12-2019

Machine Learning in the State Design Pattern

Timothy Matthew von Friesen

Machine Learning in the State Design Pattern

by

Timothy Matthew von Friesen

A thesis submitted to the College of Engineering and Science
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Masters of Science
in
Computer Engineering

Melbourne, Florida
December, 2019

We the undersigned committee hereby approve the attached thesis, “Machine Learning in the State Design Pattern,” by Timothy Matthew von Friesen.

Carlos E. Otero, Ph.D.
Associate Professor
Computer Engineering
Major Advisor

Ersoy Subasi, Ph.D.
Assistant Professor
Systems Engineering

Samuel P. Kozaitis, Ph.D.
Professor
Computer Engineering

Philip Bernhard, Ph.D.
Associate Professor and Head
Department of Computer Engineering and Sciences

Abstract

Machine Learning in the State Design Pattern

Timothy Matthew von Friesen

Carlos E. Otero, Ph. D., Major Advisor

As the Internet of Things revolution continues to become more prevalent in humanity's daily routine, securing these devices is paramount. Society has seen a substantial increase in activity in the cyber-warfare battle space, resulting in an increasing amount of security breaches every year. The responsibility of securing our devices can no longer rely solely on cyber-security engineers keeping systems hardened through Security Technical Implementation Guides and vulnerability scans; it must shift towards the developer.

Previous research has been done in this area of securing our devices. However, these solutions rely heavily on cloud computing resources to perform computationally expensive algorithms. While these solutions do work, devices without the support of an extensive cloud backbone lack adequate protection through these methods.

The goal of this thesis is to provide a modern approach through the use of machine learning and its incorporation into the typical design pattern, State Based Machines. The coupling of these two concepts can allow for an increased security posture at a small cost of overall system performance.

Table of Contents

List of Figures

List of Tables

Acknowledgment

I am grateful to my father for his continued encouragement throughout the years. Your motivation kept my thirst for knowledge never fulfilled. Without you, I would never have dreamed of achieving what I have.

I want to express my appreciation and gratitude to my advisor and mentor Dr. Carlos E. Otero. His continued support and guidance throughout my academic career have had a profound impact on my life.

I want to thank the support of everyone in the WiCE Lab throughout the years of my work on this thesis. All of the suggestions on various problems I faced provided valuable insight.

Dedication

In dedication to my wife, Alicia; without your continued support, love, and adventurous spirit I would not be where I am today. I am forever grateful.

To Infinity and Beyond…

Chapter 1
Introduction

This section discusses the changing landscape of our society in regards to the increase of Internet of Things (IoT) devices and correlation with today's new battleground that often goes unnoticed, cyber-space.

IoT Revolution

Society has mostly accepted the Internet of Things revolution into their daily lives through the use of various smart sensing devices ranging from smart sensors to sophisticated health care systems. Current estimates establish that by 2025, there will be over 75 billion connected IoT devices worldwide, up from a mere 15 billion in 2015. Equating to approximately 9 IoT devices per living human-being worldwide [1, 2]. The history of society's indulgence in IoT devices suggests that our world will continue to become further

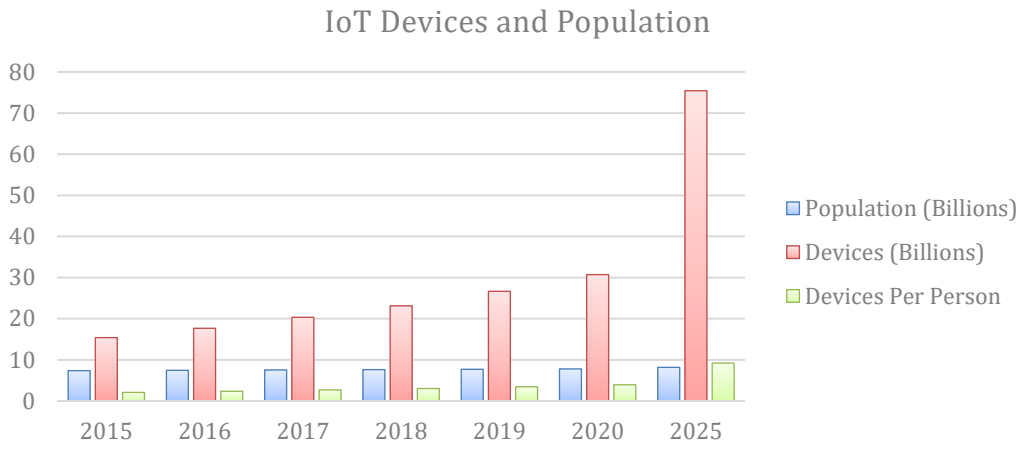

Figure 1: IoT Devices and Population

reliant on these smart devices, and their incorporation into society's core functionality will evolve.

