decreases in the Aggressive Scan. Also of note, SuT creates more alerts in the NIDS Avoidance scan instead of the TCP Connect scan, this was the opposite with 3 honeypots. STM remains higher than STI in all scan tests, and averages for both increase. Additionally, the difference between STM and SuT for the average number of distinct alerts decreases in all scans except the NIDS Avoidance scan.

Unlike scans with 3 honeypots, % HHP is the largest in the Aggressive scan and the least in the NIDS Avoidance scan, as shown in Figure 40. % SuP is still largest in the NIDS Avoidance scan but the Aggressive scan captures a higher pecentage than the TCP Connect scan. % SuP also remains higher than % HHP in the NIDS Avoidance scan, surpasses %HHP in the TCP Connect scan, and falls below % HHP in the Aggressive scan.
Figure 38. 6 Honeypots - Mean Number of Alerts
Figure 39. 6 Honeypots - Mean Number of Distinct Alerts
Figure 40. 6 Honeypots - Mean Packet Capture Percentage
5.3.4 9 Honeypots

With 9 honeypots active SAI approached SuA in the TCP Connect scan and significantly surpassed SuA in the Aggressive scan, shown in Figure 41. SuT remained on top and STM remained higher than the STI in the NIDS avoidance scan. SuT actually decreased in the number alerts generated from 6 honeypots to 9 honeypots in the Aggressive scan.

Figure 42 shows the average number of distinct alerts across different scans with 9 honeypots active. Noticeable changes are STM closing in on SuT and there no longer being a significant difference between the two, STM being higher in the NIDS Avoidance scan than the TCP Connect scan, and SuT being approximately the same in the TCP Connect and NIDS Avoidance scans, although TCP Connect is now higher. In fact, the average number of alerts actually decreases for SuT in the Aggressive and NIDS Avoidance scans.

In Figure 43, % HHP now surpasses % SuP in all scans except the NIDS Avoidance scan. Another noticeable difference is % SuP is now higher in the Aggressive scan compared with the NIDS Avoidance scan. The percentage of packets captured by % SuP actually decreases for both the TCP Connect and NIDS Avoidance scans.
**Figure 41. 9 Honeypots - Mean Number of Alerts**

The diagram illustrates the mean number of alerts for different categories: Control Group, TGP Connect, Aggressive, NIDS Avoidance, and 9 Honeypots Overall. The y-axis represents the number of alerts ranging from 0 to 140, and the x-axis lists the various categories. Each category is represented by bars with distinct patterns to indicate different data points.
Figure 42. 9 Honeypots - Mean Number of Distinct Alerts
Figure 43. 9 Honeypots - Mean Packet Capture Percentage
5.4 Summary

This chapter presents and analyzes results from the experiment in Chapter 4. Results are explored with different grouping of factors to distinguish patterns and analyze the effects each factor has on the experiment. Explanations for observed patterns were also explored. Anderson-Darling tests for normality are completed on all grouping of factors to determine if a t-test was suitable for statistical testing. Because only one set was normal, a Permutation test is used to compare statistical significance with a significance level of 10%. SuA is statistically superior in all tests except for 9 honeypots overall and the TCP Connect and Aggressive scans with 9 honeypots. SAI is statistically superior to SuA in the Aggressive scan with 9 honeypots. STM is statistically superior to STI in all tests except the TCP Connect and NIDS Avoidance scans with 3 honeypots active. SuT statistically outperforms STM in all tests except the Aggressive scan with 9 honeypots active. Finally, % HHP is statistically significant in comparison to % SuP in the Aggressive Scan overall and all tests with 9 honeypots, except the NIDS Avoidance Scan. % SuP is statically significant in all NIDS Avoidance scans, the Aggressive scan with 3 honeypots, and 3 and 6 honeypots overall. Assuming the trendline continues, % HHP would perform better than % SuP with 12 and 15 honeypots in all scans.
VI. Conclusions

6.1 Introduction

In this chapter the conclusions drawn from the experimental results are discussed in Section 6.2 for each category of metrics, which includes the number of alerts, the number of distinct alerts, and the percentage of packets captured. The significance of this research is discussed in Section 6.3. Section 6.4 describes the limitations of this research. In Section 6.5, this thesis concludes with potential future work in the research field of honeypots, network intrusion detection, and the HoneyHive framework.

6.2 Research Conclusions

This research successfully creates a framework of distributed network intrusion detection IoT honeypot sensors that capture traffic, create alerts, and beacon back to a central C2 server. The first hypothesis from Chapter 1 is mostly supported with experiment results while the second is only partially supported by the trendline of experiment results:

1. The HoneyHive framework operates correctly by not alerting on routine network traffic and alerting on non-routine network traffic.

2. The HoneyHive framework detects intrusions that traditional NIDSs cannot through the use of distributed IoT honeypot sensors and packet capture aggregation.
6.2.1 Number of Alerts

The first hypothesis is a two part question, but both parts are supported by the experiment results. When No Scan is performed (the Control Group), no false positive alerts are created by HoneyHive across all runs and varying levels of honeypots. Therefore, HoneyHive operates correctly with routine network traffic.

The next part of the question requires delving into experimental results. While HoneyHive performed admirably in the majority of runs, there were 3/270 runs (runs 270, 291, and 465) that it did not create alerts (HHI) when it should have. Additionally, HoneyHive is currently using Snort for a higher level of signature matching and alert creation. However, Snort did not create alerts for 32/270 runs that it should have. This means roughly 10% of intrusions did not have successful signature matching performed on packet captures. This is either from Snort crashing, not finishing in a timely manner, simply not creating alerts, or an error in the HoneyHive framework. Although Snort is only an augmentation for the HoneyHive framework, it is used extensively in this research for generating alerts. Because of this, the HoneyHive framework, in its current configuration, alerts on non-routine traffic only around 90% of the time. For network intrusions, it is ideal for this to be as close to 100% as possible.

The second hypothesis is supported only partially by metrics and partially by trendlines. To start, the average number of alerts steadily increased as the number of honeypots increased. Even though the number of alerts in the HoneyHive framework only statistically exceeded Suricata in the Aggressive scan with 9 honeypots it was very close to Suricata in the TCP Connect scan with 9 honeypots. If this trend continued with the additions of honeypots, then it is expected that at 12 or 15 honeypots HoneyHive would surpass Suricata. Furthermore, if the experiment was modified to match an internal network scan that did not scan the DMZ, then Suricata would
not receive any non-routine traffic. This was not done in this experiment because then there would not have been any data to run statistical tests against. All this supports the hypothesis that distributed IoT honeypot sensors can detect intrusions that traditional NIDS cannot through packet capture aggregation.

6.2.2 Number of Distinct Types of Alerts

The second hypothesis is also supported by the number of distinct types of alerts from the experiment. While, HoneyHive did not ever exceed Suricata in the number of distinct alerts, Suricata was no longer statistically significant for the Aggressive scan with 9 honeypots. This once again suggests that with more honeypots HoneyHive could outperform Suricata. What is supportive though is that Snort PCAPs merged together (STM) created a larger number of distinct alerts with statistical significance for almost all tests compared with that of PCAPs parsed individually by Snort (STI). This supports the hypothesis because alerts that were not generated by analyzing each PCAP individually were generated when the PCAPs were merged and analyzed as one with statistical significance.

6.2.3 Percentage of Packets Captured

Finally, the percentage of packets captured to and from the attacker increased in all tests for the HoneyHive framework as the number of honeypots increased. The HoneyHive framework was statistically superior to Suricata with 9 honeypots active in all tests except the NIDS Avoidance scan. Assuming the trendline holds with % HHP, with more honeypots active, a higher percentage of attacker packets can be captured. Additionally, if areas of a network are scanned without a NIDS then a traditional NIDS will not see this traffic and not create alerts. Because HoneyHive is a distributed system, it can capture this traffic that a traditional NIDS cannot by
6.3 Research Significance

While there currently exist many NIDSs, none of the existing research explored in Chapter 2 is tailored to IoT honeypots. In addition, not all NIDSs alert in real time nor offer a complete view of the network with C2 capabilities. Furthermore, few offer automated distributed packet aggregation for intrusion analysis. The HoneyHive framework addresses all these shortcomings. The HoneyB Agent is also deployable to more than just IoT honeypots. It is deployable to any host with Python 2.7 installed. This makes it very versatile and offers immediate monitoring for the host with very little setup.

The HoneyHive framework offers benefits that allow the Air Force in defending the Air Force Network (AFNet) and the DoD to protect the DoD Information Network (DODIN). It can also be used for integration in CIKR-based networks since IoT devices share some similarities with ICS. Moreover, any company that works in the realm of network security could integrate HoneyHive into their existing network security. The impact of this framework is a cross-platform, standalone, NIDS / Network Monitoring solution capable of improving the rate at which network intrusions are detected. While HoneyHive will not be the solution for every network, it is another viable tool for increasing network security through intrusion detection.

6.4 Research Limitations

While the HoneyHive framework is a great success, there are several limitations in its current configuration. The IoT honeypots from Stafira used in this research
are low-interaction, web-based honeypots. Furthermore, they rely on Honeyd 1.5c and Python 2.7, both of which are at the end of their life \cite{76}. Honeyd 1.5c was last updated in 2007 \cite{10} and the current version, Honeyd 1.6d was last updated in 2013, but Stafira reported stability issues with Honeyd 1.6d \cite{6}\cite{11}. The IoT devices that Stafira’s IoT honeypots emulate are also not the most up to date or popular IoT devices in the United States \cite{6}. Furthermore, not enough honeypots were used in this experiment and only Nmap scans were performed against them.

The HoneyB Agent script is also written in Python 2.7. To ensure future usability, it would need to be rewritten in Python 3.x or node.js. While this would not be too challenging, it could take several days or weeks to ensure correct functionality after the upgrade.

Currently the HoneyHive framework relies on a NIDS (Snort) and its community of rules to perform a higher level of signature detection as opposed to simply alerting based on interactions with honeypots. If the HoneyHive framework possessed its own self-contained, sophisticated combination of heuristics and signature matching then it would perform even more effectively and be easier to deploy to networks. Despite these limitations, the HoneyHive framework has great potential as a tool for network intrusion detection. In this experiment, Snort should have been used for both the HoneyHive framework and the DMZ NIDS listening on the SPAN port as it appears that Snort and Suricata do not create the same alerts given the same rule-set and network traffic.

The test network scans were performed on was of limited size whereas enterprise networks have hundreds if not thousands of devices. Additionally, all Honeypots were on the same network but enterprise networks often have different sub-networks comprising their internal network.
6.5 Future Work

One area of future work that would benefit the research of honeypots, network intrusion detection, and the HoneyHive framework is developing high-interaction honeypots in Docker (with secure configurations). Doing so would create more realistic honeypots and extend the capabilities of the HoneyHive framework. Memory dumps of an attacker’s interaction with honeypots could be performed to capture TTPs and help identify them based on their tradecraft. In addition, more modern and popular versions of IoT devices could be developed as honeypots. This would allow HoneyHive to blend into more networks and not be as identifiable. The DB currently stores alert information, captured traffic, and is designed to store captured passwords, binaries, and memory dumps in the event that future work in HoneyHive is able to capture these.

A major component of future work for the HoneyHive framework includes the development of GUI to help network operators quickly identify and respond to intrusions. The proposed GUI design is shown in Figure 44 and can display alerts and honeypot interactions to network operators. The left side of the GUI displays the health status of the monitored honeypots, green representing no alerts, yellow scan alerts, red exploit detected, and black meaning unreachable. Honeypots are nested under their Honeyd controller, and the Honeyd controller takes on the most severe status of the honeypots under it.

The center displays a network map with the same status indicators shown on the left side. Notice that the device 192.168.45.42 is not a monitored honeypot, yet it is still flagged red. This is because an internal scan originating from the device has been detected, so the device itself is inferred to be compromised.

Individual alerts are viewable on the right side, and the actual packet capture is available for download by clicking the green down arrow. The two alerts currently
displayed show the TITAThink and ezOutlet2 honeypots being scanned internally by 192.168.45.42, which matches the color coding of what is shown by the honeypot status (left), and the network map (middle).
Figure 44. Proposed HoneyHive GUI
Other future work includes hashing and distributing signatures across a multi-server HoneyHive framework for faster detection of binaries, exploits, and TTPs. The HoneyB Agent script could also be improved to be a sophisticated and self-contained alert engine. In addition, the HoneyHive framework currently does not implement encryption of network traffic and device authentication, as it was not required for testing, but must before deployment. All traffic sent must be encrypted before network deployment to thwart sniffing. Symmetric encryption is the proposed implementation method of encryption, as opposed to asymmetric encryption, because distributing a single shared passphrase is easier than setting up a Certificate Authority (CA), generating two keys per device, and then registering all keys with the CA. Device authentication is also important to ensure the authenticity of traffic, security of the framework, and prevent erroneous alerts from overloading network operators.

If all these areas of future work are implemented then HoneyHive could be used for more than just intrusion detection. It could be used for cyber deception much like Cymmetria’s MazeRunner framework [29].
Appendix A. HoneyHive Framework

main.js (C2 Server)

```javascript
var cyan = '\x1b[36m%s\x1b[0m';
var green = '\x1b[32m%s\x1b[0m';
var blue = '\x1b[34m%s\x1b[0m';
var yellow = '\x1b[33m%s\x1b[0m';
var red = '\x1b[31m%s\x1b[0m';
var magenta = '\x1b[35m%s\x1b[0m';

var net = require('net');
var fs = require('fs');
var buffer = require('buffer');
var path = require('path');
var fork = require('child_process').fork;
const sqlite3 = require('sqlite3').verbose();
var exec = require('child_process').exec;

var encrypt_decrypt = require('./encrypt_decrypt.js');
var crypto = require('crypto');
var password = 'honeyhive';
var algorithm = 'aes-256-cbc';

var HOST = '0.0.0.0';
var PORT = 9830;
var resetCounter = 0;
var honeydIP = [];
var honeyPots = [];
var completeTransfers = [];
var alerts = [];
var srcIPs = {};
```
var dstIPs = {};
var srcPrts = {};
var dstPrts = {};
var percentage = 0.35;

var numInteractions = 0
var numSnortAlerts = 0;
var numPackets = 0;
var numSuricataAlerts = 0;
var numSuricataTypes = 0;
var suricataPackets = 0;
var numSnortAlerts_Merged = 0;
var numSnortTypes = 0;
var numSnortTypesMerged = 0;
var snortICount = 0;
var snortMCount = 0;

// Check for DB and create it if it doesn’t exists
// spawns a child process to check / create DB
const database_creator = path.resolve("database_creator.js");

console.log(green, 'Checking Database File
');

const params = [];
const options = {
    stdio: [ 'pipe', 'pipe', 'pipe', 'ipc' ]
};

const database_child = fork(database_creator, params, options);

database_child.on('message', message =>{
    console.log(green, 'message from Database Child:', message);
function execute(command, callback){
    exec(command, function(error, stdout, stderr){
        callback(stdout);
    });
};

function countPackets(output)
{
    numPackets += parseInt(output.split('Number of packets:')[1].trim())
    console.log(green, "Num Packets: " + numPackets);
}

// Assumptions: Honeypots are not emitting malicious traffic / haven't been compromised
function alertAnalyzer()
{
    var hpThreshold = (Object.keys(dstIPs).length / (honeyPots.length))
};

if(hpThreshold > percentage)
{
    createAlert('Alert: Multiple Honeypot Interaction Detected!')
}
function createAlert(msg) {
  console.log(red, msg)
}

function parsePCAP(filename) {
  /*
  File name: C:\Snort\log\192.168.1.152_2019-10-14_0939.pcap
  Number of packets: 1894
  */
  var cmd = "C:\Program Files\Wireshark\capinfos.exe" -c "C:\Snort\log\192.168.1.152_2019-10-14_0939\192.168.1.152_2019-10-14_0939.pcap"
  execute(cmd, countPackets);
}

function snortMerge(mergeOutput) {
  console.log('Starting Snort Parser on Merged File\n');
  const snort_parser_merge = path.resolve("snort_parser.js");
  const paramsMerge = [resetCounter, 0];
  const optionsMerge = {
    stdio: [ 'pipe', 'pipe', 'pipe', 'ipc' ]
  };
  const snort_child_merge = fork(snort_parser_merge, paramsMerge, optionsMerge);
snort_child_merge.on('message', message =>{
    console.log(red, 'message from Snort Child Merged:', message);
    snortMCount +=1;
    message.forEach(function(alert)
    {
        if(alert.Count != undefined)
        {
            numSnortAlerts_Merged += alert.Count;
        }
        numSnortTypesMerged +=1;
    });
});

snort_child_merge.on('exit', (code) => {
    snortMCount +=1;
    console.log("Snort M Child Exited");
});

// looks at interfaces to automatically grab and bind on an IP
var os = require('os');
var ifaces = os.networkInterfaces();
var serverIP;

Object.keys(ifaces).forEach(function (ifname) {
    var alias = 0;

    ifaces[ifname].forEach(function (iface) {
        if ('IPv4' !== iface.family || iface.internal !== false ||
            ifname.includes('VMware')) {
            }
// skip over internal (i.e. 127.0.0.1) and non-ipv4 addresses
return;

if (alias >= 1) {
    // this single interface has multiple ipv4 addresses
    console.log(ifname + ':' + alias, iface.address);
} else {
    // this interface has only one ipv4 adress
    console.log(ifname, iface.address);
    serverIP = iface.address;
}
++alias;
});

console.log(cyan, "Server IP: " +serverIP);

// Create a server instance, and chain the listen function to it
// The function passed to net.createServer() becomes the event
// handler for the 'connection' event
// The sock object the callback function receives UNIQUE for each
// connection
var server = net.createServer(function(sock) {

    // Add a 'data' event handler to this instance of socket
    sock.on('data', function(data) {
        var JSONData = JSON.parse(data);

        /*
        header = {"Honeyd": honeydIP,
        "Honeypots": honeypots
        */

148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
"MSG": 'AUTHENTICATE'}

*/
if(JSONData.MSG == 'AUTHENTICATE')
{
    if(!honeydIP.includes(JSONData.Honeyd))
    {
        honeydIP.push(JSONData.Honeyd);
    }

    JSONData.Honeypots.forEach(function(pot)
    {
        if(!honeyPots.includes(pot))
        {
            honeyPots.push(pot);
        }
    });
}

console.log(green, "Connected Honeypots");
console.log(green, honeyPots);

// receive suricata alert count
else if(JSONData.MSG == 'SURICATA')
{
    numSuricataAlerts += JSONData.NumAlerts;
    numSuricataTypes += JSONData.NumTypes;
    suricataPackets = JSONData.NumPackets;
    console.log(green, 'Suricata Alerts Received: ' + numSuricataAlerts + ', ' + numSuricataTypes + ', ' + suricataPackets);
}
// reboot machine for fresh stable state
// C2 server should be relaunched automatically at startup
else if(JSONData.MSG == 'REBOOT') {
    // deletes all snort log files and then
    // reboots when the cmd is finished
    execute("del C:\Snort\log\* /S /F /Q",
        function(output){
            execute("shutdown /g /f /t 0", function(){
                });
        });
}

else if(JSONData.MSG == 'SNORT') {
    console.log("SNORT Command received, parsing PCAPS\n");
    // parse unscanned PCAPS
    if(!(completeTransfers === undefined || completeTransfers.length == 0)) {
        
        // mergecap -w outfile.pcapng dhcp-capture.pcapng imap-1.pcapng
        // have to add all their dirs in front of the filename too
        // then run through Snort
        var cmd = "C:\Program Files\Wireshark\mergecap" -F pcap -w "C:\Snort\log\merged\"+resetCounter\".pcap";
        completeTransfers.forEach(function(filename) {
            parsePCAP(filename);
            cmd = cmd + \" + "C:\Snort\log\"+filename;
        });
    }
console.log('Starting Snort Parser individually\n');
const snort_parser = path.resolve("snort_parser.js");

const params = [resetCounter, 1];
const options = {
  stdio: [ 'pipe', 'pipe', 'pipe', 'ipc' ]
};

const snort_child = fork(snort_parser, params, options);

snort_child.on('message', message =>{
    console.log(red, 'message from Snort Child individually:', message);
    message.forEach(function(alert)
    {
        if(alert.Count !== undefined)
        {
            numSnortAlerts += alert.Count;
        }
        numSnortTypes +=1;
    });
});

snort_child.on('exit', (code) => {
    snortICount +=1;
    console.log("Snort I Child Exitd");
});

if(completeTransfers.length > 1)
{
    //function noop(){
}
execute(cmd, snortMerge);

}
}
}

else if(JSONData.MSG == 'RESET')
{
    // need to add a wait for all the snort processes to end
    sock.write(JSON.stringify({Interactions: numInteractions,
                   SnortICount: snortICount, Snort: numSnortAlerts, SnortTypes:
                   numSnortTypes, SnortMCount: snortMCount, SnortMerged:
                   numSnortAlerts_Merged, SnortTypesMerged:numSnortTypesMerged,
                   Packets: numPackets, Suricata: numSuricataAlerts, SuricataTypes:
                   numSuricataTypes, suricataPackets: suricataPackets, numHoneypots:
                   honeyPots.length, numPCAPs: completeTransfers.length}));

    honeydIP = [];
    honeyPots = [];
    completeTransfers = [];
    alerts = [];
    srcIPs = {};
    dstIPs = {};
    srcPrts = {};
    dstPrts = {};

    snortICount = 0;
    snortMCount = 0;
    numInteractions = 0
    numSnortAlerts = 0;
    numPackets = 0;
    numSuricataAlerts = 0;
    numSuricataTypes = 0;
suricataPackets = 0;
numSnortAlerts_Merged = 0;
numSnortTypes = 0;
numSnortTypesMerged = 0;
resetCounter += 1;

fs.writeFile("C:\Snort\log\pcaps.txt", '');
    console.log('Snort PCAP File cleared
');
    console.log(yellow, "Reset Received
");
}

/*
header = {
    "Time": time.strftime("%Y-%m-%d_%H%M"),
    "Honeyd": honeydIP,
    "IP": '192.168.72.150',
    "MSG": 'HEARTBEAT'}
*/
else if(JSONData.MSG == 'HEARTBEAT')
{
    console.log(green, "Honeypot Heartbeat - Time: " + JSONData .Time + " Honeyd: " + JSONData.honeydIP + " Honeypot IP: " + JSONData.IP);
}

/*
header = {
    "Time": time.strftime("%Y-%m-%d_%H%M"),
    "TransLayer": transLayer,
    "IP_SRC": pclt_src,
    "IP_DST": pclt_dst,
    "SPORT": sport,
    "DPORT": dport,
*/
else if (JSONData.MSG == 'ALERT') {
    numInteractions +=1;
    console.log(yellow, "Honeypot interaction detected \n\tTime : "+JSONData.Time +
    "\n\tTransport Protocol: "+JSONData.TransLayer +
    "\n\tIP SRC: "+JSONData.IP_SRC +
    "\n\tSRC Port: "+JSONData.SPORT +
    "\n\tIP DST: "+JSONData.IP_DST +
    "\n\tDST Port: "+JSONData.DPORT);

    if (JSONData.IP_SRC in srcIPs) {
        srcIPs[JSONData.IP_SRC] +=1;
    }
    else {
        srcIPs[JSONData.IP_SRC] = 1;
    }

    // --------------------------
    if (JSONData.SPORT in srcPrts) {
        srcPrts[JSONData.SPORT] +=1;
    }
    else {
        srcPrts[JSONData.SPORT] = 1;
    }
if (JSONData.IP_DST in dstIPs)
{
    dstIPs[JSONData.IP_DST] +=1;
}

else {
    dstIPs[JSONData.IP_DST] = 1;
}

if (JSONData.DPORT in dstPrts)
{
    dstPrts[JSONData.DPORT] +=1;
}

else {
    dstPrts[JSONData.DPORT] = 1;
}

alertAnalyzer();

// spawns a child process to check / create DB
const database_inserter = path.resolve("database_inserter.js");

console.log(green, 'Adding to Database\n');

// would need to re-JSON-ify JSONData, so just sending the
// original data
const params = [data];
const options = {
    stdio: [ 'pipe', 'pipe', 'pipe', 'ipc' ]
};

const inserter_child = fork(database_inserter, params, options);

inserter_child.on('message', message =>{
    console.log(green, 'message from Inserter Child:', message);
});

// client is sending PCAP, open a new port for transfer state
// open a server / port for the file transfer to keep command
// data and binary data separate
else if(JSONData.MSG == 'PCAP')
{
    //console.log(green, JSONData.Filename + '
' + JSONData.File_Size + '
' + JSONData.MD5);
    //var callbackPort = JSONData.listenPort
    const transfer_server = path.resolve("transfer_server.js");
    const params = [JSONData.Filename, JSONData.File_Size, JSONData.MD5, JSONData.Port];

    const options = {
        stdio: [ 'pipe', 'pipe', 'pipe', 'ipc' ]
    };
const transferChild = fork(transfer_server, params, options);

transferChild.on('message', message =>{
    console.log(green, 'message from transfer child:', message + '
');
    if(message == 'Download Complete')
    {
        fs.appendFile("C:\Snort\log\pcaps.txt", "C:\Snort\log\"+JSONData.Filename+’\n’, function (err) {
            if (err) throw err;
            console.log('PCAP File added to Snort File list \n');
        });
        completeTransfers.push(JSONData.Filename);
    }
});

// end of PCAP

// end of on data

// Add a 'close' event handler to this instance of socket
sock.on('close', function(data) {
    console.log(yellow, 'CLOSED: ' + sock.remoteAddress + ' ' + sock.remotePort);
});

server.once('error', function(err) {
if (err.code === 'EADDRINUSE') {
    console.log('Port Already in Use!:
');
    process.exit()
}

server.listen(PORT, HOST);

console.log(cyan, 'Server bound on ' + HOST +':' + PORT);
```javascript
const crypto = require ('crypto');
const downloadMD5 = crypto.createHash ('md5');
var net = require ('net');
var fs = require ('fs');
var buffer = require ('buffer');

var HOST = '0.0.0.0';

var params = process.argv;
var fileName = String (params.slice(2, 3));
var dir = "C:\\Snort\\log"; // \\
var file_name = fileName.split ('.pcap') [0];
var file_size = params.slice (3, 4);
var md5 = params.slice (4, 5);
var PORT = parseInt (params.slice (5, 6));
var downloadedBytes = 0;

// test that args were being received correctly
//process.send("arguments: " + params.slice(2, 3) + "\n" + params.
//slice(3, 4) + "\n" + params.slice(4, 5));
//process.send("filename: " + params.slice(2, 3));

var fileStream = fs.createWriteStream (dir + '\\' + fileName);

var server = net.createServer (function (sock) {
    process.send ('Downloading PCAP: \n');
    downloadedBytes = sock.on ('data', function(data) {
        //fs.access(file, fs.constants.W_OK, (err) => {
```
// process.send('${file} ${err ? 'is not writable' : 'is writable'}');

downloadedBytes += data.length;
FileStream.write(data);

sock.on('close', function(data) {
    process.send('Download Complete');
    // process.send("Downloaded: "+ downloadedBytes + " bytes");
    server.close();
    // md5Verification();
    process.exit()
});

server.once('error', function(err) {
    if (err.code === 'EADDRINUSE') {
        process.send('Port Already in Use!: 
');
        process.exit()
    }
});

server.listen(PORT, HOST);
process.send('Transfer Server bound on ' + HOST + ':' + PORT);
snort_parser.js (Snort Log Parser)

// Run Snort on a PCAP

// Parse Snort alert log, combine like alerts,
// and other metadata in JSON format (src IP, dst IP, port range)
// keep count of occurrences
// dstIP should be the same since the pcaps are transferred as
// individually for each IP

// Send back results to main.js for compilation and monitoring
// which will have some threshold for alerting if so much traffic is
// seen
// sent to one host, or a smaller amount of traffic sent to multiple
// hosts

// Snort Alert Example
/*
[**] [1:10000005:2] NMAP TCP Scan [**]
[Priority: 0]
TCP TTL: 128 TOS: 0x0 ID: 28818 IpLen: 20 DgmLen: 52 DF
***** S* Seq: 0x5F62CB19 Ack: 0x0 Win: 0xFAF0 TcpLen: 32
TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK
*/

// Snort Alert Example
/*
[**] [1:2009582:3] ET SCAN NMAP -sS window 1024 [**]
[Classification: Attempted Information Leak] [Priority: 2]
TCP TTL: 56 TOS: 0x0 ID: 13589 IpLen: 20 DgmLen: 44
*/
var fs = require('fs');
const readline = require('readline');
var exec = require('child_process').exec;
var params = process.argv;

var iteration = parseInt(params.slice(2, 3));
var dir = "C:\\Snort\\log\\";
var individually = parseInt(params.slice(3, 4));

var lineCounter = 0;
const numLinesSnortAlert = 8;
var snortLogResults = [];
var found = false;
var indx = 0;
var lastAlert = "";

if (!fs.existsSync(dir + ' individually' + iteration))
{
  fs.mkdirSync(dir + ' individually' + iteration);
  fs.mkdirSync(dir + ' merged' + iteration);
}

function execute(command, callback){
  exec(command, function(error, stdout, stderr){
    //process.send(stdout);
    //process.send(stderr);
    callback(stdout); });}
var noop = function () {}; // do nothing.

function snortLog(output) {
    process.send(output);
}

async function processLineByLine() {
    const fileStream = fs.createReadStream(dir + '\alert.ids');

    const rl = readline.createInterface({
        input: fileStream,
        crlfDelay: Infinity
    });

    // Note: we use the crlfDelay option to recognize all instances of CR LF
    // ('\r\n') in input.txt as a single line break.

    for await (const line of rl) {
        try {
            // Type of Alert:[**] [1:10000005:2] NMAP TCP Scan [**]
            if (lineCounter % numLinesSnortAlert == 0) {
                var regex = /\[\] \[(0-9A-z)\]+\[/g;
                var alert = line.match(regex);
                var alertType = alert[1].replace(']', ' '); alertType = alertType.replace('[', ' '); alertType = alertType.trim();

                // checks to see if alert is same as prev, so we can skip
if(alertType == lastAlert)
{
    snortLogResults[indx].Count +=1;
}

else{
    found = false;
    for(var i = 0; i < snortLogResults.length; i++)
    {
        if (snortLogResults[i].Type == alertType)
        {
            found = true;
            indx = i;
            snortLogResults[i].Count +=1;
            break;
        }
    }
}

if (found == false) {
    snortLogResults.push({Type: alertType, Count: 1,
        StartDate: "", EndDate: "", StartTime: "", EndTime: "", Src: [],
        Dst:[]});
    found = true;
    indx = snortLogResults.length-1;
    lastAlert = alertType;
}

// Priority:[Priority: 0]
//else if (lineCounter%numLinesSnortAlert == 1)
121  //{
122   // Each line in input.txt will be successively available
123   // here as ‘line’.
124   // process.send(‘Line from file: ${line}’);
125  //}
126  
127  // Timestamp and Src -> Dst ’10/01-16:22:13.304233
128  192.168.1.11:61273 -> 192.168.1.150:80’
129  
130  else if (lineCounter % numLinesSnortAlert == 2)
131  {
132      var dateSrcDst = line.split(‘ ’);
133      var dateTime = dateSrcDst[0].split(‘-’);
134      var date = dateTime[0];
135      var time = dateTime[1].split(‘.’)[0];
136  
137      var src = dateSrcDst[1];
138      var srcSplit = src.split(‘:’);
139      var srcIP = srcSplit[0];
140      var srcPrt = parseInt(srcSplit[1], 10);
141  
142      var dst = dateSrcDst[3];
143      var dstSplit = dst.split(‘:’);
144      var dstIP = dstSplit[0];
145      var dstPrt = parseInt(dstSplit[1], 10);
146  
147  // first entry
148  if(snortLogResults[indx].StartDate == "")
149  {
150      snortLogResults[indx].StartDate = date;
151      snortLogResults[indx].StartTime = time;
152  }
else {
    snortLogResults[indx].EndDate = date;
    snortLogResults[indx].EndTime = time;
}

if (!snortLogResults[indx].Src.includes(src))
{
    snortLogResults[indx].Src.push(src);
}

if (!snortLogResults[indx].Dst.includes(dst))
{
    snortLogResults[indx].Dst.push(dst);
}

// process.send(snortLogResults);

// TCP TTL:128 TOS:0x0 ID:28818 IpLen:20 DgmLen:52 DF
// else if (lineCounter%numLinesSnortAlert == 3)
//{
//    // Each line in input.txt will be successively available here as 'line'.
//    // process.send('Line from file: ${line}');
//}
//}

// *****S* Seq: 0x5F6CB19 Ack: 0x0 Win: 0xFAF0 TcpLen: 32
// else if (lineCounter%numLinesSnortAlert == 4)
//{
//    // Each line in input.txt will be successively available here as 'line'.
//    // process.send('Line from file: ${line}');
181  //}
182
183  // TCP Options (6) => MSS: 1460 NOP WS: 8 NOP NOP SackOK
184  else if (lineCounter%numLinesSnortAlert == 5)
185  {
186       // has 5 lines for alert instead of 6 or 7
187       // modify by 2 to catch up
188       if(!(line.includes('TCP Options')))
189       {
190           lineCounter +=2;
191       }
192  }
193
194  else if (lineCounter%numLinesSnortAlert == 6)
195  {
196       // has 6 lines for alert instead of 7
197       // modify by 1 to catch up
198       if(!(line.includes('Xref')))
199       {
200           lineCounter +=1;
201       }
202  }
203
204  // blank line between alerts
205  //else if (lineCounter%numLinesSnortAlert == 7)
206  //{}
207
208  }//
209  lineCounter +=1;
210 }
211 process.send(snortLogResults);
212 }
catch (err)
{
    process.send(err)
}

//process.exit()

var cmd = '"C:\Snort\bin\snort.exe" -c "C:\Snort\etc\snort.conf" ';
if (individually == 1)
{
    dir = dir + 'individually' + iteration;
    cmd = cmd + --pcap-file "C:\Snort\log\pcaps.txt" --pcap-reset -l ' + dir;
}
else {
    dir = dir + 'merged' + iteration;
    cmd = cmd + -r "C:\Snort\log\merged' + iteration + '.pcap" -l ' +
    dir;
}

execute(cmd, processLineByLine);
import subprocess
import thread
import socket
import json
import time
import os
import base64
from datetime import datetime, date, time, timedelta

from scapy.all import *

# color honey yellow is #a98307
port = 9830
transferPort = 9831
honeyHiveIP = '192.168.1.233'
honeydIP = '192.168.1.154'
honeypots = ["192.168.1.150", "192.168.1.151", "192.168.1.152"]
connections = {}
timeOutSeconds = 9000
sessionTimeout = timedelta(seconds=timeOutSeconds)
autoTransferTimeout = timedelta(seconds=timeOutSeconds)
connections_Lock = thread.allocate_lock()
honeyd = None
devnull = open(os.devnull, 'wb')
alertMode = False

def main():
    global honeyd

    pcap_monitor_id = thread.start_new_thread(pcapMonitor, ()
    #heartbeat_id = thread.start_new_thread(heartbeat, ()

honeyB_Agent.py (HoneyB Agent)
# I want this in my main console output
# once run, it causes blocking
print "Scapy Packet Sniffer Engaged"

sniff(iface="eth0", prn=processPacket, store=0)

def pcapMonitor():
    while True:
        connections_Lock.acquire()
        uct = datetime.utcnow()
        remove = []
        for ip in connections:
            if (uct - connections[ip].get("time") >=
                autoTransferTimeout):
                transferFile(ip, connections[ip].get('filename'))
                remove.append(ip)
                print "Automatic Transfer"
        for i in remove:
            connections.pop(i)
        connections_Lock.release()
        floatSeconds = timeOutSeconds*1.0
        time.sleep(floatSeconds)

    # method called when a packet is received
    # parses the packet to identify the dest ip, port, ect.
    # will implement scan recognition and attacker pivoting
    def processPacket(packet):

        # automatically named files based on ip, date, and time
        now = datetime.now()
        uct = datetime.utcnow()
# ignore traffic not to honeypots

```python
if packet.haslayer(IP) and (not packet[IP].dst in honeypots) and (not packet[IP].src in honeypots) and (not packet[IP].dst == honeydIP) and (not packet[IP].src == honeydIP):
    None
```

# ignore 9830 and 9831 because they are packets we are sending

```python
elif packet.haslayer(TCP) and (packet[TCP].dport == port or packet[TCP].dport == transferPort or packet[TCP].sport == port or packet[TCP].sport == transferPort):
    None
```

# ignores some Ubuntu traffic that is not being filtered out my /etc/hosts

```python
elif packet.haslayer(UDP) and (packet[UDP].sport == 68 or packet[UDP].dport == 68 or packet[UDP].dport == 5353 or packet[UDP].dport == 53 or packet[UDP].sport == 53):
    None
```

```python
elif packet.haslayer(UDP) and (packet[UDP].dport == 9830):
    print "Command Received"
    print now.strftime("%Y-%m-%d_%H%M")
    runCommand(packet[UDP].load.strip('
'))
```

# ensures this packet has an IP

```python
# log non 9830 and 9831 traffic from c2 server1
elif packet.haslayer(IP) and alertMode:
    pckt_src=packet[IP].src
    pckt_dst=packet[IP].dst
    pckt_ttl=packet[IP].ttl
```
# if the honeypot doesn't already have an ongoing pcap log file
# and the time since last packet is less than 5 minutes
connections_Lock.acquire()

if any(x in connections for x in [pckt_dst, pckt_src]):
    ip = ""
    filename = ""

    # determine which IP was in the connections list
    if pckt_dst in connections:
        ip = pckt_dst
    else:
        ip = pckt_src

    # time difference is less than than session timeout
    # therefore, keep logging to file
    # and update its timestamp to the new time
    # new time - oldtime < allowed time
    if (uct - connections[ip].get("time") < sessionTimeout):
        filename = connections[ip].get("filename")
        connections[ip].update({"time": uct})

    # else there was a timeout so need a new file
    # there was a connection at one point so the IP
    # is in our honeypot list, no need to check again
    # update time of connection to now
else:
    print "Transferring PCAP"
```python
# if packet.haslayer(TCP):
    # print "Packet: Source IP %s, Port %s\n Dest IP %s, Port %s" % (pckt_src, packet[TCP].sport, pckt_dst, packet[TCP].dport)
# else:
    # print "Packet: Source IP %s\n Dest IP %s" % (pckt_src, pckt_dst)
transferFile(ip, connections[ip].get('filename'))

# create the new pcap file
filename = ip + "_" + now.strftime("%Y-%m-%d_%H%M") + "_pcap"
# print "New filename: " + filename
# print connections[ip]
connections[ip].update({"time": uct})
connections[ip].update({"filename": filename})
# print connections[ip]
alert(packet, now)

# appends packet to output file
wrpcap(filename, packet, append=True)

# ips not in connected list, but this may be first interaction
# with the honeypots. Want to assume first captured packet
# be sent to honeypot, but compromised hp could beacon out if
# hardware or software make it
elif any(x in honeypots for x in [pckt_dst, pckt_src]):
ip = ""

# determine which ip was in the connections list
```
if pkt_dst in honeypots:
    ip = pkt_dst
else:
    ip = pkt_src

# create the new pcap file
filename = ip + "_" + now.strftime("%Y-%m-%d_%H%M") + ".pcap"

# print "filename: " + filename
connections[ip] = {"filename": filename, "time": uct}
alert(packet, now)

# appends packet to output file
wrpcap(filename, packet, append=True)
connections_Lock.release()

# receive and execute commands
def runCommand(msg):
    global connections
    global honeyd
    global alertMode

    print msg

    # transfer all current pcaps
    if(msg == "TRANSFER"):  
        connections_Lock.acquire()  
        for ip in connections:
            transferFile(ip, connections[ip].get('filename'))
        # return all values to 0
        connections = {}
connections_Lock.release()

elif(msg == "RESET"):
    # transfer all current pcaps
    connections_Lock.acquire()
    for ip in connections:
        transferFile(ip, connections[ip].get('filename'))
    # return all values to 0
    connections = {}
    connections_Lock.release()

    # kill all honeyd
    if(honeyd != None):
        subprocess.Popen(['sudo killall honeyd'], stdin=None,
                          stdout=devnull, stderr=devnull, shell=True)

        honeyd = None
        alertMode = False

    # stops program completely
elif(msg == "KILL"):
    # kill all honeyd
    if(honeyd != None):
        subprocess.Popen(['sudo killall honeyd'], stdin=None,
                          stdout=devnull, stderr=devnull, shell=True)

        # kill this script
        subprocess.Popen(['sudo killall python'], stdin=None, stdout=devnull,
                          stderr=devnull, shell=True)

    # starts just honeyd
elif(msg == "START"):
# relaunch honeyd
if (honeyd == None):
    honeyd = subprocess.Popen(['sudo /home/edge/Desktop/honeyhive/scripts/startHoney.sh'], stdin=None, stdout=devnull, stderr=devnull, shell=True)
    alertMode = True

# authenticate
authenticate()

# stops just honeyd
elif (msg == 'STOP'):
    # kill all honeyd
    if (honeyd != None):
        subprocess.Popen(['sudo killall honeyd'], stdin=None, stdout=devnull, stderr=devnull, shell=True)
    honeyd = None
    alertMode = False

elif (msg == 'REBOOT'):
    subprocess.call(['sudo', 'reboot'])

def heartbeat():
    # header information to send server
    # python dictionary
    header = {'Time': time.strftime('%Y-%m-%d_%H%M'),
              'Honeyd': honeydIP,
              'IP': '192.168.72.150',
              'MSG': 'HEARTBEAT'}

    # converts dictionary to json format
    jsonHeader = json.dumps(header)
try:
    # attempts to connect to the server and tell it to open a
    # transfer port
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = (honeyHiveIP, port)
    sock.connect(server_address)

    sock.sendall(jsonHeader)

except socket.error, e:
    print "Error creating Heartbeat socket: %s" %e

# clean-up actions that must be executed under all circumstances
finally:
    sock.close()

def authenticate():