As IoT technology advances and continues to close the gap between the digital and physical worlds through the use of various controllers, actuators, and sensors, this results in the unlimited potential for the use of these devices. These devices have become incorporated into many mission-critical applications, such as environmental monitoring, healthcare, traffic, emergency and infrastructure management, defense, business and intelligent material science [3]. Due to the open nature of Internet connectivity, the large scale, and the complexity of these architectures' new threats and challenges for the security community are emerging; hackers may exploit critical vulnerabilities in a wide range of IoT applications and devices [4]. IoT devices have and continue their presence in critical applications controlling applications, and this poses a substantial threat to society as exploited vulnerabilities can result in significant damage to infrastructure and even loss of life.

Cyber Warfare

With the growing number of vulnerable IoT devices becoming incorporated into personal, professional, health, defense, and infrastructure, the security of these devices must be addressed, and have adequate solutions implemented. As advanced cyber threats have become more sophisticated, there has been a shift in their goal from immediate damage infliction to long-term persistence access to devices and networks. Resulting in an average detection time exceeding several months, with some cases taking over a year to detect. Post detection resolution has remained low at an average of about a week to contain the threat [5]. These attacks have continued to increase over the past decade with no sign of subsiding and an average attack occurring every 39 seconds, thus resulting in millions of compromised records every year in the United States alone [6, 7].

In this thesis, it is intended to provide a solution to State-Based IoT devices by incorporating Machine Learning into the design pattern. This approach is intended to detect intruders who may be lurking or masking their presence within healthy system operational characteristics.

This thesis is structured as follows:

- In Chapter 2, background information will be provided on current intrusion detection methods to include applicable machine learning techniques in moderate detail. Additionally, we will review the traditional State Design Pattern with

modern-day examples. Lastly, we will look at valid metrics regarding system performance monitoring and its implications.

- In Chapter 3, we will present enhancements to the State Design Pattern; to include the test environment setup and execution.

- In Chapter 4, we will review detailed results on essential anomaly detection. Additionally, we will review these results against the purposed the modified State Design Pattern for anomaly detection.

- In Chapter 5, we will discuss an outline of future work, and the thesis concluded.

Chapter 2
Background

The purpose of this work is to address the growing concern of our everyday smart devices and their lack of security against malicious actors. In this section, we discuss the anomaly of Host-Based Intrusion Detection System (HIDS) methods. Including an overview of the Isolation Forest Anomaly Detection Algorithm. Lastly, we discuss a typical IoT system architecture with State Design Patterns and system operational characteristics.

Intrusion Detection

Intrusion Detection Systems (IDS) are designed to monitor systems for behavior that appears to be malicious in intent, accomplishing this with various techniques; anomaly detection is one method and implemented at different layers of system operation, for example, Network IDSs (NIDS). The primary function of an IDS is to reliably alert the system operator of any potential system compromise through the use of any implementation and technique mentioned above.

Anomaly Detection

Anomalies are significant variations from the typical operating characteristics of a system. Detection of anomalies requires the characterization of the statistical profile of healthy behavior, which allows for the detection of operational characteristics outside of the norm

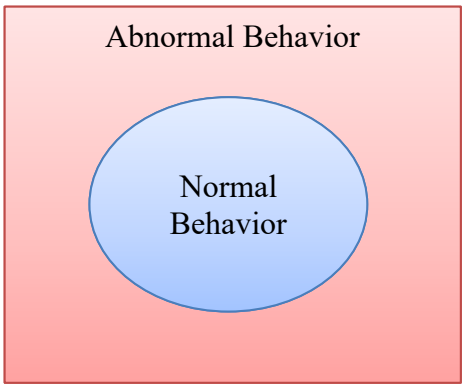

Figure 2: Intrusion Detection System

[8]. A clear case where system behavior is defined by the bounds of what is considered normal and abnormal behavior allows for classification of perceived behavior is depicted in Figure 2. There are three main principles in various anomaly detection algorithms: distance, density, and rank basis. Each providing benefits for detection for different system modeling requirements.

Analysis of system behavior, when applied to any anomaly detection algorithm, will result in one of three possible cases:

1. False Positive: Operation is classified as abnormal when the actual operation is normal.

2. False Negative: Operation is classified as normal when the actual operation is abnormal.

3. Correct Detection: Operation is classified correctly; abnormal is abnormal or normal is normal.

It is impossible for any anomaly detection system to achieve a 100% correct detection rate; however, careful design considerations require weighing the accuracy and desired error rate of the system. For example, A high False-Negative rate would result in attacks going unnoticed whereas a high False-Positive rate would result in increased system notifications.