    # header information to send server
    # python dictionary
    header = {"Honeyd": honeydIP,
              "Honeypots": honeypots,
              "MSG": 'AUTHENTICATE'}

    # converts dictionary to json format
    jsonHeader = json.dumps(header)

try:
    # attempts to connect to the server and tell it to open a
    # transfer port
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

178
server_address = (honeyHiveIP, port)
sock.connect(server_address)
sock.sendall(jsonHeader)

except socket.error, e:
    print "Error creating Authenticate socket: %s" %e

# clean-up actions that must be executed under all circumstances
finally:
    sock.close()

def alert(packet, time):
    sport = 0
dport = 0
transLayer = 'TCP'
if packet.haslayer(TCP):
    sport=packet[TCP].sport
dport=packet[TCP].dport

elif packet.haslayer(UDP):
    sport=packet[UDP].sport
dport=packet[UDP].dport
    transLayer = 'UDP'

elif packet.haslayer(ICMP):
    transLayer = 'ICMP'

else:
    transLayer = 'Other'
pckt_src = packet[IP].src
pckt_dst = packet[IP].dst
pckt_ttl = packet[IP].ttl

# header information to send server
# python dictionary
header = {"Time": time.strftime("%Y-%m-%d_%H%M"),
    "TransLayer": transLayer,
    "IP_SRC": pckt_src,
    "IP_DST": pckt_dst,
    "SPORT": sport,
    "DPORT": dport,
    "MSG": 'ALERT'}

# converts dictionary to json format
jsonHeader = json.dumps(header)

try:
    # attempts to connect to the server and tell it to open a transfer port
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = (honeyHiveIP, port)
    sock.connect(server_address)
    sock.sendall(jsonHeader)

except socket.error, e:
    print "Error creating Alert socket: %s" %e

# clean-up actions that must be executed under all circumstances
finally:
    sock.close()
# sends transfer file command to the server

def transferFile(ip, filename):

    # run command and return output
    hashPcap = subprocess.check_output(["md5sum", filename])

    # header information to send server
    # python dictionary
    header = {"Honeyd": honeydIP,
              "IP": ip,
              "Filename": filename,
              "File_Size": os.path.getsize(filename),
              "MD5": hashPcap.split(' ')[0],
              "MSG": 'PCAP',
              "Port": transferPort}

    # converts dictionary to json format
    jsonHeader = json.dumps(header)

    try:
        # attempts to connect to the server and tell it to open a transfer port
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_address = (honeyHiveIP, port)
        sock.connect(server_address)

        sock.sendall(jsonHeader)

    except socket.error, e:
        print "Error creating Transfer Server socket: %s" %e
# clean-up actions that must be executed under all circumstances

```python
finally:
    sock.close()
```

# allows time for the C2 server to receive and start the transfer server

```python
# on the requested port
time.sleep(3)
```

```python
try:
    transSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    transfer_address = (honeyHiveIP, transferPort)
    transSock.connect(transfer_address)

    # reads and sends the file to the server
    f = open(filename, "rb")
    l = f.read(1024)

    while (l):
        # encrypt data before sending
        transSock.send(l)
        l = f.read(1024)

    f.close()
    subprocess.call([\'rm\', \'-f\', filename])
```

```python
except socket.error, e:
    print "Error creating Transfer File socket: %s" %e
```

# clean-up actions that must be executed under all circumstances

```python
finally:
    transSock.close()
```
389

390 main()
Appendix B. Honeyd Configuration File

```plaintext
iotHoneyd.conf

create default

set default default tcp action block
set default default udp action block
set default default icmp action block

create titacamera

set titacamera personality "Linux 2.3.28-33"
set titacamera default tcp action reset
add titacamera tcp port 80 "TitaCamera/camera_web.sh"
add titacamera tcp port 554 open
add titacamera tcp port 49152 open
add titacamera udp port 443 filtered
add titacamera udp port 990 filtered
add titacamera udp port 1900 filtered
add titacamera udp port 1901 filtered
add titacamera udp port 3702 open
add titacamera udp port 16896 filtered
add titacamera udp port 18676 filtered
add titacamera udp port 19956 filtered
add titacamera udp port 22986 filtered
add titacamera udp port 30697 filtered
add titacamera udp port 32772 filtered
add titacamera udp port 32777 filtered

create proliphixthermostat

set proliphixthermostat personality "D-Link Print Server"
```

184
set proliphixthermostat default tcp action reset

# Regular Thermostat Interface when Internet is available
# add proliphixthermostat tcp port 80
    "ProliphixThermostat/thermostat_web.sh"

# Thermostat Interface when no internet is available
add proliphixthermostat tcp port 80
    "ProliphixThermostat/thermostat_web_nointernet.sh"

create ezoutlet

set ezoutlet personality "IBM OS/2 Warp 4.0"
set ezoutlet default tcp action reset
add ezoutlet tcp port 80 "ezOutlet/outlet_web.sh"

set titacamera ethernet "7C:DD:90:00:22:82"
bind 192.168.45.150 titacamera
# dhcp titacamera on eth0
set proliphixthermostat ethernet "00:11:49:00:62:46"
bind 192.168.45.151 proliphixthermostat
# dhcp proliphixthermostat on eth0
set ezoutlet ethernet "00:03:EA:0E:11:67"
bind 192.168.45.152 ezoutlet
# dhcp ezoutlet on eth0
Appendix C. suricataConnect.py

suricataConnect.py

```python
import subprocess
import socket
import json
import os

# color honey yellow is #a98307
port = 9830
honeyHiveIP = '192.168.1.233'
suricataIP = '192.168.1.231'
attacker = '192.168.1.230'
suricataRunning = False

devnull = open(os.devnull, 'wb')

def main():
    print "Scapy Packet Sniffer Engaged"
    sniff(iface="eth0", prn=processPacket, store=0)

def processPacket(packet):
    global totalPackets
    if packet.haslayer(IP) and (packet[IP].src == attacker or packet[IP].dst == attacker):
        totalPackets += 1
    # running on mirrored port so make sure it's for this IP

```

186
if packet.haslayer(UDP) and packet[UDP].dport == 9830 and packet[IP].dst == suricataIP:
    print "Command Received"
    runCommand(packet[UDP].load.strip(\'\n\'))

# receive and execute commands
def runCommand(msg):
    global suricataRunning
    global packetRXStart, packetTXStart
    global totalPackets

    logGrep = subprocess.Popen(['sudo', 'grep', '-i', 'scan', '/var/log/suricata/fast.log'], stdout=subprocess.PIPE)

    wc = subprocess.Popen(['wc', '-l'], stdin=logGrep.stdout, stdout=subprocess.PIPE)
    out, err = wc.communicate()
    numAlerts = int(out)

    typeCountGrep = subprocess.Popen(['sudo', 'grep', '-i', 'scan', '/var/log/suricata/fast.log'], stdout=subprocess.PIPE)
    cutRuleSig = subprocess.Popen(['cut', '-f', '4', '-d', ' '], stdin=typeCountGrep.stdout, stdout=subprocess.PIPE)
    sort = subprocess.Popen(['sort'], stdin=cutRuleSig.stdout, stdout=subprocess.PIPE)
    uniq = subprocess.Popen(['uniq'], stdin=sort.stdout, stdout=subprocess.PIPE)
    wcTypes = subprocess.Popen(['wc', '-l'], stdin=uniq.stdout, stdout=subprocess.PIPE)
    out2, err2 = wcTypes.communicate()
    numAlertTypes = int(out2)
print msg

# transfer current scan alert count and clear the log
if(msg == "TRANSFER"):
    # packets sent and received after run complete
    countRXEnd = subprocess.Popen(['cat', '/sys/class/net/eth0/statistics/rx_packets'], stdout=subprocess.PIPE, close_fds=True)
    packetRXEnd, err = countRXEnd.communicate()
    countTXEnd = subprocess.Popen(['cat', '/sys/class/net/eth0/statistics/tx_packets'], stdout=subprocess.PIPE, close_fds=True)
    packetTXEnd, err = countTXEnd.communicate()

    packetCount = int(packetRXEnd) + int(packetTXEnd) - int(packetRXStart) - int(packetTXStart)

    # send alert count
    suricata(numAlerts, numAlertTypes, totalPackets)

    print "Alerts: " + str(numAlerts) + str(numAlertTypes) + str(packetCount)

    # clear log file
    os.system("sudo rm /var/log/suricata/fast.log")
    subprocess.call(['sudo', 'touch', '/var/log/suricata/fast.log'])

elif(msg == "RESET"):
    # packets sent and received after run complete
    countRXEnd = subprocess.Popen(['cat', '/sys/class/net/eth0/statistics/rx_packets'], stdout=subprocess.PIPE, close_fds=True)
    packetRXEnd, err = countRXEnd.communicate()}
countTXEnd = subprocess.Popen(['cat', '/sys/class/net/eth0/statistics/tx_packets'], stdout=subprocess.PIPE, close_fds=True)
packetTXEnd, err = countTXEnd.communicate()
packetCount = int(packetRXEnd) + int(packetTXEnd) - int(packetRXStart) - int(packetTXStart)

# send alert count
suricata(numAlerts, numAlertTypes, totalPackets)

print "Alerts: " + str(numAlerts) + str(numAlertTypes) + str(packetCount)

# finds and kills suricata based on PID
if(suricataRunning):
    subprocess.Popen(['sudo ./killSuricata.sh'], stdout=subprocess.PIPE, shell=True)
    suricataRunning = False

# clear log file
os.system("sudo rm /var/log/suricata/fast.log")
subprocess.call(['sudo', 'touch', '/var/log/suricata/fast.log'])

# stops program completely
elif(msg == "KILL"):
    if(suricataRunning):
        subprocess.Popen(['sudo ./killSuricata.sh'], stdout=subprocess.PIPE, shell=True)

    # kill this script
    subprocess.Popen(['"sudo killall python"'], stdin=None, stdout=devnull, stderr=devnull, shell=True)
# starts just suricata

```python
eif(msg == "START"):
    countRXStart = subprocess.Popen(['cat', '/sys/class/net/eth0/statistics/rx_packets'], stdout=subprocess.PIPE, close_fds=True)
    packetRXStart, err = countRXStart.communicate()
    countTXStart = subprocess.Popen(['cat', '/sys/class/net/eth0/statistics/tx_packets'], stdout=subprocess.PIPE, close_fds=True)
    packetTXStart, err = countTXStart.communicate()

    # launch suricata
    # sudo suricata -c /etc/suricata/suricata.yaml -i eth0
    if(not suricataRunning):
        #subprocess.Popen(["sudo "], stdin=None, stdout=devnull, stderr=devnull, shell=True)
        subprocess.Popen(["sudo suricata -c /etc/suricata/suricata.yaml -i eth0"], stdin=None, stdout=devnull, stderr=devnull, shell=True)
    suricataRunning = True
```

# stops just suricata

```python
eif(msg == "STOP"):
    # kill suricata
    # sudo kill $(ps aux | grep '[s]udo suricata -c /etc/suricata/suricata.yaml -i eth0' | awk '{print $2}')
    if(suricataRunning):
        subprocess.Popen(["sudo ./killSuricata.sh"], stdout=subprocess.PIPE, shell=True)
    suricataRunning = False
```

# stops just suricata

```python
eif(msg == "REBOOT"):
    subprocess.call(["sudo", "reboot"])
```
def suricata(numAlerts, numTypes, numPackets):
    # header information to send server
    # python dictionary
    header = {'NumAlerts': numAlerts,
              'NumTypes': numTypes,
              'NumPackets': numPackets,
              'MSG': 'SURICATA'}

    # converts dictionary to json format
    jsonHeader = json.dumps(header)

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        # creates a socket for the connection to the server
        server_address = (honeyHiveIP, port)

        # attempts to connect to the server
        sock.connect(server_address)
        sock.sendall(jsonHeader)

    except socket.error, e:
        print "Error creating Suricata socket: %s" %e

    finally:
        # closes connection
        sock.close()
Appendix D. runExperiment.py

```python
import subprocess
import sys
import socket
import json
import time as t
import random
import os
from datetime import datetime, date, time, timedelta

devnull = open(os.devnull, 'wb')

rasPi = '192.168.1.231'

honeyd1 = '192.168.1.154'
honeyd2 = '192.168.1.164'
honeyd3 = '192.168.1.174'
honeyd4 = '192.168.1.184'

port = 9830
honeyHiveIP = '192.168.1.233'
network = '192.168.1.150-192'

```
order = []

experimentFile = 'experiment.txt'

expRunNum = 0
numRuns = 30

def main():
    global order

    # checks to see if a file with the run order exists and makes sure it’s not empty
    if os.path.exists(experimentFile) and os.path.getsize(experimentFile) > 0:
        f = open(experimentFile, 'r')
        for line in f:
            value = line.split()
            lambdaFunc = None

            print value

            # address of the functions can change between runs
            # better to just check the name than store the address

            value

            if value[0] == 'nmapScan':
                lambdaFunc = nmapScan
            elif value[0] == 'baseline':
                lambdaFunc = baseline
            elif value[0] == 'wget':
                lambdaFunc = wget

            # appends all experiments in the file to be run
            order.append({"Lambda": lambdaFunc, "Name": value[0], "HoneyPots": int(value[1]), "Level": int(value[2])})

        f.close()
for i in range(numRuns):
    # runExperiment(i+1)
    # order = []
    randomizeOrder()
    random.shuffle(order)
    f = open(experimentFile, 'a')
    for run in order:
        f.write(run['Name'] + ' ' + str(run['HoneyPots']) + ' ' + str(run['Level']) + ' ' + '\n')
    f.close()

    # removes very last newline '\n' so that there isn't a blank line at the end of the file
    fT = open(experimentFile, 'rb+')
    fT.seek(-1, os.SEEK_END)
    fT.truncate()
    fT.close()

    runExperiment(1)

def runExperiment(runNumber):
    global expRunNum

    for run in order:
        # starts the specified number of honeypots, and suricata
        print "Sending Honeypot Starts"
        sendCmd(run["HoneyPots"], "START\n")
        sendCmd(run["HoneyPots"], "START\n")
        sendCmd(run["HoneyPots"], "START\n")

        # allows enough time for everything to reach a stable state
        # takes about 30 seconds for suricata to startup on the raspberry pi
t.sleep(40.0)

# packets sent and received before run
# cat /sys/class/net/enp2s0/statistics/rx_packets
# RX number of packets received
# TX number of packets transmitted
# need to cat both for full picture
countRXStart = subprocess.Popen(['cat', '/sys/class/net/enp2s0/statistics/rx_packets'], stdout=subprocess.PIPE, close_fds=True)
packetRXStart, err = countRXStart.communicate()

countTXStart = subprocess.Popen(['cat', '/sys/class/net/enp2s0/statistics/tx_packets'], stdout=subprocess.PIPE, close_fds=True)
packetTXStart, err = countTXStart.communicate()

start = datetime.now()
print "Start Time:" + start.strftime("%Y-%m-%d_%H%M")
print "nmapScan" + " Level:" + str(run["Level"]) + ' Honeypots: ' + str(run["HoneyPots"])

# runs the specified test with corresponding level
run["Lambda"](run["Level"], run["HoneyPots"])

# packets sent and received after run complete
countRXEnd = subprocess.Popen(['cat', '/sys/class/net/enp2s0/statistics/rx_packets'], stdout=subprocess.PIPE, close_fds=True)
packetRXEnd, err = countRXEnd.communicate()
countTXEnd = subprocess.Popen(['cat', '/sys/class/net/enp2s0/statistics/tx_packets'], stdout=subprocess.PIPE, close_fds=True)
packetTXEnd, err = countTXEnd.communicate()

# print packetCountEnd
packetCount = int(packetRXEnd) + int(packetTXEnd) - int(packetRXStart) - int(packetTXStart)

print 'Packets Sent and Received: ' + str(packetCount)

end = datetime.now()
print "End Time:" + end.strftime("%Y-%m-%d_%H%M")
elapsedTime = (end - start).total_seconds()
print "Elapsed Time:" + str(elapsedTime)

print "Sending Honeypot Resets"

# stops all honeypots and forces transfer of pcap, stops suricata
sendCmd(run["HoneyPots"], "RESET\n")
sendCmd(run["HoneyPots"], "RESET\n")
sendCmd(run["HoneyPots"], "RESET\n")

# time required for honeypots and suricata to send all data to c2 server
t.sleep(45.0)

sendSnort()

# allows time for Snort to parse PCAP files
t.sleep(90.0)

# tells the C2 server to reset and send gathered stats to this script
res = sendReset()
# write gathered results to corresponding csv file
writeResults(run, res, packetCount, elapsedTime)

# removes run from experimentFile since it successfully completed
lines = open(experimentFile).readlines()
with open(experimentFile, 'w') as f:
    f.writelines(lines[1:])
f.close()

# Reboots all machines for a clean stable start state
print "Sending Honeypot Reboots"
sendCmd(run["HoneyPots"], "REBOOT\n")
sendCmd(run["HoneyPots"], "REBOOT\n")
sendCmd(run["HoneyPots"], "REBOOT\n")
sendReboot()

# allows plenty of time for all devices to reboot
# and startup scripts before proceeding to next iteration
# win 10 VM takes longest (30 seconds for reboot, another 15
# for c2 server startup - 45s total)
# sometimes can take longer and crash script....
t.sleep(45.0)

# makes sure c2 server is up and running berfore starting
next iteration
sendStart()
expRunNum += 1

# randomizes the order in which all 36 tests are run
def randomizeOrder():
    # initial testing
    #test = ["Lambda": wget, "Name": 'wget']

test = ["Lambda": nmapScan, "Name": 'nmapScan'] #, {"Lambda": baseline, "Name": 'baseline'}] #, {"Lambda": wget, "Name": 'wget '}

for i in range(4):
    numHps = i;
    for testType in test:
        for level in range(4):
            order.append("Lambda": testType["Lambda"], "Name": testType['Name'], "HoneyPots": numHps, "Level": level})
        #random.shuffle(order)

# tests to see if alerts are generated for different kinds of scans
# Levels are the different scan types:
    # nmap 192.168.1.0/24 -sT
    # nmap 192.168.1.0/24 -A
    # sudo nmap 192.168.1.0/24 --max-hostgroup 1 --randomize-hosts -f 8

def nmapScan(scanType, numHps):
    global expRunNum
    if (scanType == 0):
        print "Control Group: sleep(1341)"
        t.sleep(1341)
    elif (scanType == 1):
        print "nmap ipLst.txt -sT -Pn"
        subprocess.call(['nmap', '-iL', 'ipLst.txt', '-sT', '-Pn', '-oX', 'run'+str(expRunNum)+'.xml'], stdin=None, stdout=None, stderr=None, shell=False)
elif (scanType == 2):
    print "nmap ipLst.txt -A -Pn"
    subprocess.call(['nmap', '-iL', 'ipLst.txt', '-A', '-Pn', '-oX', 'run'+str(expRunNum)+'.xml'], stdin=None, stdout=None, stderr=None, shell=False)
elif (scanType == 3):
    print "sudo nmap ipLst.txt --scan-delay 1075ms --randomize-hosts -f 8 -Pn"
    subprocess.call(['sudo', 'nmap', '-iL', 'ipLst.txt', '--scan-delay', '1075ms', '--randomize-hosts', '-f', '8', '-Pn', '-oX', 'run'+str(expRunNum)+'.xml'], stdin=None, stdout=None, stderr=None, shell=False)

# tests to see if an alert is generated for malicious wgets that modify IoT device
# Levels are the different types of devices:
# TITAThink Camera 192.168.1.1[5-8]0
# Prolophix Thermostat 192.168.1.1[5-8]1
# ez-Outlet 2 Power Outlet 192.168.1.1[5-8]2

def wget(iotDevice, numHps):
    if (iotDevice == 0):
        print "Control Group: sleep(30)"
        t.sleep(30)
    elif (iotDevice == 1):
        for ip in range (numHps + 1):
            subprocess.call(['wget', '--user', 'admin', '--password', 'admin', '--tries', '1', '--timeout', '1', 'http://'+camera[ip]+'/form/deleteStorageAllApply?lang=en'],
                            stdin=None, stdout=None, stderr=None, shell=False)
        print "wget TITAThink Cameras"
elif (iotDevice == 2):
    print "wget Prolophix Thermostats"
    for ip in range (numHps + 1):
        subprocess.call(["wget", '--user', 'admin', '--password', 'admin', '--tries', '1', '--timeout', '1', 'http://'+thermostat[ip]+'/index.shtml'],
                        stdin=None, stdout=None, stderr=None, shell=False)

elif (iotDevice == 3):
    print "wget ez-Outlets"
    for ip in range (numHps + 1):
        subprocess.call(["wget", '--tries', '1', '--timeout', '1', 'http://'+outlet[ip]+'/invert.cgi'],
                        stdin=None, stdout=None, stderr=None, shell=False)

# tests to see if alerts are generated with no alert traffic
# levels: 20s, 660s, 6000s - to match length of each scan (from pilot studies)

def baseline(runTime, numHps):
    if (runTime == 1):
        print "Baseline 30s"
        t.sleep(30.0)
    elif (runTime == 2):
        print "Baseline 660s"
        t.sleep(660.0)
    else:
        print "Baseline 1500s"
        t.sleep(1500.0)

def sendCmd(numHps, cmd):
    # 1 non-mirrored
if (numHPs == 1):
    hp1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    hp1.sendto(cmd, (honeyd1, port))
    hp1.close()

# 1 non-mirrored, 1 mirrored
elif (numHPs == 2):
    hp1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    hp1.sendto(cmd, (honeyd1, port))
    hp1.close()

    hp3 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    hp3.sendto(cmd, (honeyd3, port))
    hp3.close()

# 2 non-mirrored, 1 mirrored
elif (numHPs == 3):
    hp1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    hp1.sendto(cmd, (honeyd1, port))
    hp1.close()

    hp2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    hp2.sendto(cmd, (honeyd2, port))
    hp2.close()

    hp3 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    hp3.sendto(cmd, (honeyd3, port))
    hp3.close()

# 2 non-mirrored, 2 mirrored
elif (numHPs == 4):
    hp1 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
hp1.sendto(cmd, (honeyd1, port))
hp1.close()

hp2 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
hp2.sendto(cmd, (honeyd2, port))
hp2.close()

hp3 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
hp3.sendto(cmd, (honeyd3, port))
hp3.close()

hp4 = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
hp4.sendto(cmd, (honeyd4, port))
hp4.close()

# starts suricata on raspberry pi
ras = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
ras.sendto(cmd, (rasPi, port))
ras.close()

def sendStart():
    print "START sent to C2 Server"
    while True:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            # creates a socket for the connection to the server
            server_address = (honeyHiveIP, port)
            # attempts to connect to the server
            sock.connect(server_address)
```python
# header information to send server
# python dictionary
header = {"MSG": 'START'}

# converts dictionary to json format
jsonHeader = json.dumps(header)

# sends all data
sock.sendall(jsonHeader)

except socket.error, e:
    print "Error creating Start Socket: %s" %e
    continue
    break
    sock.close()

def sendSnort():
    print "SNORT sent to C2 Server"

    while True:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

            # creates a socket for the connection to the server
            server_address = (honeyHiveIP, port)

            # attempts to connect to the server
            sock.connect(server_address)

            # header information to send server
            # python dictionary
```

header = json.dumps(header)

jsonHeader = json.dumps(header)

# sends all data
sock.sendall(jsonHeader)

except socket.error, e:
    print "Error creating Snort Socket: %s" % e
    continue
    break
sock.close()

def sendReboot():
    print "REBOOT Sent to C2 Server"
    while True:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

            # creates a socket for the connection to the server
            server_address = (honeyHiveIP, port)

            # attempts to connect to the server
            sock.connect(server_address)

            # header information to send server
            # python dictionary
            header = json.dumps(header)

            # converts dictionary to json format
            jsonHeader = json.dumps(header)

            # sends all data
            sock.sendall(jsonHeader)

except socket.error, e:
    print "Error creating Snort Socket: %s" % e
    continue
    break
sock.close()
# sends all data
sock.sendall(jsonHeader)

except socket.error, e:
    print "Error creating Reboot Socket: %s" %e
    continue
    break
sock.close()

def sendReset():
    print "RESET sent to C2 Server"
    while True:
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

            # creates a socket for the connection to the server
            server_address = (honeyHiveIP, port)

            # attempts to connect to the server
            sock.connect(server_address)

            # header information to send server
            # python dictionary
            header = {"MSG": 'RESET'}

            # converts dictionary to json format
            jsonHeader = json.dumps(header)

            # sends all data
            sock.sendall(jsonHeader)
#(Interactions: numInteractions, Snort: numSnortAlerts,

Packets: numPackets)

res = sock.recv(4096)

e

except socket.error, e:
    print "Error creating Reset Socket: %s" %e
    continue
    break
sock.close()

return json.loads(res)

def writeResults(runInfo, results, packetCount, elapsedTime):

    print "Writing Results"
    # add suricata packet count and suricata num types of alerts
count

    output = str(runInfo['Level']) + ',' + str(runInfo['HoneyPots'])
     + ',' + str(results['numHoneypots']) + ',' + str(results['
Interactions']) + ',' + str(results['numPCAPs']) + ',' + str( results['SnortICount']) + ',' + str(results['SnortMCount']) + ',
     ' + str(results['Snort']) + ',' + str(results['SnortTypes']) + '
     ,'+ str(results['SnortMerged']) + ',' + str(results['
SnortTypesMerged']) + ',' + str(results['Suricata']) + ',' + str,
     (results['SuricataTypes']) + ',', + str(packetCount) + ',', + str( results['Packets']) + ',', + str(results['suricataPackets']) + ',',
     + str(elapsedTime) + '
    print "Output: " + output

    #nmap

    if (runInfo['Name'] == 'nmapScan'):
        nmap_file = open('nmap.csv', mode='a')
nmap_file.write(output)
nmap_file.close()

# baseline
eelif (runInfo["Name"] == 'baseline'):
    baseline_file = open('baseline.csv', mode='a')
    baseline_file.write(output)
    baseline_file.close()

# wget
else:
    wget_file = open('wget.csv', mode='a')
    wget_file.write(output)
    wget_file.close()

main()
# Appendix E. Experiment Results

## Table 13. Experiment Results

<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>8</td>
<td>0</td>
<td>18104</td>
<td>0%</td>
<td>32%</td>
<td>56978</td>
<td>2365</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96</td>
<td>0%</td>
<td>7%</td>
<td>1436</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>0</td>
<td>15447</td>
<td>0%</td>
<td>27%</td>
<td>57756</td>
<td>2365</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>8</td>
<td>0</td>
<td>3892</td>
<td>0%</td>
<td>12%</td>
<td>31239</td>
<td>312</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>TCP Connect</td>
<td>6</td>
<td>35</td>
<td>5</td>
<td>33</td>
<td>7</td>
<td>40</td>
<td>8</td>
<td>10706</td>
<td>9110</td>
<td>33%</td>
<td>28%</td>
<td>32030</td>
<td>129</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Aggressive</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
<td>12</td>
<td>13508</td>
<td>31709</td>
<td>2%</td>
<td>4%</td>
<td>893271</td>
<td>1093</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>125</td>
<td>12</td>
<td>0</td>
<td>32153</td>
<td>0%</td>
<td>4%</td>
<td>842658</td>
<td>679</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89</td>
<td>0%</td>
<td>7%</td>
<td>1218</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>115</td>
<td>14</td>
<td>0</td>
<td>31803</td>
<td>0%</td>
<td>4%</td>
<td>857500</td>
<td>1086</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>124</td>
<td>12</td>
<td>13456</td>
<td>19050</td>
<td>7%</td>
<td>11%</td>
<td>181337</td>
<td>1829</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>5</td>
<td>7942</td>
<td>13030</td>
<td>10%</td>
<td>16%</td>
<td>79264</td>
<td>6459</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>TCP Connect</td>
<td>6</td>
<td>27</td>
<td>5</td>
<td>25</td>
<td>7</td>
<td>30</td>
<td>6</td>
<td>8490</td>
<td>5569</td>
<td>28%</td>
<td>18%</td>
<td>30590</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Aggressive</td>
<td>3</td>
<td>61</td>
<td>7</td>
<td>69</td>
<td>9</td>
<td>107</td>
<td>11</td>
<td>11647</td>
<td>31299</td>
<td>2%</td>
<td>5%</td>
<td>673715</td>
<td>1163</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>112</td>
<td>12</td>
<td>0</td>
<td>32658</td>
<td>0%</td>
<td>4%</td>
<td>727012</td>
<td>1593</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>TCP Connect</td>
<td>6</td>
<td>32</td>
<td>5</td>
<td>33</td>
<td>8</td>
<td>44</td>
<td>8</td>
<td>10434</td>
<td>9281</td>
<td>33%</td>
<td>29%</td>
<td>31725</td>
<td>139</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TCP Connect</td>
<td>6</td>
<td>33</td>
<td>5</td>
<td>28</td>
<td>7</td>
<td>38</td>
<td>8</td>
<td>10209</td>
<td>7751</td>
<td>33%</td>
<td>25%</td>
<td>30671</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>33</td>
<td>7</td>
<td>63</td>
<td>8</td>
<td>32051</td>
<td>36139</td>
<td>32%</td>
<td>36%</td>
<td>100940</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0%</td>
<td>7%</td>
<td>1437</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>19</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>93</td>
<td>8</td>
<td>87</td>
<td>10</td>
<td>54</td>
<td>9</td>
<td>18912</td>
<td>12077</td>
<td>7%</td>
<td>5%</td>
<td>267195</td>
<td>1089</td>
</tr>
<tr>
<td>20</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>62</td>
<td>8</td>
<td>19988</td>
<td>36159</td>
<td>23%</td>
<td>41%</td>
<td>87887</td>
<td>2365</td>
</tr>
<tr>
<td>21</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>67</td>
<td>6</td>
<td>62</td>
<td>8</td>
<td>53</td>
<td>7</td>
<td>16992</td>
<td>12077</td>
<td>52%</td>
<td>37%</td>
<td>32767</td>
<td>1096</td>
</tr>
<tr>
<td>22</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>112</td>
<td>0</td>
<td>31907</td>
<td>0%</td>
<td>5%</td>
<td>702602</td>
<td>698</td>
</tr>
<tr>
<td>23</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>55</td>
<td>7</td>
<td>55</td>
<td>9</td>
<td>111</td>
<td>12</td>
<td>14269</td>
<td>31707</td>
<td>2%</td>
<td>4%</td>
<td>712468</td>
<td>838</td>
</tr>
<tr>
<td>24</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>117</td>
<td>12</td>
<td>32710</td>
<td>0%</td>
<td>5%</td>
<td>699017</td>
<td>622</td>
</tr>
<tr>
<td>25</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>6</td>
<td>13444</td>
<td>18133</td>
<td>19%</td>
<td>26%</td>
<td>69043</td>
<td>2366</td>
</tr>
<tr>
<td>26</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>21</td>
<td>5</td>
<td>21</td>
<td>7</td>
<td>31</td>
<td>8</td>
<td>2413</td>
<td>2120</td>
<td>6%</td>
<td>6%</td>
<td>37184</td>
<td>404</td>
</tr>
<tr>
<td>27</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>65</td>
<td>8</td>
<td>19797</td>
<td>36120</td>
<td>23%</td>
<td>41%</td>
<td>87561</td>
<td>2367</td>
</tr>
<tr>
<td>28</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>173</td>
<td>0%</td>
<td>17%</td>
<td>1045</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>109</td>
<td>12</td>
<td>13053</td>
<td>31143</td>
<td>2%</td>
<td>4%</td>
<td>721156</td>
<td>742</td>
</tr>
<tr>
<td>30</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>13</td>
<td>5</td>
<td>16</td>
<td>5</td>
<td>45</td>
<td>10</td>
<td>5732</td>
<td>9613</td>
<td>15%</td>
<td>24%</td>
<td>39352</td>
<td>288</td>
</tr>
<tr>
<td>31</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>0%</td>
<td>8%</td>
<td>1164</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>31</td>
<td>8</td>
<td>2381</td>
<td>3588</td>
<td>9%</td>
<td>14%</td>
<td>25923</td>
<td>79</td>
</tr>
<tr>
<td>33</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>17</td>
<td>5</td>
<td>19</td>
<td>7</td>
<td>66</td>
<td>8</td>
<td>21364</td>
<td>36139</td>
<td>21%</td>
<td>35%</td>
<td>101819</td>
<td>2523</td>
</tr>
<tr>
<td>34</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>7%</td>
<td>1096</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>65</td>
<td>0%</td>
<td>6%</td>
<td>1035</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>126</td>
<td>8</td>
<td>118</td>
<td>8</td>
<td>77</td>
<td>9</td>
<td>22445</td>
<td>12080</td>
<td>60%</td>
<td>32%</td>
<td>37418</td>
<td>1108</td>
</tr>
<tr>
<td>37</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>28</td>
<td>8</td>
<td>2607</td>
<td>2516</td>
<td>7%</td>
<td>7%</td>
<td>35691</td>
<td>398</td>
</tr>
<tr>
<td>38</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>20</td>
<td>6</td>
<td>2319</td>
<td>1802</td>
<td>9%</td>
<td>7%</td>
<td>24792</td>
<td>47</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td>39</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96</td>
<td>0%</td>
<td>9%</td>
<td></td>
<td>1051</td>
<td>1341</td>
</tr>
<tr>
<td>40</td>
<td>TCP Connect</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>43</td>
<td>8</td>
<td>0</td>
<td>12008</td>
<td>0%</td>
<td>9%</td>
<td></td>
<td>4151</td>
<td>1325</td>
</tr>
<tr>
<td>41</td>
<td>Aggressive</td>
<td>9</td>
<td>220</td>
<td>204</td>
<td>9</td>
<td>67</td>
<td>7</td>
<td>30154</td>
<td>12663</td>
<td>83%</td>
<td>35%</td>
<td>35%</td>
<td></td>
<td>36358</td>
<td>1346</td>
</tr>
<tr>
<td>42</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0%</td>
<td>8%</td>
<td></td>
<td>1173</td>
<td>1341</td>
</tr>
<tr>
<td>43</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>35</td>
<td>7</td>
<td>65</td>
<td>8</td>
<td>32047</td>
<td>36187</td>
<td>32%</td>
<td>36%</td>
<td></td>
<td>100513</td>
<td>2364</td>
</tr>
<tr>
<td>44</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0%</td>
<td>8%</td>
<td></td>
<td>1473</td>
<td>1341</td>
</tr>
<tr>
<td>45</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0%</td>
<td>8%</td>
<td></td>
<td>1473</td>
<td>1341</td>
</tr>
<tr>
<td>46</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0%</td>
<td>8%</td>
<td></td>
<td>1473</td>
<td>1341</td>
</tr>
<tr>
<td>47</td>
<td>TCP Connect</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>23</td>
<td>8</td>
<td>2665</td>
<td>3932</td>
<td>10%</td>
<td>14%</td>
<td></td>
<td>27946</td>
<td>180</td>
</tr>
<tr>
<td>48</td>
<td>TCP Connect</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>8</td>
<td>17267</td>
<td>7562</td>
<td>47%</td>
<td>21%</td>
<td></td>
<td>36457</td>
<td>25</td>
</tr>
<tr>
<td>49</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>115</td>
<td>11</td>
<td>0</td>
<td>32110</td>
<td>0%</td>
<td>3%</td>
<td></td>
<td>990706</td>
<td>950</td>
</tr>
<tr>
<td>50</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>33</td>
<td>7</td>
<td>68</td>
<td>8</td>
<td>32056</td>
<td>36234</td>
<td>32%</td>
<td>36%</td>
<td></td>
<td>100504</td>
<td>2365</td>
</tr>
<tr>
<td>51</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>8</td>
<td>0</td>
<td>6685</td>
<td>0%</td>
<td>24%</td>
<td></td>
<td>28329</td>
<td>321</td>
</tr>
<tr>
<td>52</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>52</td>
<td>7</td>
<td>0</td>
<td>18122</td>
<td>0%</td>
<td>32%</td>
<td></td>
<td>57333</td>
<td>2367</td>
</tr>
<tr>
<td>53</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>52</td>
<td>0%</td>
<td>5%</td>
<td></td>
<td>1281</td>
<td>1341</td>
</tr>
<tr>
<td>54</td>
<td>Aggressive</td>
<td>9</td>
<td>174</td>
<td>7</td>
<td>155</td>
<td>9</td>
<td>54</td>
<td>7</td>
<td>37572</td>
<td>12096</td>
<td>97%</td>
<td>31%</td>
<td></td>
<td>38909</td>
<td>1111</td>
</tr>
<tr>
<td>55</td>
<td>TCP Connect</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>26</td>
<td>6</td>
<td>2457</td>
<td>4213</td>
<td>8%</td>
<td>14%</td>
<td></td>
<td>30038</td>
<td>252</td>
</tr>
<tr>
<td>56</td>
<td>TCP Connect</td>
<td>9</td>
<td>54</td>
<td>5</td>
<td>42</td>
<td>7</td>
<td>35</td>
<td>8</td>
<td>16239</td>
<td>6219</td>
<td>45%</td>
<td>17%</td>
<td></td>
<td>36342</td>
<td>23</td>
</tr>
<tr>
<td>57</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0%</td>
<td>6%</td>
<td></td>
<td>1591</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td>6</td>
<td>0</td>
<td>18200</td>
<td>0%</td>
<td>31%</td>
<td></td>
<td>58469</td>
<td>2367</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33</td>
<td>8</td>
<td>0</td>
<td>4873</td>
<td>0%</td>
<td>16%</td>
<td>30904</td>
<td>366</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>69</td>
<td>7</td>
<td>66</td>
<td>9</td>
<td>36</td>
<td>9</td>
<td>11084</td>
<td>3%</td>
<td>6%</td>
<td>385004</td>
<td>1828</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>TCP Connect</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>7</td>
<td>10031</td>
<td>7352</td>
<td>19%</td>
<td>14%</td>
<td>51586</td>
<td>2677</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0%</td>
<td>9%</td>
<td>1117</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>8</td>
<td>0</td>
<td>6123</td>
<td>0%</td>
<td>26%</td>
<td>23572</td>
<td>247</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>78</td>
<td>7</td>
<td>74</td>
<td>9</td>
<td>132</td>
<td>10</td>
<td>9304</td>
<td>18645</td>
<td>3%</td>
<td>6%</td>
<td>44115</td>
<td>1817</td>
</tr>
<tr>
<td>65</td>
<td>TCP Connect</td>
<td>6</td>
<td>27</td>
<td>5</td>
<td>24</td>
<td>7</td>
<td>34</td>
<td>8</td>
<td>8321</td>
<td>6698</td>
<td>27%</td>
<td>21%</td>
<td>31299</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>6%</td>
<td>1261</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>TCP Connect</td>
<td>9</td>
<td>55</td>
<td>5</td>
<td>42</td>
<td>7</td>
<td>37</td>
<td>8</td>
<td>16691</td>
<td>8561</td>
<td>46%</td>
<td>24%</td>
<td>36386</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>5</td>
<td>0</td>
<td>4564</td>
<td>0%</td>
<td>15%</td>
<td>31116</td>
<td>409</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>TCP Connect</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>16</td>
<td>6</td>
<td>2369</td>
<td>2076</td>
<td>7%</td>
<td>6%</td>
<td>34185</td>
<td>330</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0%</td>
<td>8%</td>
<td>1188</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0%</td>
<td>8%</td>
<td>1211</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>6</td>
<td>16174</td>
<td>6893</td>
<td>44%</td>
<td>10%</td>
<td>36607</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>109</td>
<td>0%</td>
<td>10%</td>
<td>1123</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>103</td>
<td>7</td>
<td>99</td>
<td>9</td>
<td>119</td>
<td>10</td>
<td>24344</td>
<td>17919</td>
<td>6%</td>
<td>5%</td>
<td>382701</td>
<td>1826</td>
</tr>
<tr>
<td>75</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>84</td>
<td>0%</td>
<td>7%</td>
<td>1216</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>125</td>
<td>11</td>
<td>14966</td>
<td>18280</td>
<td>36%</td>
<td>44%</td>
<td>41339</td>
<td>1806</td>
</tr>
<tr>
<td>77</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>110</td>
<td>0%</td>
<td>8%</td>
<td>1460</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>5</td>
<td>62</td>
<td>8</td>
<td>19992</td>
<td>36290</td>
<td>23%</td>
<td>41%</td>
<td>88185</td>
<td>2366</td>
<td></td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>79</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>8</td>
<td>0</td>
<td>5310</td>
<td>18%</td>
<td>0%</td>
<td>18%</td>
<td>29927</td>
<td>321</td>
</tr>
<tr>
<td>80</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>129</td>
<td>9</td>
<td>123</td>
<td>9</td>
<td>26</td>
<td>9</td>
<td>18030</td>
<td>3%</td>
<td>18%</td>
<td>4%</td>
<td>514449</td>
<td>1089</td>
</tr>
<tr>
<td>81</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>42</td>
<td>8</td>
<td>577</td>
<td>10136</td>
<td>32%</td>
<td>2%</td>
<td>3%</td>
<td>32128</td>
<td>154</td>
</tr>
<tr>
<td>82</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>108</td>
<td>6</td>
<td>112</td>
<td>7</td>
<td>67</td>
<td>8</td>
<td>19822</td>
<td>41%</td>
<td>23%</td>
<td>17%</td>
<td>87978</td>
<td>2365</td>
</tr>
<tr>
<td>83</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>9</td>
<td>29</td>
<td>8</td>
<td>2634</td>
<td>2217</td>
<td>6%</td>
<td>7%</td>
<td>6%</td>
<td>36804</td>
<td>471</td>
</tr>
<tr>
<td>84</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>8%</td>
<td>0%</td>
<td>8%</td>
<td>1114</td>
<td>1341</td>
</tr>
<tr>
<td>85</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>79</td>
<td>7%</td>
<td>7%</td>
<td>7%</td>
<td>1057</td>
<td>1341</td>
</tr>
<tr>
<td>86</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>36</td>
<td>5</td>
<td>35</td>
<td>5</td>
<td>26</td>
<td>8</td>
<td>2523</td>
<td>6%</td>
<td>7%</td>
<td>6%</td>
<td>35480</td>
<td>382</td>
</tr>
<tr>
<td>87</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>137</td>
<td>8</td>
<td>122</td>
<td>10</td>
<td>124</td>
<td>12</td>
<td>37932</td>
<td>45%</td>
<td>96%</td>
<td>45%</td>
<td>39660</td>
<td>1802</td>
</tr>
<tr>
<td>88</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>156</td>
<td>9</td>
<td>137</td>
<td>10</td>
<td>120</td>
<td>10</td>
<td>29815</td>
<td>43%</td>
<td>70%</td>
<td>3%</td>
<td>43211</td>
<td>1806</td>
</tr>
<tr>
<td>89</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>74</td>
<td>7%</td>
<td>0%</td>
<td>7%</td>
<td>1026</td>
<td>1341</td>
</tr>
<tr>
<td>90</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>196</td>
<td>20%</td>
<td>0%</td>
<td>20%</td>
<td>1001</td>
<td>1341</td>
</tr>
<tr>
<td>91</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>61</td>
<td>5</td>
<td>39</td>
<td>7</td>
<td>35</td>
<td>8</td>
<td>15533</td>
<td>19%</td>
<td>43%</td>
<td>19%</td>
<td>36470</td>
<td>20</td>
</tr>
<tr>
<td>92</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>18</td>
<td>5</td>
<td>63</td>
<td>8</td>
<td>19817</td>
<td>41%</td>
<td>23%</td>
<td>41%</td>
<td>87659</td>
<td>2367</td>
</tr>
<tr>
<td>93</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>104</td>
<td>7</td>
<td>104</td>
<td>9</td>
<td>118</td>
<td>10</td>
<td>23544</td>
<td>4%</td>
<td>5%</td>
<td>4%</td>
<td>507274</td>
<td>1829</td>
</tr>
<tr>
<td>94</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>31</td>
<td>8</td>
<td>2608</td>
<td>14%</td>
<td>9%</td>
<td>14%</td>
<td>30338</td>
<td>295</td>
</tr>
<tr>
<td>95</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>33</td>
<td>7</td>
<td>59</td>
<td>8</td>
<td>32071</td>
<td>36%</td>
<td>32%</td>
<td>36%</td>
<td>100046</td>
<td>2365</td>
</tr>
<tr>
<td>96</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>43</td>
<td>7</td>
<td>46</td>
<td>9</td>
<td>107</td>
<td>12</td>
<td>12392</td>
<td>3%</td>
<td>1%</td>
<td>3%</td>
<td>1120935</td>
<td>932</td>
</tr>
<tr>
<td>97</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>58</td>
<td>7</td>
<td>59</td>
<td>9</td>
<td>92</td>
<td>8</td>
<td>13785</td>
<td>41%</td>
<td>41%</td>
<td>147%</td>
<td>33270</td>
<td>408</td>
</tr>
<tr>
<td>98</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>47</td>
<td>7</td>
<td>48</td>
<td>9</td>
<td>103</td>
<td>12</td>
<td>11984</td>
<td>3%</td>
<td>1%</td>
<td>3%</td>
<td>1121648</td>
<td>953</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>99</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>30</td>
<td>6</td>
<td>13457</td>
<td>18121</td>
<td>19%</td>
<td>26%</td>
<td>69068</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>TCP Connect 3</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>11</td>
<td>7</td>
<td>33</td>
<td>8</td>
<td>2642</td>
<td>2661</td>
<td>8%</td>
<td>8%</td>
<td>35200</td>
<td>375</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>Aggressive 9</td>
<td>8</td>
<td>206</td>
<td>10</td>
<td>178</td>
<td>11</td>
<td>123</td>
<td>10</td>
<td>35124</td>
<td>16630</td>
<td>108%</td>
<td>51%</td>
<td>32566</td>
<td>1870</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>69</td>
<td>8</td>
<td>19816</td>
<td>36170</td>
<td>22%</td>
<td>41%</td>
<td>88593</td>
<td>2363</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>TCP Connect 6</td>
<td>6</td>
<td>26</td>
<td>5</td>
<td>25</td>
<td>7</td>
<td>41</td>
<td>8</td>
<td>8405</td>
<td>8325</td>
<td>26%</td>
<td>26%</td>
<td>32046</td>
<td>179</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>31</td>
<td>8</td>
<td>11941</td>
<td>17841</td>
<td>17%</td>
<td>25%</td>
<td>70457</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>Control Group 6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>0%</td>
<td>6%</td>
<td>1551</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>Control Group 6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>0%</td>
<td>5%</td>
<td>1688</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>23</td>
<td>8</td>
<td>19817</td>
<td>36292</td>
<td>22%</td>
<td>41%</td>
<td>88230</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>TCP Connect 6</td>
<td>6</td>
<td>23</td>
<td>5</td>
<td>20</td>
<td>7</td>
<td>39</td>
<td>8</td>
<td>6603</td>
<td>7154</td>
<td>20%</td>
<td>22%</td>
<td>32983</td>
<td>146</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>121</td>
<td>4</td>
<td>124</td>
<td>5</td>
<td>64</td>
<td>8</td>
<td>19821</td>
<td>36117</td>
<td>23%</td>
<td>41%</td>
<td>87948</td>
<td>2366</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>5</td>
<td>32</td>
<td>8</td>
<td>13458</td>
<td>18123</td>
<td>19%</td>
<td>26%</td>
<td>69950</td>
<td>2366</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>TCP Connect 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>0%</td>
<td>5%</td>
<td>1688</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>Control Group 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>85</td>
<td>0%</td>
<td>6%</td>
<td>1384</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>Control Group 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>174</td>
<td>0%</td>
<td>10%</td>
<td>1710</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>TCP Connect 6</td>
<td>6</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>32</td>
<td>6</td>
<td>2387</td>
<td>7857</td>
<td>8%</td>
<td>25%</td>
<td>31182</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>32</td>
<td>7</td>
<td>63</td>
<td>8</td>
<td>32064</td>
<td>36118</td>
<td>32%</td>
<td>36%</td>
<td>100385</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>Control Group 9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>76</td>
<td>0%</td>
<td>7%</td>
<td>1071</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>Control Group 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>0%</td>
<td>6%</td>
<td>1466</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>TCP Connect 9</td>
<td>9</td>
<td>13</td>
<td>5</td>
<td>16</td>
<td>7</td>
<td>48</td>
<td>10</td>
<td>5785</td>
<td>7542</td>
<td>15%</td>
<td>19%</td>
<td>39194</td>
<td>280</td>
<td></td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>119</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>108</td>
<td>7</td>
<td>108</td>
<td>9</td>
<td>234</td>
<td>10</td>
<td>22964</td>
<td>15869</td>
<td>7%</td>
<td>5%</td>
<td>324561</td>
<td>555</td>
</tr>
<tr>
<td>120</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>67</td>
<td>0%</td>
<td>6%</td>
<td>1210</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>68</td>
<td>8</td>
<td>19998</td>
<td>36136</td>
<td>23%</td>
<td>41%</td>
<td>87976</td>
<td>2366</td>
</tr>
<tr>
<td>122</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>24</td>
<td>5</td>
<td>22</td>
<td>7</td>
<td>45</td>
<td>8</td>
<td>6286</td>
<td>9568</td>
<td>19%</td>
<td>29%</td>
<td>33472</td>
<td>203</td>
</tr>
<tr>
<td>123</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>50</td>
<td>7</td>
<td>47</td>
<td>9</td>
<td>269</td>
<td>12</td>
<td>8637</td>
<td>35068</td>
<td>2%</td>
<td>8%</td>
<td>413867</td>
<td>439</td>
</tr>
<tr>
<td>124</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>34</td>
<td>7</td>
<td>61</td>
<td>8</td>
<td>32091</td>
<td>36125</td>
<td>32%</td>
<td>36%</td>
<td>100790</td>
<td>2367</td>
</tr>
<tr>
<td>125</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>116</td>
<td>12</td>
<td>0</td>
<td>31918</td>
<td>0%</td>
<td>3%</td>
<td>1095314</td>
<td>718</td>
</tr>
<tr>
<td>126</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>76</td>
<td>0%</td>
<td>7%</td>
<td>1107</td>
<td>1341</td>
</tr>
<tr>
<td>127</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>8</td>
<td>6440</td>
<td>0%</td>
<td>25%</td>
<td>25899</td>
<td>252</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>92</td>
<td>0%</td>
<td>7%</td>
<td>1360</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96</td>
<td>0%</td>
<td>8%</td>
<td>1200</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>130</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>118</td>
<td>12</td>
<td>0</td>
<td>32203</td>
<td>0%</td>
<td>3%</td>
<td>1000779</td>
<td>721</td>
</tr>
<tr>
<td>131</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>94</td>
<td>0%</td>
<td>8%</td>
<td>1185</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>133</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>134</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>135</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>136</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>103</td>
<td>11</td>
<td>0</td>
<td>30912</td>
<td>0%</td>
<td>5%</td>
<td>626567</td>
<td>1019</td>
</tr>
<tr>
<td>137</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
<td>31%</td>
<td>58084</td>
<td>2369</td>
<td></td>
</tr>
<tr>
<td>138</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>121</td>
<td>12</td>
<td>0</td>
<td>31342</td>
<td>0%</td>
<td>5%</td>
<td>617717</td>
<td>604</td>
</tr>
</tbody>
</table>