Unsupervised Learning: Isolation Forest Algorithm

Isolation forests are an anomaly classification algorithm that has grown in popularity over the past several years for its inherent speed, accuracy, and minimal computational requirements in comparison to other algorithms. Isolation forests consist of a series of Isolation Trees, which are similar in structure to Binary Search Trees. This structure allows for isolating anomalies rather than profiling normal instances; in other words, anomalies are few and different which make them more susceptible to isolation than normal points [9].

Isolation Trees, which are the core unit of Isolation Forests, work best with smaller datasets, which leads to a favorable sub-sampling scenario. Large datasets reduce the Isolation Forest's ability to isolate anomalies from normal instances, thus reducing the accuracy of anomaly detection. This solution plays directly into the two primary problems facing anomaly detection:

1. Masking: Data points belonging to a normal and anomaly cluster are merged into a single cluster, causing outliers to become classified as normal.

2. Swamping: Data points classified as outliers cause a misclassification of normal data points (Normal classified as abnormal).

The very nature of Isolation Forests is the ability to build partial models by sub-sampling with alleviates the effects of masking and swamping. This improvement is due to sub-sampling resulting in better isolation of anomalies within the individual Isolation Trees and each Isolation Tree possessing the ability to be specialized as it may or may not contain an anomaly [9].

Isolation Trees are built by recursively dividing each sample-set, randomly selecting an attribute of the sample and the split value. This process is continued until the tree reaches its height limit, or no more data can be to the tree. As mentioned earlier Isolation Trees are a proper binary tree in that each node has no children or precisely two children. This requirement creates a tree with a total number of external nodes n and internal nodes of $n - 1$ resulting in a total node count of $2n - 1$. Anomalies are determined based on their path length or number of edges between the node and the root node.

One of the most important benefits of the Isolation Forest Algorithm is that the training data does not have stringent requirements as other algorithms do. Since Isolation Trees function with a sub-set of data, large data sets are not required. Additionally, unsupervised training (no classification data) does not significantly impact the accuracy of the model.

IoT Devices

Internet of Things is a reference to physical objects featuring Internet Protocol (IP) connectivity, more specifically devices that are not general-purpose computing devices such as a personal computer or any other computer. Devices referred to as IoT devices in this paper are internet-connected devices such as; thermostats, sensors, cars, and home automation.

The very nature of these devices is to fulfill a single purpose or function; this classifies IoT devices as types of embedded systems. Embedded systems are dedicated computer systems designed for one or two specific functions, often embedded as part of a complete device system including hardware such as electrical and mechanical components [10]. Embedded systems are often designed to optimize power, size, cost, and performance thus creating a highly specialized system that often has little room for extra unintended processing.

Most embedded devices are not processing batch jobs but rather event-driven processing. This processing method means that they continuously wait for the occurrence of some external or internal event, which has led to the embedded devices following a State Based Machine design pattern [11].

State Based Machine Design Pattern

The State Based design pattern allows an object to alter its behavior when its internal state changes; this causes the object to alter is operational characteristics with each state. Use of this design pattern is typical in several applications:

- System behavior depends on its current running state, which can dynamically change during operation.

- System internal operation depends heavily on the object's state often in large multipart conditional statements.

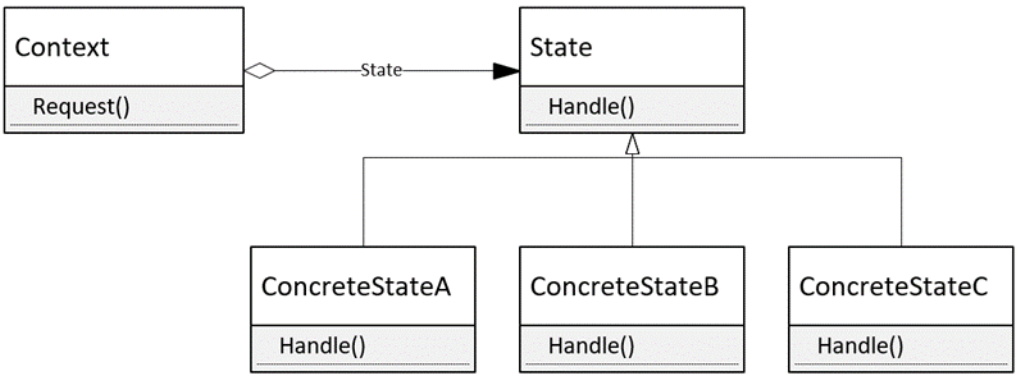

Figure 3: State Design Pattern

State Based design patterns are consist of a single context class and several concrete state classes that all inherit from the same base state interface. As the design pattern outlines, the only requirement for each state is to implement their *handle()* method. The required *handle()* method contains the state's operational code and ultimately controls the behavior of the application [12].