Table 13 continued from previous page
<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>139</td>
<td>Control Group 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96</td>
<td>0%</td>
<td>4%</td>
<td>2441</td>
<td>1341</td>
</tr>
<tr>
<td>140</td>
<td>TCP Connect 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>8</td>
<td>4563</td>
<td>0%</td>
<td>19%</td>
<td>23664</td>
<td>221</td>
</tr>
<tr>
<td>141</td>
<td>TCP Connect 6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>8</td>
<td>8454</td>
<td>27%</td>
<td>18%</td>
<td>30797</td>
<td>51</td>
</tr>
<tr>
<td>142</td>
<td>Control Group 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0%</td>
<td>6%</td>
<td>1681</td>
<td>1341</td>
</tr>
<tr>
<td>143</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>6</td>
<td>18153</td>
<td>0%</td>
<td>31%</td>
<td>57711</td>
<td>2366</td>
</tr>
<tr>
<td>144</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>33</td>
<td>6</td>
<td>12073</td>
<td>17%</td>
<td>26%</td>
<td>70010</td>
<td>2369</td>
</tr>
<tr>
<td>145</td>
<td>Control Group 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89</td>
<td>0%</td>
<td>5%</td>
<td>1641</td>
<td>1341</td>
</tr>
<tr>
<td>146</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>18218</td>
<td>0%</td>
<td>31%</td>
<td>57985</td>
<td>2366</td>
</tr>
<tr>
<td>147</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>32</td>
<td>5</td>
<td>36</td>
<td>5</td>
<td>66</td>
<td>19891</td>
<td>22%</td>
<td>41%</td>
<td>89170</td>
<td>2367</td>
</tr>
<tr>
<td>148</td>
<td>TCP Connect 9</td>
<td>9</td>
<td>17</td>
<td>4</td>
<td>14</td>
<td>6</td>
<td>34</td>
<td>5507</td>
<td>15%</td>
<td>17%</td>
<td>36474</td>
<td>21</td>
</tr>
<tr>
<td>149</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>6</td>
<td>18118</td>
<td>0%</td>
<td>31%</td>
<td>57733</td>
<td>2365</td>
</tr>
<tr>
<td>150</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>64</td>
<td>19808</td>
<td>22%</td>
<td>41%</td>
<td>88909</td>
<td>2368</td>
</tr>
<tr>
<td>151</td>
<td>Aggressive 3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>118</td>
<td>11018</td>
<td>2%</td>
<td>5%</td>
<td>637042</td>
<td>666</td>
</tr>
<tr>
<td>152</td>
<td>TCP Connect 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>8</td>
<td>0%</td>
<td>19%</td>
<td>25994</td>
<td>245</td>
</tr>
<tr>
<td>153</td>
<td>Aggressive 6</td>
<td>5</td>
<td>101</td>
<td>7</td>
<td>100</td>
<td>9</td>
<td>121</td>
<td>24777</td>
<td>83%</td>
<td>59%</td>
<td>29869</td>
<td>1795</td>
</tr>
<tr>
<td>154</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>129</td>
<td>6</td>
<td>140</td>
<td>7</td>
<td>66</td>
<td>32106</td>
<td>32%</td>
<td>36%</td>
<td>101266</td>
<td>2365</td>
</tr>
<tr>
<td>155</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>3</td>
<td>62</td>
<td>13939</td>
<td>14%</td>
<td>33%</td>
<td>96538</td>
<td>7855</td>
</tr>
<tr>
<td>156</td>
<td>TCP Connect 6</td>
<td>6</td>
<td>23</td>
<td>5</td>
<td>25</td>
<td>7</td>
<td>35</td>
<td>8337</td>
<td>25%</td>
<td>16%</td>
<td>33002</td>
<td>188</td>
</tr>
<tr>
<td>157</td>
<td>Control Group 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>5%</td>
<td>1703</td>
<td>1341</td>
</tr>
<tr>
<td>158</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>122</td>
<td>32463</td>
<td>0%</td>
<td>5%</td>
<td>629445</td>
<td>1035</td>
</tr>
</tbody>
</table>
Table 13 continued from previous page

<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>159</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>55</td>
<td>5</td>
<td>40</td>
<td>7</td>
<td>38</td>
<td>8</td>
<td>17989</td>
<td>7239</td>
<td>49%</td>
<td>20%</td>
<td>36566</td>
<td>25</td>
</tr>
<tr>
<td>160</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>31</td>
<td>6</td>
<td>13178</td>
<td>18013</td>
<td>19%</td>
<td>26%</td>
<td>70151</td>
<td>2369</td>
</tr>
<tr>
<td>161</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>28</td>
<td>5</td>
<td>27</td>
<td>7</td>
<td>113</td>
<td>12</td>
<td>10649</td>
<td>31231</td>
<td>2%</td>
<td>5%</td>
<td>641598</td>
<td>1095</td>
</tr>
<tr>
<td>162</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>81</td>
<td>0%</td>
<td>4%</td>
<td>2128</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>163</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>5</td>
<td>32</td>
<td>6</td>
<td>12056</td>
<td>18171</td>
<td>17%</td>
<td>26%</td>
<td>70605</td>
<td>2366</td>
</tr>
<tr>
<td>164</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>94</td>
<td>0%</td>
<td>5%</td>
<td>1815</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>140</td>
<td>0%</td>
<td>6%</td>
<td>2354</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>166</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>0%</td>
<td>7%</td>
<td>1163</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>167</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>0%</td>
<td>5%</td>
<td>1404</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>168</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89</td>
<td>0%</td>
<td>8%</td>
<td>1099</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>169</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>77</td>
<td>0%</td>
<td>7%</td>
<td>1114</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>170</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>47</td>
<td>7</td>
<td>47</td>
<td>9</td>
<td>42</td>
<td>11</td>
<td>10987</td>
<td>30930</td>
<td>2%</td>
<td>5%</td>
<td>640906</td>
<td>653</td>
</tr>
<tr>
<td>171</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>12</td>
<td>12157</td>
<td>32081</td>
<td>2%</td>
<td>5%</td>
<td>641763</td>
<td>655</td>
</tr>
<tr>
<td>172</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>146</td>
<td>7</td>
<td>121</td>
<td>9</td>
<td>123</td>
<td>12</td>
<td>38851</td>
<td>17979</td>
<td>95%</td>
<td>44%</td>
<td>40997</td>
<td>1810</td>
</tr>
<tr>
<td>173</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>6</td>
<td>18042</td>
<td>0%</td>
<td>31%</td>
<td>58041</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>7461</td>
<td>18246</td>
<td>11%</td>
<td>26%</td>
<td>70969</td>
<td>4842</td>
</tr>
<tr>
<td>175</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>191</td>
<td>0%</td>
<td>14%</td>
<td>1348</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>121</td>
<td>12</td>
<td>32360</td>
<td>0%</td>
<td>5%</td>
<td>643431</td>
<td>701</td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>5</td>
<td>23</td>
<td>6</td>
<td>4535</td>
<td>2192</td>
<td>14%</td>
<td>7%</td>
<td>32258</td>
<td>493</td>
</tr>
<tr>
<td>178</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>5</td>
<td>34</td>
<td>6</td>
<td>13252</td>
<td>18704</td>
<td>19%</td>
<td>26%</td>
<td>70878</td>
<td>2367</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>179</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>176</td>
<td>7</td>
<td>149</td>
<td>10</td>
<td>124</td>
<td>10</td>
<td>30504</td>
<td>18477</td>
<td>69%</td>
<td>42%</td>
<td>44339</td>
<td>1809</td>
</tr>
<tr>
<td>180</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>21</td>
<td>5</td>
<td>17</td>
<td>7</td>
<td>36</td>
<td>8</td>
<td>5538</td>
<td>6557</td>
<td>15%</td>
<td>18%</td>
<td>36719</td>
<td>21</td>
</tr>
<tr>
<td>181</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>122</td>
<td>12</td>
<td>0</td>
<td>32033</td>
<td>0%</td>
<td>5%</td>
<td>643158</td>
<td>593</td>
</tr>
<tr>
<td>182</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>53</td>
<td>7</td>
<td>23947</td>
<td>12114</td>
<td>97%</td>
<td>49%</td>
<td>24802</td>
<td>1854</td>
</tr>
<tr>
<td>183</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>57</td>
<td>5</td>
<td>42</td>
<td>7</td>
<td>40</td>
<td>8</td>
<td>16630</td>
<td>7488</td>
<td>45%</td>
<td>20%</td>
<td>36608</td>
<td>26</td>
</tr>
<tr>
<td>184</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>6</td>
<td>0</td>
<td>18170</td>
<td>0%</td>
<td>29%</td>
<td>63306</td>
<td>3548</td>
</tr>
<tr>
<td>185</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>23</td>
<td>5</td>
<td>23</td>
<td>7</td>
<td>122</td>
<td>12</td>
<td>7402</td>
<td>32247</td>
<td>1%</td>
<td>5%</td>
<td>634299</td>
<td>1902</td>
</tr>
<tr>
<td>186</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>219</td>
<td>7</td>
<td>176</td>
<td>9</td>
<td>184</td>
<td>12</td>
<td>30316</td>
<td>18705</td>
<td>68%</td>
<td>42%</td>
<td>44587</td>
<td>1819</td>
</tr>
<tr>
<td>187</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>65</td>
<td>5</td>
<td>44</td>
<td>7</td>
<td>38</td>
<td>8</td>
<td>19160</td>
<td>7960</td>
<td>52%</td>
<td>21%</td>
<td>37108</td>
<td>34</td>
</tr>
<tr>
<td>188</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>0%</td>
<td>0%</td>
<td>6%</td>
<td>1639</td>
<td>1341</td>
</tr>
<tr>
<td>189</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>67</td>
<td>0%</td>
<td>0%</td>
<td>4%</td>
<td>1630</td>
<td>1341</td>
</tr>
<tr>
<td>190</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>62</td>
<td>8</td>
<td>19824</td>
<td>36130</td>
<td>22%</td>
<td>41%</td>
<td>88777</td>
<td>2366</td>
</tr>
<tr>
<td>191</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>47</td>
<td>6</td>
<td>50</td>
<td>8</td>
<td>55</td>
<td>7</td>
<td>12240</td>
<td>11964</td>
<td>54%</td>
<td>53%</td>
<td>22766</td>
<td>942</td>
</tr>
<tr>
<td>192</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>73</td>
<td>0%</td>
<td>0%</td>
<td>4%</td>
<td>1680</td>
<td>1341</td>
</tr>
<tr>
<td>193</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>94</td>
<td>0%</td>
<td>0%</td>
<td>6%</td>
<td>1686</td>
<td>1341</td>
</tr>
<tr>
<td>194</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>18268</td>
<td>0%</td>
<td>32%</td>
<td>57911</td>
<td>2365</td>
</tr>
<tr>
<td>195</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>94</td>
<td>7</td>
<td>88</td>
<td>7</td>
<td>62</td>
<td>8</td>
<td>24616</td>
<td>13503</td>
<td>54%</td>
<td>29%</td>
<td>45843</td>
<td>1747</td>
</tr>
<tr>
<td>196</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>28</td>
<td>5</td>
<td>25</td>
<td>7</td>
<td>30</td>
<td>8</td>
<td>8641</td>
<td>6581</td>
<td>24%</td>
<td>18%</td>
<td>36486</td>
<td>27</td>
</tr>
<tr>
<td>197</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>222</td>
<td>10</td>
<td>0</td>
<td>15258</td>
<td>0%</td>
<td>45%</td>
<td>34217</td>
<td>577</td>
</tr>
<tr>
<td>198</td>
<td>Aggressive</td>
<td>3</td>
<td>1</td>
<td>126</td>
<td>9</td>
<td>120</td>
<td>9</td>
<td>105</td>
<td>12</td>
<td>8073</td>
<td>31319</td>
<td>1%</td>
<td>6%</td>
<td>552845</td>
<td>910</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>199</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>667</td>
<td>0%</td>
<td>20%</td>
<td>3260</td>
<td>1341</td>
</tr>
<tr>
<td>200</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>22</td>
<td>5</td>
<td>25</td>
<td>7</td>
<td>45</td>
<td>0</td>
<td>0</td>
<td>8132</td>
<td>25%</td>
<td>33%</td>
<td>32216</td>
<td>169</td>
</tr>
<tr>
<td>201</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>108</td>
<td>6</td>
<td>114</td>
<td>7</td>
<td>65</td>
<td>8</td>
<td>0</td>
<td>19828</td>
<td>22%</td>
<td>41%</td>
<td>89449</td>
<td>2369</td>
</tr>
<tr>
<td>202</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0%</td>
<td>5%</td>
<td>1876</td>
<td>1341</td>
</tr>
<tr>
<td>203</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>31</td>
<td>8</td>
<td>0</td>
<td>2490</td>
<td>7%</td>
<td>6%</td>
<td>36686</td>
<td>386</td>
</tr>
<tr>
<td>204</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>8</td>
<td>0</td>
<td>4125</td>
<td>0%</td>
<td>12%</td>
<td>33202</td>
<td>467</td>
</tr>
<tr>
<td>205</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>71</td>
<td>8</td>
<td>0</td>
<td>18208</td>
<td>18%</td>
<td>36%</td>
<td>100925</td>
<td>8985</td>
</tr>
<tr>
<td>206</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>184</td>
<td>7</td>
<td>157</td>
<td>9</td>
<td>65</td>
<td>7</td>
<td>0</td>
<td>36488</td>
<td>97%</td>
<td>34%</td>
<td>37611</td>
<td>1106</td>
</tr>
<tr>
<td>207</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>104</td>
<td>8</td>
<td>95</td>
<td>8</td>
<td>88</td>
<td>9</td>
<td>0</td>
<td>17461</td>
<td>72%</td>
<td>49%</td>
<td>24233</td>
<td>953</td>
</tr>
<tr>
<td>208</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>0%</td>
<td>5%</td>
<td>1750</td>
<td>1341</td>
</tr>
<tr>
<td>209</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>30</td>
<td>5</td>
<td>39</td>
<td>8</td>
<td>61</td>
<td>8</td>
<td>0</td>
<td>31890</td>
<td>31%</td>
<td>36%</td>
<td>101483</td>
<td>2369</td>
</tr>
<tr>
<td>210</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>10</td>
<td>0</td>
<td>5200</td>
<td>0%</td>
<td>18%</td>
<td>29611</td>
<td>350</td>
</tr>
<tr>
<td>211</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>67</td>
<td>8</td>
<td>0</td>
<td>19935</td>
<td>22%</td>
<td>40%</td>
<td>89865</td>
<td>2368</td>
</tr>
<tr>
<td>212</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>157</td>
<td>7</td>
<td>128</td>
<td>9</td>
<td>62</td>
<td>7</td>
<td>0</td>
<td>33662</td>
<td>95%</td>
<td>36%</td>
<td>35500</td>
<td>1116</td>
</tr>
<tr>
<td>213</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>24</td>
<td>5</td>
<td>25</td>
<td>7</td>
<td>121</td>
<td>11</td>
<td>0</td>
<td>7981</td>
<td>1%</td>
<td>5%</td>
<td>640899</td>
<td>1198</td>
</tr>
<tr>
<td>214</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>8</td>
<td>0</td>
<td>4763</td>
<td>0%</td>
<td>15%</td>
<td>31533</td>
<td>369</td>
</tr>
<tr>
<td>215</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>36</td>
<td>8</td>
<td>0</td>
<td>2376</td>
<td>6%</td>
<td>20%</td>
<td>36809</td>
<td>22</td>
</tr>
<tr>
<td>216</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>43</td>
<td>7</td>
<td>44</td>
<td>9</td>
<td>112</td>
<td>12</td>
<td>0</td>
<td>12380</td>
<td>2%</td>
<td>5%</td>
<td>626577</td>
<td>1156</td>
</tr>
<tr>
<td>217</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>39</td>
<td>7</td>
<td>42</td>
<td>9</td>
<td>108</td>
<td>12</td>
<td>0</td>
<td>11487</td>
<td>2%</td>
<td>5%</td>
<td>635276</td>
<td>790</td>
</tr>
<tr>
<td>218</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>0%</td>
<td>4%</td>
<td>1784</td>
<td>1341</td>
</tr>
</tbody>
</table>
Table 13 continued from previous page

<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>219</td>
<td>Aggressive</td>
<td>3</td>
<td>91</td>
<td>7</td>
<td>89</td>
<td>9</td>
<td>43</td>
<td>11</td>
<td>11601</td>
<td>30455</td>
<td>2%</td>
<td>5%</td>
<td>635512</td>
<td>1226</td>
<td></td>
</tr>
<tr>
<td>220</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>45</td>
<td>5</td>
<td>56</td>
<td>7</td>
<td>67</td>
<td>8</td>
<td>31991</td>
<td>35993</td>
<td>31%</td>
<td>35%</td>
<td>102228</td>
<td>2396</td>
<td></td>
</tr>
<tr>
<td>221</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>0%</td>
<td>3%</td>
<td>2763</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>Aggressive</td>
<td>6</td>
<td>119</td>
<td>7</td>
<td>121</td>
<td>9</td>
<td>241</td>
<td>12</td>
<td>24515</td>
<td>16528</td>
<td>72%</td>
<td>48%</td>
<td>34137</td>
<td>520</td>
<td></td>
</tr>
<tr>
<td>223</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>8</td>
<td>4164</td>
<td>0%</td>
<td>15%</td>
<td>27524</td>
<td>232</td>
<td></td>
<td></td>
</tr>
<tr>
<td>224</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>12</td>
<td>3</td>
<td>13</td>
<td>4</td>
<td>56</td>
<td>9</td>
<td>11827</td>
<td>29122</td>
<td>12%</td>
<td>30%</td>
<td>95912</td>
<td>7710</td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>6</td>
<td>18251</td>
<td>0%</td>
<td>31%</td>
<td>58184</td>
<td>2367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>226</td>
<td>Aggressive</td>
<td>9</td>
<td>110</td>
<td>9</td>
<td>84</td>
<td>9</td>
<td>127</td>
<td>10</td>
<td>30828</td>
<td>17955</td>
<td>89%</td>
<td>52%</td>
<td>34637</td>
<td>1803</td>
<td></td>
</tr>
<tr>
<td>227</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>415</td>
<td>0%</td>
<td>13%</td>
<td>3174</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>228</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>18100</td>
<td>0%</td>
<td>31%</td>
<td>58233</td>
<td>2365</td>
<td></td>
<td></td>
</tr>
<tr>
<td>229</td>
<td>Aggressive</td>
<td>9</td>
<td>131</td>
<td>7</td>
<td>113</td>
<td>10</td>
<td>123</td>
<td>10</td>
<td>32154</td>
<td>16534</td>
<td>96%</td>
<td>49%</td>
<td>33417</td>
<td>1066</td>
<td></td>
</tr>
<tr>
<td>230</td>
<td>TCP Connect</td>
<td>9</td>
<td>14</td>
<td>5</td>
<td>12</td>
<td>5</td>
<td>41</td>
<td>8</td>
<td>2890</td>
<td>6663</td>
<td>8%</td>
<td>18%</td>
<td>36512</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>231</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>8</td>
<td>2524</td>
<td>0%</td>
<td>7%</td>
<td>34101</td>
<td>547</td>
<td></td>
<td></td>
</tr>
<tr>
<td>232</td>
<td>Aggressive</td>
<td>3</td>
<td>41</td>
<td>6</td>
<td>46</td>
<td>9</td>
<td>107</td>
<td>12</td>
<td>11240</td>
<td>31468</td>
<td>2%</td>
<td>5%</td>
<td>631313</td>
<td>843</td>
<td></td>
</tr>
<tr>
<td>233</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>6</td>
<td>4072</td>
<td>0%</td>
<td>13%</td>
<td>30692</td>
<td>361</td>
<td></td>
<td></td>
</tr>
<tr>
<td>234</td>
<td>Aggressive</td>
<td>3</td>
<td>42</td>
<td>7</td>
<td>48</td>
<td>9</td>
<td>113</td>
<td>12</td>
<td>11541</td>
<td>32069</td>
<td>2%</td>
<td>5%</td>
<td>634563</td>
<td>1833</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32345</td>
<td>0%</td>
<td>1247%</td>
<td>2593</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>236</td>
<td>TCP Connect</td>
<td>6</td>
<td>33</td>
<td>5</td>
<td>35</td>
<td>7</td>
<td>37</td>
<td>8</td>
<td>10589</td>
<td>8003</td>
<td>33%</td>
<td>25%</td>
<td>32002</td>
<td>137</td>
<td></td>
</tr>
<tr>
<td>237</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>77</td>
<td>0%</td>
<td>5%</td>
<td>1668</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>238</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>88</td>
<td>0%</td>
<td>5%</td>
<td>1670</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>239</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>77</td>
<td>0%</td>
<td>4%</td>
<td></td>
<td>1714</td>
<td>1341</td>
</tr>
<tr>
<td>240</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>189</td>
<td>0%</td>
<td>10%</td>
<td></td>
<td>1882</td>
<td>1341</td>
</tr>
<tr>
<td>241</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>33</td>
<td>8</td>
<td>1985</td>
<td>6%</td>
<td>8%</td>
<td></td>
<td>33381</td>
<td>331</td>
</tr>
<tr>
<td>242</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>18125</td>
<td>0%</td>
<td>31%</td>
<td></td>
<td>57950</td>
<td>2368</td>
</tr>
<tr>
<td>243</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>119</td>
<td>12</td>
<td>31340</td>
<td>0%</td>
<td>5%</td>
<td></td>
<td>595363</td>
<td>657</td>
</tr>
<tr>
<td>244</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>73</td>
<td>0%</td>
<td>4%</td>
<td></td>
<td>1856</td>
<td>1341</td>
</tr>
<tr>
<td>245</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>76</td>
<td>0%</td>
<td>4%</td>
<td></td>
<td>1721</td>
<td>1341</td>
</tr>
<tr>
<td>246</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>93</td>
<td>7</td>
<td>88</td>
<td>9</td>
<td>26</td>
<td>7</td>
<td>21261</td>
<td>63%</td>
<td>36%</td>
<td></td>
<td>33586</td>
<td>1102</td>
</tr>
<tr>
<td>247</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>113</td>
<td>12</td>
<td>32255</td>
<td>0%</td>
<td>5%</td>
<td></td>
<td>612987</td>
<td>556</td>
</tr>
<tr>
<td>248</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>134</td>
<td>8</td>
<td>123</td>
<td>11</td>
<td>129</td>
<td>11</td>
<td>23659</td>
<td>65%</td>
<td>50%</td>
<td></td>
<td>36619</td>
<td>1803</td>
</tr>
<tr>
<td>249</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>44</td>
<td>7</td>
<td>50</td>
<td>9</td>
<td>118</td>
<td>12</td>
<td>10722</td>
<td>33%</td>
<td>2%</td>
<td></td>
<td>628083</td>
<td>969</td>
</tr>
<tr>
<td>250</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>30</td>
<td>8</td>
<td>32</td>
<td>8</td>
<td>15</td>
<td>8</td>
<td>10666</td>
<td>23%</td>
<td>94%</td>
<td></td>
<td>47052</td>
<td>4293</td>
</tr>
<tr>
<td>251</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>18157</td>
<td>0%</td>
<td>31%</td>
<td></td>
<td>58055</td>
<td>2367</td>
</tr>
<tr>
<td>252</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>36276</td>
<td>0%</td>
<td>62%</td>
<td></td>
<td>58487</td>
<td>2369</td>
</tr>
<tr>
<td>253</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>7</td>
<td>4096</td>
<td>0%</td>
<td>13%</td>
<td></td>
<td>30987</td>
<td>345</td>
</tr>
<tr>
<td>254</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>78</td>
<td>0%</td>
<td>3%</td>
<td></td>
<td>2343</td>
<td>1341</td>
</tr>
<tr>
<td>255</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>8</td>
<td>18145</td>
<td>0%</td>
<td>31%</td>
<td></td>
<td>58453</td>
<td>2369</td>
</tr>
<tr>
<td>256</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>30</td>
<td>7</td>
<td>28</td>
<td>7</td>
<td>11</td>
<td>5</td>
<td>8162</td>
<td>24%</td>
<td>67%</td>
<td></td>
<td>33560</td>
<td>126</td>
</tr>
<tr>
<td>257</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>4%</td>
<td></td>
<td>1838</td>
<td>1341</td>
</tr>
<tr>
<td>258</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>0%</td>
<td>4%</td>
<td></td>
<td>1919</td>
<td>1341</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>----</td>
</tr>
<tr>
<td>259</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>153</td>
<td>7</td>
<td>134</td>
<td>9</td>
<td>125</td>
<td>10</td>
<td>38937</td>
<td>18166</td>
<td>93%</td>
<td>43%</td>
<td>42059</td>
<td>1815</td>
</tr>
<tr>
<td>260</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>8</td>
<td>0</td>
<td>4333</td>
<td>0%</td>
<td>12%</td>
<td>34864</td>
<td>710</td>
<td></td>
</tr>
<tr>
<td>261</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>94</td>
<td>0%</td>
<td>5%</td>
<td>2027</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>262</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>70</td>
<td>7</td>
<td>73</td>
<td>7</td>
<td>113</td>
<td>11</td>
<td>11198</td>
<td>31090</td>
<td>2%</td>
<td>5%</td>
<td>623431</td>
<td>647</td>
</tr>
<tr>
<td>263</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>31</td>
<td>6</td>
<td>13468</td>
<td>18135</td>
<td>17%</td>
<td>23%</td>
<td>77784</td>
<td>2367</td>
</tr>
<tr>
<td>264</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>51</td>
<td>8</td>
<td>0</td>
<td>3966</td>
<td>0%</td>
<td>14%</td>
<td>29352</td>
<td>403</td>
<td></td>
</tr>
<tr>
<td>265</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>76</td>
<td>0%</td>
<td>4%</td>
<td>1918</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>266</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>12</td>
<td>10279</td>
<td>31317</td>
<td>2%</td>
<td>5%</td>
<td>641550</td>
<td>639</td>
</tr>
<tr>
<td>267</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>4</td>
<td>26</td>
<td>5</td>
<td>7091</td>
<td>18248</td>
<td>8%</td>
<td>21%</td>
<td>85346</td>
<td>6785</td>
</tr>
<tr>
<td>268</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>130</td>
<td>7</td>
<td>118</td>
<td>10</td>
<td>112</td>
<td>10</td>
<td>37829</td>
<td>17290</td>
<td>93%</td>
<td>42%</td>
<td>40685</td>
<td>1801</td>
</tr>
<tr>
<td>269</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>113</td>
<td>12</td>
<td>30741</td>
<td>0%</td>
<td>5%</td>
<td>634409</td>
<td>991</td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>41</td>
<td>8</td>
<td>3000</td>
<td>6372</td>
<td>8%</td>
<td>17%</td>
<td>38513</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>271</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>70</td>
<td>7</td>
<td>73</td>
<td>10</td>
<td>28</td>
<td>8</td>
<td>10369</td>
<td>8125</td>
<td>31%</td>
<td>24%</td>
<td>33185</td>
<td>3742</td>
</tr>
<tr>
<td>272</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>21</td>
<td>5</td>
<td>22</td>
<td>6</td>
<td>32</td>
<td>6</td>
<td>13465</td>
<td>18143</td>
<td>18%</td>
<td>24%</td>
<td>76634</td>
<td>2367</td>
</tr>
<tr>
<td>273</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>84</td>
<td>0%</td>
<td>5%</td>
<td>1831</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>274</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>21</td>
<td>5</td>
<td>25</td>
<td>6</td>
<td>62</td>
<td>8</td>
<td>19986</td>
<td>36280</td>
<td>21%</td>
<td>38%</td>
<td>96653</td>
<td>2366</td>
</tr>
<tr>
<td>275</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>82</td>
<td>7</td>
<td>76</td>
<td>9</td>
<td>123</td>
<td>10</td>
<td>22437</td>
<td>17137</td>
<td>70%</td>
<td>53%</td>
<td>32080</td>
<td>1796</td>
</tr>
<tr>
<td>276</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>36</td>
<td>8</td>
<td>13375</td>
<td>18266</td>
<td>17%</td>
<td>24%</td>
<td>77461</td>
<td>2369</td>
</tr>
<tr>
<td>277</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>33</td>
<td>8</td>
<td>13344</td>
<td>18078</td>
<td>17%</td>
<td>24%</td>
<td>76664</td>
<td>2369</td>
</tr>
<tr>
<td>278</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>35</td>
<td>5</td>
<td>30</td>
<td>7</td>
<td>42</td>
<td>8</td>
<td>10420</td>
<td>7431</td>
<td>27%</td>
<td>19%</td>
<td>38563</td>
<td>22</td>
</tr>
</tbody>
</table>
Table 13 continued from previous page

<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>279</td>
<td>Aggressive</td>
<td>3</td>
<td>0</td>
<td>60</td>
<td>7</td>
<td>64</td>
<td>9</td>
<td>101</td>
<td>12</td>
<td>13963</td>
<td>31282</td>
<td>2%</td>
<td>5%</td>
<td>635043</td>
<td>643</td>
</tr>
<tr>
<td>280</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>81</td>
<td>0%</td>
<td>5%</td>
<td>1624</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>281</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>114</td>
<td>6</td>
<td>92</td>
<td>9</td>
<td>73</td>
<td>7</td>
<td>24621</td>
<td>12493</td>
<td>94%</td>
<td>48%</td>
<td>26233</td>
<td>938</td>
</tr>
<tr>
<td>282</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>231</td>
<td>7</td>
<td>202</td>
<td>9</td>
<td>189</td>
<td>10</td>
<td>40885</td>
<td>18035</td>
<td>93%</td>
<td>41%</td>
<td>43828</td>
<td>1820</td>
</tr>
<tr>
<td>283</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>57</td>
<td>8</td>
<td>21550</td>
<td>36430</td>
<td>21%</td>
<td>36%</td>
<td>100883</td>
<td>4659</td>
</tr>
<tr>
<td>284</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>21</td>
<td>5</td>
<td>22</td>
<td>7</td>
<td>43</td>
<td>8</td>
<td>6104</td>
<td>9148</td>
<td>17%</td>
<td>26%</td>
<td>35026</td>
<td>203</td>
</tr>
<tr>
<td>285</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>0%</td>
<td>4%</td>
<td>1700</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>286</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>6</td>
<td>3637</td>
<td>0%</td>
<td>12%</td>
<td>30859</td>
<td>508</td>
<td></td>
</tr>
<tr>
<td>287</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>16</td>
<td>6</td>
<td>2414</td>
<td>1195</td>
<td>9%</td>
<td>4%</td>
<td>27226</td>
<td>60</td>
</tr>
<tr>
<td>288</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>26</td>
<td>5</td>
<td>24</td>
<td>7</td>
<td>34</td>
<td>8</td>
<td>8376</td>
<td>6666</td>
<td>26%</td>
<td>20%</td>
<td>32762</td>
<td>39</td>
</tr>
<tr>
<td>289</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>40</td>
<td>7</td>
<td>46</td>
<td>9</td>
<td>103</td>
<td>12</td>
<td>11202</td>
<td>30472</td>
<td>2%</td>
<td>5%</td>
<td>639607</td>
<td>855</td>
</tr>
<tr>
<td>290</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>8</td>
<td>3873</td>
<td>0%</td>
<td>12%</td>
<td>31268</td>
<td>315</td>
<td></td>
</tr>
<tr>
<td>291</td>
<td>TCP Connect</td>
<td>3</td>
<td>0</td>
<td>26</td>
<td>5</td>
<td>37</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>14062</td>
<td>10565</td>
<td>29%</td>
<td>22%</td>
<td>48180</td>
<td>2727</td>
</tr>
<tr>
<td>292</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>65</td>
<td>8</td>
<td>18087</td>
<td>36178</td>
<td>17%</td>
<td>34%</td>
<td>107584</td>
<td>2364</td>
</tr>
<tr>
<td>293</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>69</td>
<td>0%</td>
<td>4%</td>
<td>1663</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>294</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>46</td>
<td>7</td>
<td>45</td>
<td>9</td>
<td>236</td>
<td>10</td>
<td>8961</td>
<td>15931</td>
<td>27%</td>
<td>48%</td>
<td>33465</td>
<td>406</td>
</tr>
<tr>
<td>295</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>33</td>
<td>7</td>
<td>60</td>
<td>8</td>
<td>32040</td>
<td>36261</td>
<td>30%</td>
<td>34%</td>
<td>107230</td>
<td>2368</td>
</tr>
<tr>
<td>296</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>655</td>
<td>0%</td>
<td>19%</td>
<td>3416</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>297</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>15</td>
<td>5</td>
<td>19</td>
<td>7</td>
<td>46</td>
<td>8</td>
<td>5862</td>
<td>8239</td>
<td>16%</td>
<td>22%</td>
<td>37122</td>
<td>431</td>
</tr>
<tr>
<td>298</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>8</td>
<td>3904</td>
<td>0%</td>
<td>15%</td>
<td>26627</td>
<td>293</td>
<td></td>
</tr>
</tbody>
</table>
Table 13 continued from previous page

<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>299</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>6</td>
<td>0</td>
<td>18051</td>
<td>0%</td>
<td>28%</td>
<td>63930</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>Control Group 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96</td>
<td>0%</td>
<td>5%</td>
<td>1802</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>301</td>
<td>Aggressive 3</td>
<td>3</td>
<td>40</td>
<td>7</td>
<td>43</td>
<td>7</td>
<td>105</td>
<td>12</td>
<td>11784</td>
<td>31123</td>
<td>2%</td>
<td>5%</td>
<td>622328</td>
<td>814</td>
<td></td>
</tr>
<tr>
<td>302</td>
<td>Control Group 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0%</td>
<td>5%</td>
<td>2006</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>303</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>6</td>
<td>0</td>
<td>18111</td>
<td>0%</td>
<td>28%</td>
<td>64609</td>
<td>2368</td>
<td></td>
</tr>
<tr>
<td>304</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>11</td>
<td>5</td>
<td>13374</td>
<td>36263</td>
<td>17%</td>
<td>47%</td>
<td>76766</td>
<td>2366</td>
<td></td>
</tr>
<tr>
<td>305</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>35</td>
<td>7</td>
<td>59</td>
<td>8</td>
<td>31378</td>
<td>36098</td>
<td>28%</td>
<td>32%</td>
<td>113753</td>
<td>5503</td>
<td></td>
</tr>
<tr>
<td>306</td>
<td>Aggressive 6</td>
<td>6</td>
<td>99</td>
<td>7</td>
<td>88</td>
<td>9</td>
<td>47</td>
<td>7</td>
<td>20432</td>
<td>12197</td>
<td>82%</td>
<td>49%</td>
<td>25016</td>
<td>1059</td>
<td></td>
</tr>
<tr>
<td>307</td>
<td>Control Group 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>67</td>
<td>0%</td>
<td>4%</td>
<td>1759</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>308</td>
<td>Control Group 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>85</td>
<td>0%</td>
<td>4%</td>
<td>1944</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>309</td>
<td>TCP Connect 3</td>
<td>3</td>
<td>18</td>
<td>5</td>
<td>16</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>6146</td>
<td>3578</td>
<td>22%</td>
<td>13%</td>
<td>27547</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>310</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>112</td>
<td>12</td>
<td>0</td>
<td>31399</td>
<td>0%</td>
<td>5%</td>
<td>628763</td>
<td>614</td>
<td></td>
</tr>
<tr>
<td>311</td>
<td>Control Group 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97</td>
<td>0%</td>
<td>5%</td>
<td>1895</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>312</td>
<td>Control Group 6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>74</td>
<td>0%</td>
<td>4%</td>
<td>1889</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>313</td>
<td>Control Group 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>168</td>
<td>0%</td>
<td>9%</td>
<td>1826</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>314</td>
<td>TCP Connect 3</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>22</td>
<td>6</td>
<td>2241</td>
<td>2545</td>
<td>8%</td>
<td>9%</td>
<td>26881</td>
<td>41</td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>Control Group 9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0%</td>
<td>5%</td>
<td>1925</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>316</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>105</td>
<td>12</td>
<td>0</td>
<td>31669</td>
<td>0%</td>
<td>5%</td>
<td>614810</td>
<td>849</td>
<td></td>
</tr>
<tr>
<td>317</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>5</td>
<td>65</td>
<td>8</td>
<td>19993</td>
<td>36001</td>
<td>21%</td>
<td>38%</td>
<td>95888</td>
<td>2369</td>
<td></td>
</tr>
<tr>
<td>318</td>
<td>Control Group 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>79</td>
<td>0%</td>
<td>4%</td>
<td>1888</td>
<td>1341</td>
<td></td>
</tr>
</tbody>
</table>
Table 13 continued from previous page

<table>
<thead>
<tr>
<th>Trial</th>
<th>$ST$</th>
<th>$HP$</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>319</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>90</td>
<td>6</td>
<td>96</td>
<td>7</td>
<td>34</td>
<td>8</td>
<td>13433</td>
<td>18197</td>
<td>18%</td>
<td>24%</td>
<td>76571</td>
<td>2367</td>
</tr>
<tr>
<td>320</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>34</td>
<td>7</td>
<td>66</td>
<td>8</td>
<td>32035</td>
<td>36181</td>
<td>29%</td>
<td>33%</td>
<td>108858</td>
<td>2368</td>
</tr>
<tr>
<td>321</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>0%</td>
<td>4%</td>
<td>2039</td>
<td>1341</td>
</tr>
<tr>
<td>322</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>32</td>
<td>5</td>
<td>36</td>
<td>5</td>
<td>20</td>
<td>8</td>
<td>15965</td>
<td>36234</td>
<td>17%</td>
<td>38%</td>
<td>95469</td>
<td>2365</td>
</tr>
<tr>
<td>323</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>188</td>
<td>8</td>
<td>166</td>
<td>10</td>
<td>129</td>
<td>11</td>
<td>41038</td>
<td>17657</td>
<td>93%</td>
<td>40%</td>
<td>44184</td>
<td>1821</td>
</tr>
<tr>
<td>324</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>0%</td>
<td>0%</td>
<td>4%</td>
<td>2394</td>
<td>1341</td>
</tr>
<tr>
<td>325</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>34</td>
<td>7</td>
<td>62</td>
<td>8</td>
<td>32021</td>
<td>36011</td>
<td>29%</td>
<td>33%</td>
<td>108974</td>
<td>2365</td>
</tr>
<tr>
<td>326</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>18</td>
<td>5</td>
<td>23</td>
<td>8</td>
<td>16006</td>
<td>72494</td>
<td>16%</td>
<td>75%</td>
<td>97299</td>
<td>2368</td>
</tr>
<tr>
<td>327</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>19</td>
<td>5</td>
<td>16</td>
<td>7</td>
<td>39</td>
<td>8</td>
<td>5310</td>
<td>11010</td>
<td>12%</td>
<td>26%</td>
<td>42974</td>
<td>414</td>
</tr>
<tr>
<td>328</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>108</td>
<td>8</td>
<td>99</td>
<td>10</td>
<td>121</td>
<td>11</td>
<td>26021</td>
<td>17738</td>
<td>83%</td>
<td>57%</td>
<td>31384</td>
<td>1800</td>
</tr>
<tr>
<td>329</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
<td>1726</td>
<td>1341</td>
</tr>
<tr>
<td>330</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>54</td>
<td>7</td>
<td>20217</td>
<td>12029</td>
<td>82%</td>
<td>49%</td>
<td>24647</td>
</tr>
<tr>
<td>331</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>6</td>
<td>13495</td>
<td>18172</td>
<td>18%</td>
<td>24%</td>
<td>76744</td>
<td>2367</td>
</tr>
<tr>
<td>332</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>115</td>
<td>7</td>
<td>118</td>
<td>9</td>
<td>270</td>
<td>10</td>
<td>20471</td>
<td>37090</td>
<td>5%</td>
<td>9%</td>
<td>402106</td>
<td>435</td>
</tr>
<tr>
<td>333</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>13</td>
<td>5</td>
<td>32</td>
<td>8</td>
<td>4387</td>
<td>3487</td>
<td>14%</td>
<td>11%</td>
<td>31504</td>
<td>290</td>
</tr>
<tr>
<td>334</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td>10</td>
<td>0</td>
<td>34635</td>
<td>0%</td>
<td>6%</td>
<td>623664</td>
<td>608</td>
</tr>
<tr>
<td>335</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>5</td>
<td>59</td>
<td>8</td>
<td>20028</td>
<td>36223</td>
<td>21%</td>
<td>38%</td>
<td>95552</td>
<td>2368</td>
</tr>
<tr>
<td>336</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>20</td>
<td>4</td>
<td>29</td>
<td>7</td>
<td>27</td>
<td>6</td>
<td>8568</td>
<td>18721</td>
<td>10%</td>
<td>23%</td>
<td>82583</td>
<td>5964</td>
</tr>
<tr>
<td>337</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>103</td>
<td>0%</td>
<td>0%</td>
<td>6%</td>
<td>1634</td>
<td>1341</td>
</tr>
<tr>
<td>338</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>194</td>
<td>9</td>
<td>175</td>
<td>9</td>
<td>83</td>
<td>9</td>
<td>34898</td>
<td>12110</td>
<td>96%</td>
<td>33%</td>
<td>36407</td>
<td>1099</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>339</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>106</td>
<td>12</td>
<td>0</td>
<td>31205</td>
<td>0%</td>
<td>5%</td>
<td>634197</td>
<td>610</td>
<td></td>
</tr>
<tr>
<td>340</td>
<td>Aggressive 9</td>
<td>8</td>
<td>136</td>
<td>7</td>
<td>119</td>
<td>9</td>
<td>55</td>
<td>7</td>
<td>32316</td>
<td>12068</td>
<td>95%</td>
<td>35%</td>
<td>34152</td>
<td>1107</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>6</td>
<td>0</td>
<td>18122</td>
<td>0%</td>
<td>28%</td>
<td>63660</td>
<td>2365</td>
<td></td>
</tr>
<tr>
<td>342</td>
<td>Aggressive 3</td>
<td>3</td>
<td>43</td>
<td>9</td>
<td>48</td>
<td>9</td>
<td>100</td>
<td>12</td>
<td>10602</td>
<td>30874</td>
<td>2%</td>
<td>5%</td>
<td>636944</td>
<td>660</td>
<td></td>
</tr>
<tr>
<td>343</td>
<td>TCP Connect 6</td>
<td>6</td>
<td>37</td>
<td>5</td>
<td>30</td>
<td>7</td>
<td>37</td>
<td>8</td>
<td>10525</td>
<td>8991</td>
<td>32%</td>
<td>27%</td>
<td>33078</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>344</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>20</td>
<td>4</td>
<td>35</td>
<td>7</td>
<td>68</td>
<td>8</td>
<td>26008</td>
<td>36143</td>
<td>24%</td>
<td>34%</td>
<td>107648</td>
<td>2369</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>113</td>
<td>12</td>
<td>0</td>
<td>31063</td>
<td>0%</td>
<td>5%</td>
<td>627436</td>
<td>2852</td>
<td></td>
</tr>
<tr>
<td>346</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>131</td>
<td>4</td>
<td>137</td>
<td>7</td>
<td>60</td>
<td>8</td>
<td>32048</td>
<td>36141</td>
<td>30%</td>
<td>34%</td>
<td>107316</td>
<td>2366</td>
<td></td>
</tr>
<tr>
<td>347</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>28</td>
<td>6</td>
<td>13447</td>
<td>18548</td>
<td>15%</td>
<td>21%</td>
<td>88230</td>
<td>2367</td>
<td></td>
</tr>
<tr>
<td>348</td>
<td>NIDS Avoidance 6</td>
<td>6</td>
<td>13</td>
<td>5</td>
<td>18</td>
<td>6</td>
<td>59</td>
<td>8</td>
<td>12304</td>
<td>36518</td>
<td>11%</td>
<td>33%</td>
<td>112233</td>
<td>1102</td>
<td></td>
</tr>
<tr>
<td>349</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>0</td>
<td>18614</td>
<td>0%</td>
<td>17%</td>
<td>112058</td>
<td>5860</td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>Control Group 9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>195</td>
<td>0%</td>
<td>10%</td>
<td>1972</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>351</td>
<td>Control Group 9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>0%</td>
<td>4%</td>
<td>2368</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>352</td>
<td>TCP Connect 3</td>
<td>3</td>
<td>18</td>
<td>5</td>
<td>20</td>
<td>5</td>
<td>22</td>
<td>8</td>
<td>6054</td>
<td>3768</td>
<td>19%</td>
<td>12%</td>
<td>31188</td>
<td>335</td>
<td></td>
</tr>
<tr>
<td>353</td>
<td>NIDS Avoidance 3</td>
<td>3</td>
<td>19</td>
<td>5</td>
<td>24</td>
<td>5</td>
<td>27</td>
<td>5</td>
<td>9310</td>
<td>18073</td>
<td>11%</td>
<td>21%</td>
<td>86161</td>
<td>4077</td>
<td></td>
</tr>
<tr>
<td>354</td>
<td>Control Group 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>83</td>
<td>0%</td>
<td>5%</td>
<td>1821</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>TCP Connect 9</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>41</td>
<td>8</td>
<td>2053</td>
<td>7648</td>
<td>5%</td>
<td>20%</td>
<td>38907</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>116</td>
<td>12</td>
<td>0</td>
<td>31734</td>
<td>0%</td>
<td>5%</td>
<td>624185</td>
<td>616</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>TCP Connect 9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>8</td>
<td>0</td>
<td>7076</td>
<td>0%</td>
<td>18%</td>
<td>38669</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>358</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>32</td>
<td>5</td>
<td>31</td>
<td>5</td>
<td>63</td>
<td>8</td>
<td>24826</td>
<td>35919</td>
<td>23%</td>
<td>33%</td>
<td>107681</td>
<td>2367</td>
<td></td>
</tr>
</tbody>
</table>
Table 13 continued from previous page

<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>SuP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>359</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>5</td>
<td>14</td>
<td>7</td>
<td>27</td>
<td>10</td>
<td>4436</td>
<td>3843</td>
<td>14%</td>
<td>12%</td>
<td>31436</td>
<td>291</td>
</tr>
<tr>
<td>360</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>96</td>
<td>7</td>
<td>94</td>
<td>9</td>
<td>60</td>
<td>7</td>
<td>20159</td>
<td>12622</td>
<td>8%</td>
<td>5%</td>
<td>244766</td>
<td>1133</td>
</tr>
<tr>
<td>361</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>12</td>
<td>5</td>
<td>27</td>
<td>8</td>
<td>4394</td>
<td>2310</td>
<td>15%</td>
<td>8%</td>
<td>29712</td>
<td>220</td>
</tr>
<tr>
<td>362</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97</td>
<td>0%</td>
<td>5%</td>
<td>1793</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>363</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>40</td>
<td>5</td>
<td>42</td>
<td>7</td>
<td>37</td>
<td>6</td>
<td>16204</td>
<td>9634</td>
<td>37%</td>
<td>22%</td>
<td>43952</td>
<td>1133</td>
</tr>
<tr>
<td>364</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>64</td>
<td>7</td>
<td>63</td>
<td>7</td>
<td>112</td>
<td>8</td>
<td>9671</td>
<td>12918</td>
<td>19%</td>
<td>26%</td>
<td>49775</td>
<td>4317</td>
</tr>
<tr>
<td>365</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>6</td>
<td>18100</td>
<td>0%</td>
<td>0%</td>
<td>27%</td>
<td>66522</td>
<td>2367</td>
</tr>
<tr>
<td>366</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>24</td>
<td>8</td>
<td>5880</td>
<td>4023</td>
<td>11%</td>
<td>7%</td>
<td>55036</td>
<td>3398</td>
</tr>
<tr>
<td>367</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>79</td>
<td>0%</td>
<td>4%</td>
<td>2185</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>368</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>81</td>
<td>0%</td>
<td>5%</td>
<td>1660</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>369</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>25</td>
<td>6</td>
<td>16003</td>
<td>36208</td>
<td>17%</td>
<td>38%</td>
<td>95336</td>
<td>2367</td>
</tr>
<tr>
<td>370</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>6</td>
<td>0</td>
<td>6548</td>
<td>0%</td>
<td>24%</td>
<td>27640</td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>371</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>111</td>
<td>12</td>
<td>0</td>
<td>31316</td>
<td>0%</td>
<td>5%</td>
<td>628280</td>
<td>617</td>
<td></td>
<td></td>
</tr>
<tr>
<td>372</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>5</td>
<td>0</td>
<td>49444</td>
<td>0%</td>
<td>77%</td>
<td>64362</td>
<td>2367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>373</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>41</td>
<td>7</td>
<td>47</td>
<td>9</td>
<td>101</td>
<td>11</td>
<td>10525</td>
<td>31215</td>
<td>2%</td>
<td>5%</td>
<td>633644</td>
<td>634</td>
</tr>
<tr>
<td>374</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>112</td>
<td>7</td>
<td>109</td>
<td>9</td>
<td>121</td>
<td>12</td>
<td>24876</td>
<td>17773</td>
<td>69%</td>
<td>49%</td>
<td>36026</td>
<td>1798</td>
</tr>
<tr>
<td>375</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>77</td>
<td>0%</td>
<td>0%</td>
<td>1680</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>376</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>153</td>
<td>7</td>
<td>137</td>
<td>9</td>
<td>129</td>
<td>12</td>
<td>28284</td>
<td>18672</td>
<td>67%</td>
<td>44%</td>
<td>42108</td>
<td>1811</td>
</tr>
<tr>
<td>377</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>78</td>
<td>0%</td>
<td>3%</td>
<td>2959</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>378</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>69</td>
<td>7</td>
<td>61</td>
<td>8</td>
<td>48</td>
<td>8</td>
<td>22333</td>
<td>12944</td>
<td>44%</td>
<td>25%</td>
<td>51282</td>
<td>1197</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>379</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>13</td>
<td>5</td>
<td>16</td>
<td>7</td>
<td>63</td>
<td>7</td>
<td>5135</td>
<td>12788</td>
<td>15%</td>
<td>36%</td>
<td>35075</td>
<td>1113</td>
</tr>
<tr>
<td>380</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>17</td>
<td>5</td>
<td>64</td>
<td>8</td>
<td>19975</td>
<td>36099</td>
<td>21%</td>
<td>38%</td>
<td>95653</td>
<td>2365</td>
</tr>
<tr>
<td>381</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>17</td>
<td>5</td>
<td>20</td>
<td>5</td>
<td>58</td>
<td>8</td>
<td>17417</td>
<td>36140</td>
<td>16%</td>
<td>33%</td>
<td>110377</td>
<td>2470</td>
</tr>
<tr>
<td>382</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>67</td>
<td>5</td>
<td>68</td>
<td>8</td>
<td>52</td>
<td>8</td>
<td>21808</td>
<td>13154</td>
<td>41%</td>
<td>25%</td>
<td>52637</td>
<td>1289</td>
</tr>
<tr>
<td>383</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>84</td>
<td>0%</td>
<td>4%</td>
<td>2054</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>384</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>8</td>
<td>31%</td>
<td>27%</td>
<td>50786</td>
<td>2402</td>
</tr>
<tr>
<td>385</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>31</td>
<td>8</td>
<td>18069</td>
<td>0%</td>
<td>28%</td>
<td>64507</td>
<td>2365</td>
<td></td>
</tr>
<tr>
<td>386</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>79</td>
<td>0%</td>
<td>4%</td>
<td>1965</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>387</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>16</td>
<td>4</td>
<td>19</td>
<td>5</td>
<td>63</td>
<td>8</td>
<td>14146</td>
<td>36176</td>
<td>13%</td>
<td>33%</td>
<td>110131</td>
<td>2368</td>
</tr>
<tr>
<td>388</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>0%</td>
<td>5%</td>
<td>1837</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>389</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>16</td>
<td>5</td>
<td>14</td>
<td>5</td>
<td>57</td>
<td>10</td>
<td>4065</td>
<td>9457</td>
<td>11%</td>
<td>25%</td>
<td>37141</td>
<td>389</td>
</tr>
<tr>
<td>390</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>111</td>
<td>12</td>
<td>0</td>
<td>31238</td>
<td>0%</td>
<td>2%</td>
<td>1269081</td>
</tr>
<tr>
<td>391</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>62</td>
<td>7</td>
<td>58</td>
<td>9</td>
<td>123</td>
<td>10</td>
<td>15636</td>
<td>17798</td>
<td>39%</td>
<td>45%</td>
<td>39801</td>
<td>1798</td>
</tr>
<tr>
<td>392</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>43</td>
<td>7</td>
<td>35</td>
<td>7</td>
<td>37</td>
<td>8</td>
<td>12264</td>
<td>8118</td>
<td>37%</td>
<td>25%</td>
<td>33134</td>
<td>62</td>
</tr>
<tr>
<td>393</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>67</td>
<td>0%</td>
<td>3%</td>
<td>2202</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>394</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6240</td>
<td>0%</td>
<td>25%</td>
<td>24754</td>
<td>261</td>
<td>2871</td>
<td></td>
</tr>
<tr>
<td>395</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>28</td>
<td>5</td>
<td>24</td>
<td>7</td>
<td>28</td>
<td>6</td>
<td>7907</td>
<td>5451</td>
<td>24%</td>
<td>16%</td>
<td>33117</td>
<td>63</td>
</tr>
<tr>
<td>396</td>
<td>Aggressive</td>
<td>3</td>
<td>3</td>
<td>42</td>
<td>7</td>
<td>44</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>10654</td>
<td>30983</td>
<td>1%</td>
<td>2%</td>
<td>1290814</td>
<td>697</td>
</tr>
<tr>
<td>397</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>29</td>
<td>6</td>
<td>32</td>
<td>7</td>
<td>32</td>
<td>6</td>
<td>12307</td>
<td>18299</td>
<td>16%</td>
<td>23%</td>
<td>78976</td>
<td>4020</td>
</tr>
<tr>
<td>398</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>7</td>
<td>0</td>
<td>4954</td>
<td>0%</td>
<td>15%</td>
<td>32433</td>
<td>853</td>
<td></td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>----------</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>---------</td>
<td>----</td>
</tr>
<tr>
<td>399</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>111</td>
<td>12</td>
<td>0</td>
<td>32373</td>
<td>0%</td>
<td>2%</td>
<td>1330966</td>
<td>791</td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>Aggressive</td>
<td>3</td>
<td>41</td>
<td>7</td>
<td>46</td>
<td>101</td>
<td>12</td>
<td>10886</td>
<td>29789</td>
<td>1%</td>
<td>2%</td>
<td>1368807</td>
<td>784</td>
<td></td>
<td></td>
</tr>
<tr>
<td>401</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>6</td>
<td>6</td>
<td>4046</td>
<td>0%</td>
<td>17%</td>
<td>24177</td>
<td>243</td>
<td></td>
<td></td>
</tr>
<tr>
<td>402</td>
<td>Aggressive</td>
<td>6</td>
<td>41</td>
<td>7</td>
<td>41</td>
<td>117</td>
<td>13</td>
<td>9866</td>
<td>15540</td>
<td>7%</td>
<td>11%</td>
<td>139521</td>
<td>1850</td>
<td></td>
<td></td>
</tr>
<tr>
<td>403</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>392</td>
<td>0%</td>
<td>19%</td>
<td>2104</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13196</td>
<td>18%</td>
<td>24%</td>
<td>74274</td>
<td>2367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>405</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>16</td>
<td>4</td>
<td>19</td>
<td>60</td>
<td>8</td>
<td>19288</td>
<td>35979</td>
<td>20%</td>
<td>37%</td>
<td>97429</td>
<td>5004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>406</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>17680</td>
<td>0%</td>
<td>29%</td>
<td>60017</td>
<td>2367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>407</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>271</td>
<td>0%</td>
<td>15%</td>
<td>1852</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>408</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>65</td>
<td>0%</td>
<td>3%</td>
<td>2257</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>409</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>18</td>
<td>5</td>
<td>22</td>
<td>6</td>
<td>65</td>
<td>8</td>
<td>13949</td>
<td>15%</td>
<td>38%</td>
<td>92160</td>
<td>2364</td>
<td></td>
<td></td>
</tr>
<tr>
<td>410</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>34</td>
<td>8</td>
<td>13443</td>
<td>18%</td>
<td>25%</td>
<td>72896</td>
<td>2366</td>
<td></td>
<td></td>
</tr>
<tr>
<td>411</td>
<td>TCP Connect</td>
<td>9</td>
<td>78</td>
<td>7</td>
<td>47</td>
<td>8</td>
<td>45</td>
<td>10</td>
<td>16941</td>
<td>43%</td>
<td>16%</td>
<td>39820</td>
<td>71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>412</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>0%</td>
<td>30%</td>
<td>59644</td>
<td>2364</td>
<td></td>
<td></td>
</tr>
<tr>
<td>413</td>
<td>Aggressive</td>
<td>6</td>
<td>82</td>
<td>7</td>
<td>78</td>
<td>9</td>
<td>126</td>
<td>10</td>
<td>22631</td>
<td>4%</td>
<td>3%</td>
<td>617511</td>
<td>1827</td>
<td></td>
<td></td>
</tr>
<tr>
<td>414</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>33</td>
<td>5</td>
<td>41</td>
<td>8</td>
<td>66</td>
<td>8</td>
<td>32040</td>
<td>30%</td>
<td>34%</td>
<td>106518</td>
<td>2361</td>
<td></td>
<td></td>
</tr>
<tr>
<td>415</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3560</td>
<td>0%</td>
<td>15%</td>
<td>23808</td>
<td>249</td>
<td></td>
<td></td>
</tr>
<tr>
<td>416</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>0%</td>
<td>5%</td>
<td>1716</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>417</td>
<td>TCP Connect</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>38</td>
<td>8</td>
<td>18274</td>
<td>39%</td>
<td>26%</td>
<td>46571</td>
<td>1926</td>
<td></td>
<td></td>
</tr>
<tr>
<td>418</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>5%</td>
<td>1725</td>
<td>1341</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
<td>ET</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------</td>
<td>------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>419</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>33</td>
<td>5</td>
<td>22</td>
<td>5</td>
<td>30</td>
<td>5</td>
<td>6497</td>
<td>4413</td>
<td>21%</td>
<td>14%</td>
<td>30685</td>
<td>438</td>
</tr>
<tr>
<td>420</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>112</td>
<td>0%</td>
<td>8%</td>
<td>1442</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>421</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>12</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>67</td>
<td>8</td>
<td>13982</td>
<td>35230</td>
<td>15%</td>
<td>30%</td>
<td>91372</td>
<td>2367</td>
</tr>
<tr>
<td>422</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>47</td>
<td>8</td>
<td>2333</td>
<td>9318</td>
<td>7%</td>
<td>27%</td>
<td>34092</td>
<td>291</td>
</tr>
<tr>
<td>423</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>6</td>
<td>17868</td>
<td>0%</td>
<td>30%</td>
<td>60313</td>
<td>2363</td>
<td></td>
</tr>
<tr>
<td>424</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93</td>
<td>0%</td>
<td>5%</td>
<td>1920</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>425</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>127</td>
<td>12</td>
<td>40262</td>
<td>15467</td>
<td>95%</td>
<td>37%</td>
<td>42275</td>
<td>1815</td>
</tr>
<tr>
<td>426</td>
<td>Aggressive</td>
<td>9</td>
<td>8</td>
<td>132</td>
<td>9</td>
<td>115</td>
<td>10</td>
<td>46</td>
<td>7</td>
<td>31240</td>
<td>12512</td>
<td>96%</td>
<td>39%</td>
<td>32389</td>
<td>1099</td>
</tr>
<tr>
<td>427</td>
<td>Aggressive</td>
<td>6</td>
<td>5</td>
<td>65</td>
<td>6</td>
<td>58</td>
<td>8</td>
<td>51</td>
<td>7</td>
<td>17467</td>
<td>11541</td>
<td>80%</td>
<td>53%</td>
<td>21869</td>
<td>956</td>
</tr>
<tr>
<td>428</td>
<td>TCP Connect</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>17</td>
<td>5</td>
<td>6302</td>
<td>0%</td>
<td>20%</td>
<td>30743</td>
<td>709</td>
<td></td>
</tr>
<tr>
<td>429</td>
<td>NIDS Avoidance</td>
<td>6</td>
<td>6</td>
<td>21</td>
<td>5</td>
<td>23</td>
<td>5</td>
<td>63</td>
<td>8</td>
<td>14513</td>
<td>35542</td>
<td>13%</td>
<td>32%</td>
<td>112434</td>
<td>4904</td>
</tr>
<tr>
<td>430</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>60</td>
<td>5</td>
<td>40</td>
<td>7</td>
<td>35</td>
<td>8</td>
<td>18381</td>
<td>6910</td>
<td>47%</td>
<td>18%</td>
<td>38861</td>
<td>22</td>
</tr>
<tr>
<td>431</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>28</td>
<td>6</td>
<td>2128</td>
<td>17759</td>
<td>2%</td>
<td>17%</td>
<td>102732</td>
<td>12076</td>
</tr>
<tr>
<td>432</td>
<td>TCP Connect</td>
<td>3</td>
<td>3</td>
<td>10</td>
<td>5</td>
<td>11</td>
<td>5</td>
<td>20</td>
<td>6</td>
<td>4206</td>
<td>2022</td>
<td>14%</td>
<td>7%</td>
<td>31038</td>
<td>391</td>
</tr>
<tr>
<td>433</td>
<td>TCP Connect</td>
<td>6</td>
<td>6</td>
<td>28</td>
<td>5</td>
<td>30</td>
<td>7</td>
<td>45</td>
<td>8</td>
<td>7174</td>
<td>11116</td>
<td>22%</td>
<td>34%</td>
<td>32913</td>
<td>204</td>
</tr>
<tr>
<td>434</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>5%</td>
<td>1748</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>435</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>9</td>
<td>131</td>
<td>4</td>
<td>137</td>
<td>7</td>
<td>62</td>
<td>8</td>
<td>32068</td>
<td>36130</td>
<td>30%</td>
<td>34%</td>
<td>106204</td>
<td>2361</td>
</tr>
<tr>
<td>436</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>13165</td>
<td>0%</td>
<td>18%</td>
<td>0%</td>
<td>72645</td>
<td>2368</td>
</tr>
<tr>
<td>437</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>40</td>
<td>6</td>
<td>4264</td>
<td>11456</td>
<td>9%</td>
<td>25%</td>
<td>45205</td>
<td>447</td>
</tr>
<tr>
<td>438</td>
<td>TCP Connect</td>
<td>9</td>
<td>9</td>
<td>64</td>
<td>5</td>
<td>50</td>
<td>7</td>
<td>40</td>
<td>8</td>
<td>19618</td>
<td>9842</td>
<td>43%</td>
<td>22%</td>
<td>45758</td>
<td>1173</td>
</tr>
</tbody>
</table>
Table 13 continued from previous page

<table>
<thead>
<tr>
<th>Trial</th>
<th>ST</th>
<th>HP</th>
<th>HHI</th>
<th>SAI</th>
<th>STI</th>
<th>SAM</th>
<th>STM</th>
<th>SuA</th>
<th>SuT</th>
<th>HHP</th>
<th>% HHP</th>
<th>% SuP</th>
<th>AP</th>
<th>ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>439</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>35</td>
<td>7</td>
<td>63</td>
<td>8</td>
<td>32070</td>
<td>35431</td>
<td>30%</td>
<td>33%</td>
<td>107572</td>
<td>2364</td>
</tr>
<tr>
<td>440</td>
<td>Control Group 3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>74</td>
<td>0%</td>
<td>4%</td>
<td>1764</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>441</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>20</td>
<td>4</td>
<td>20</td>
<td>5</td>
<td>22</td>
<td>6</td>
<td>20031</td>
<td>35028</td>
<td>19%</td>
<td>33%</td>
<td>107602</td>
<td>2369</td>
</tr>
<tr>
<td>442</td>
<td>Control Group 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0%</td>
<td>4%</td>
<td>2096</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>Control Group 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>0%</td>
<td>4%</td>
<td>1704</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>444</td>
<td>TCP Connect 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>66</td>
<td>0%</td>
<td>4%</td>
<td>1704</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>445</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>261</td>
<td>0%</td>
<td>14%</td>
<td>1837</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>446</td>
<td>Control Group 3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>261</td>
<td>0%</td>
<td>14%</td>
<td>1837</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>447</td>
<td>NIDS Avoidance 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16999</td>
<td>0%</td>
<td>27%</td>
<td>63788</td>
<td>2369</td>
<td></td>
</tr>
<tr>
<td>448</td>
<td>TCP Connect 3</td>
<td>3</td>
<td>9</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>16</td>
<td>6</td>
<td>2360</td>
<td>1656</td>
<td>8%</td>
<td>5%</td>
<td>31020</td>
<td>267</td>
</tr>
<tr>
<td>449</td>
<td>Control Group 6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>407</td>
<td>0%</td>
<td>18%</td>
<td>2313</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>450</td>
<td>Control Group 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>77</td>
<td>0%</td>
<td>5%</td>
<td>1627</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>451</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29840</td>
<td>0%</td>
<td>5%</td>
<td>616091</td>
<td>608</td>
<td></td>
</tr>
<tr>
<td>452</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>62</td>
<td>8</td>
<td>32003</td>
<td>34564</td>
<td>30%</td>
<td>32%</td>
</tr>
<tr>
<td>453</td>
<td>NIDS Avoidance 9</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>33</td>
<td>7</td>
<td>62</td>
<td>8</td>
<td>32383</td>
<td>35168</td>
<td>30%</td>
<td>33%</td>
<td>107745</td>
<td>2366</td>
</tr>
<tr>
<td>454</td>
<td>Aggressive 6</td>
<td>6</td>
<td>5</td>
<td>51</td>
<td>9</td>
<td>49</td>
<td>9</td>
<td>239</td>
<td>10</td>
<td>9973</td>
<td>13414</td>
<td>35%</td>
<td>47%</td>
<td>28845</td>
</tr>
<tr>
<td>455</td>
<td>TCP Connect 9</td>
<td>9</td>
<td>66</td>
<td>7</td>
<td>61</td>
<td>9</td>
<td>45</td>
<td>8</td>
<td>19490</td>
<td>9039</td>
<td>39%</td>
<td>18%</td>
<td>49478</td>
<td>2956</td>
</tr>
<tr>
<td>456</td>
<td>Aggressive 9</td>
<td>9</td>
<td>8</td>
<td>45</td>
<td>7</td>
<td>46</td>
<td>9</td>
<td>244</td>
<td>12</td>
<td>8123</td>
<td>12856</td>
<td>19%</td>
<td>30%</td>
<td>42822</td>
</tr>
<tr>
<td>457</td>
<td>Aggressive 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>115</td>
<td>12</td>
<td>0</td>
<td>13926</td>
<td>0%</td>
<td>4%</td>
<td>328201</td>
</tr>
<tr>
<td>458</td>
<td>TCP Connect 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>8</td>
<td>5795</td>
<td>0%</td>
<td>27%</td>
<td>21736</td>
<td>51</td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
</tr>
<tr>
<td>-------</td>
<td>---------------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>459</td>
<td>TCP Connect</td>
<td>6</td>
<td>30</td>
<td>5</td>
<td>28</td>
<td>7</td>
<td>33</td>
<td>8</td>
<td>8437</td>
<td>7180</td>
<td>26%</td>
<td>22%</td>
<td>32494</td>
<td>21</td>
</tr>
<tr>
<td>460</td>
<td>Aggressive</td>
<td>3</td>
<td>36</td>
<td>6</td>
<td>44</td>
<td>9</td>
<td>105</td>
<td>12</td>
<td>10791</td>
<td>8698</td>
<td>4%</td>
<td>3%</td>
<td>249548</td>
<td>576</td>
</tr>
<tr>
<td>461</td>
<td>TCP Connect</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>2757</td>
<td>2426</td>
<td>9%</td>
<td>8%</td>
<td>29896</td>
<td>206</td>
</tr>
<tr>
<td>462</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>16</td>
<td>5</td>
<td>30</td>
<td>5</td>
<td>13365</td>
<td>17607</td>
<td>17%</td>
<td>23%</td>
<td>77673</td>
<td>2369</td>
</tr>
<tr>
<td>463</td>
<td>TCP Connect</td>
<td>6</td>
<td>13</td>
<td>5</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>4178</td>
<td>0</td>
<td>13%</td>
<td>0%</td>
<td>32501</td>
<td>25</td>
</tr>
<tr>
<td>464</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>133</td>
<td>12</td>
<td>8757</td>
<td>0%</td>
<td>3%</td>
<td>250691</td>
<td>579</td>
<td></td>
</tr>
<tr>
<td>465</td>
<td>TCP Connect</td>
<td>3</td>
<td>69</td>
<td>7</td>
<td>77</td>
<td>9</td>
<td>20</td>
<td>8</td>
<td>20917</td>
<td>1962</td>
<td>66%</td>
<td>6%</td>
<td>31710</td>
<td>349</td>
</tr>
<tr>
<td>466</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>86</td>
<td>0%</td>
<td>5%</td>
<td>1777</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>467</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>73</td>
<td>0%</td>
<td>4%</td>
<td>1805</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>468</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>24</td>
<td>4</td>
<td>33</td>
<td>7</td>
<td>65</td>
<td>8</td>
<td>30137</td>
<td>36170</td>
<td>27%</td>
<td>33%</td>
<td>110085</td>
<td>2361</td>
</tr>
<tr>
<td>469</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>32</td>
<td>5</td>
<td>37</td>
<td>5</td>
<td>33</td>
<td>6</td>
<td>13309</td>
<td>18103</td>
<td>17%</td>
<td>24%</td>
<td>76477</td>
<td>2367</td>
</tr>
<tr>
<td>470</td>
<td>Control Group</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89</td>
<td>0%</td>
<td>4%</td>
<td>1985</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>471</td>
<td>Control Group</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>69</td>
<td>0%</td>
<td>5%</td>
<td>1435</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>472</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0%</td>
<td>7%</td>
<td>1291</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>473</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>18158</td>
<td>0%</td>
<td>29%</td>
<td>63496</td>
<td>2365</td>
<td></td>
</tr>
<tr>
<td>474</td>
<td>Control Group</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>75</td>
<td>0%</td>
<td>6%</td>
<td>1290</td>
<td>1341</td>
<td></td>
</tr>
<tr>
<td>475</td>
<td>Aggressive</td>
<td>3</td>
<td>50</td>
<td>7</td>
<td>45</td>
<td>9</td>
<td>193</td>
<td>12</td>
<td>12677</td>
<td>9238</td>
<td>4%</td>
<td>3%</td>
<td>322296</td>
<td>1622</td>
</tr>
<tr>
<td>476</td>
<td>NIDS Avoidance</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>62</td>
<td>8</td>
<td>32065</td>
<td>36238</td>
<td>30%</td>
<td>34%</td>
<td>106835</td>
<td>2361</td>
</tr>
<tr>
<td>477</td>
<td>NIDS Avoidance</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>6</td>
<td>18168</td>
<td>0%</td>
<td>28%</td>
<td>65130</td>
<td>2364</td>
<td></td>
</tr>
<tr>
<td>478</td>
<td>Aggressive</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>112</td>
<td>12</td>
<td>9159</td>
<td>0%</td>
<td>1%</td>
<td>639420</td>
<td>606</td>
<td></td>
</tr>
<tr>
<td>Trial</td>
<td>ST</td>
<td>HP</td>
<td>HHI</td>
<td>SAI</td>
<td>STI</td>
<td>SAM</td>
<td>STM</td>
<td>SuA</td>
<td>SuT</td>
<td>HHP</td>
<td>SuP</td>
<td>% HHP</td>
<td>% SuP</td>
<td>AP</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>479</td>
<td>Control Group</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>83</td>
<td></td>
<td>0%</td>
<td>7%</td>
<td>1192</td>
</tr>
<tr>
<td>480</td>
<td>NIDS Avoidance</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>15</td>
<td>5</td>
<td>29</td>
<td>5</td>
<td>13465</td>
<td>18132</td>
<td>18%</td>
<td>24%</td>
<td>75482</td>
</tr>
</tbody>
</table>
Appendix F. permutation_test.py

permutation_test.py

```python
import os
import numpy
import random

numPermutations = 900000
experimentFile = 'results.csv'

set1 = []
set2 = []
combined = []
mean1 = 0
mean2 = 0
difference = 0
numExceedDif = 0

# checks to see if a file with the run order exists and makes sure it's not empty
if os.path.exists(experimentFile) and os.path.getsize(experimentFile) > 0:
    f = open(experimentFile, 'r')
    for line in f:
        value = line.split(',
lambdaFunc = None

        # print value[0]
        set1.append(float(value[0]))
        # print value[1]
        set2.append(float(value[1].rstrip()))
```

233
f.close()

# print 'set1'
# print set1
# print 'set2'
# print set2

mean1 = numpy.average(set1)
print 'Mean1: ' + str(mean1)
mean2 = numpy.average(set2)
print 'Mean2: ' + str(mean2)
difference = abs(mean1 - mean2)
print "Difference: " + str(difference)
combined = set1 + set2
# print combined

for x in range(numPermutations):
    random.shuffle(combined)

    sample1Avg = numpy.average(combined[:len(combined)//2])
sample2Avg = numpy.average(combined[len(combined)//2:])
    if abs(sample1Avg - sample2Avg) > difference:
        numExceedDif +=1
result = numExceedDif * 1.0 / numPermutations
print result

else:
    print "Error, file not found!"
Bibliography


235


HoneyHive – A Network Intrusion Detection System Framework Utilizing Distributed Internet of Things Honeypot Sensors

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

Joseph A. Misher
Department of Homeland Security
Cyber Physical Division, Federal Protective Service
800 North Capitol Street NW, Washington D.C. 20001
COMM 202-658-8806
Email: Joseph.misher@hq.dhs.gov

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Exploding over the past decade, the number of Internet of Things (IoT) devices connected to the Internet jumped from 3.8 billion in 2015 to 17.8 billion in 2018. Because so many IoT devices remain unpatched, unmonitored, and left on, they have become a tantalizing target for attackers to gain network access or add another device to their botnet. HoneyHive is a framework that uses distributed IoT honeypots as Network Intrusion Detection Systems (NIDS) sensors that beacon back to a centralized Command and Control (C2) server. The tests in this experiment involve four types of scans and four levels of active honeypots against the HoneyHive framework and a traditional NIDS on the simulated test network. This research successfully created a framework of distributed network intrusion detection IoT honeypot sensors that capture traffic, create alerts, and beacon back to a central C2 server. The HoneyHive framework successfully detected intrusions that traditional NIDS cannot through the use of distributed IoT honeypot sensors and packet capture aggregation.