- Context: Defines the interface of interest to clients and maintains an instance of a ConcreteState subclass that defines the current state.

- State: Defines an interface for encapsulating the behavior associated with a particulate state of context.

- ConcreteState: Implements behavior associated with the state of context.

System Metrics

Typical system operational characteristics come in the form of various performance metrics of the system. These can range anywhere from physical hardware usage such as random-access memory (RAM), central processing unit (CPU), graphics processing unit (GPU), and network input/output to software statistics such as system calls [13]. The operating system natively captures hardware measurements and displayed in default monitoring software such as Task Manager for Windows or Top for Linux. Monitoring system calls are made possible with applications such as *strace,* which could potentially add excessive unwanted overhead on an already resource-limited device.

Chapter 3
Machine Learning and the State Machine

In this chapter, concepts from Chapter 2 are brought together in a unique way to solve the relevant problem of securing IoT devices through the use of HIDSs. This critical issue is accomplished by modifying the design principles at the very core of a typical embedded system, or IoT device. We also conduct a review of the modifications and associated implementation.

Overview

State Based embedded systems are designed to perform specialized functions often receiving input in some form either internal or external to the device to drive its current functional behavior. For example, a typical WiFi-Enabled camera, found in a large portion of households throughout the world, would change its device operation based upon images captured from its camera input device, i.e., alerting the user if motion is detected. Leveraging the fact that HIDSs monitor systems as a whole for abnormal behavior and embedded systems are designed to perform a specialized function, it is feasible to implement a lightweight HIDS specialized for a device in which its operation is reliant on a State Machine Design Pattern. However, characterizing the runtime performance of the entire system as a whole could result in an inability to isolate anomalies from normal runtime performance accurately. Alternately, characterizing the system performance based on the intended function or state, the system is currently can provide an accurate depiction

of normal system operation for each given state, thus providing increased anomaly isolation over the former.

Consider Figure 4, generated using a random number generator, measuring some feature that accurately characterizes the system for any given moment in time. Overall the system looks unpredictable. However, a model could be fit to this data and detect anomalies outside of the entire system's modeled characteristic.

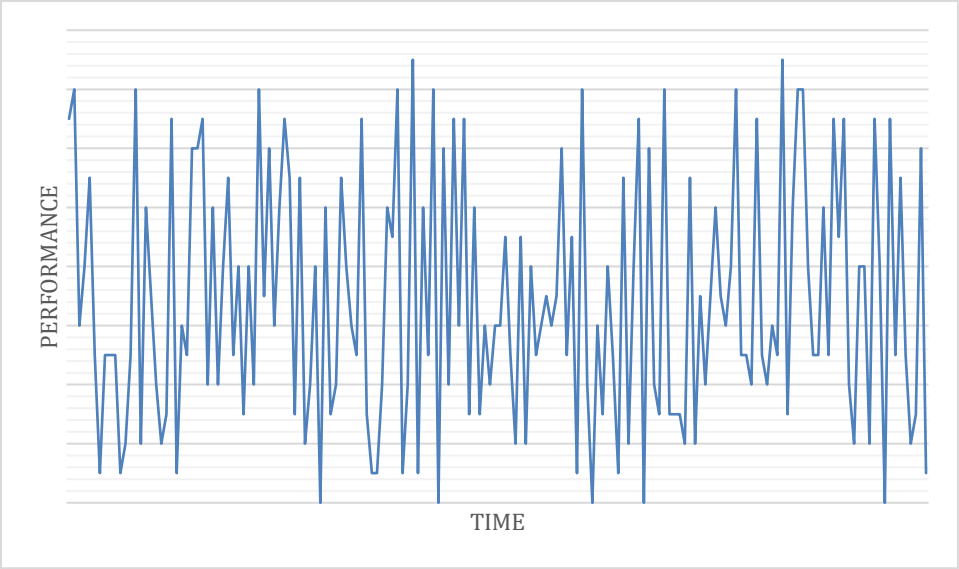

Figure 4: Overall Performance vs. Time

Similarly, as above, the random data present is in Figure 5; however, the distribution of the data is into a series of system functional states. This separation breaks the overall system performance into four separate states; in this case, allowing for increased granularity in characterizing system performance on what functionality the state-based system is intended to perform in each given state.

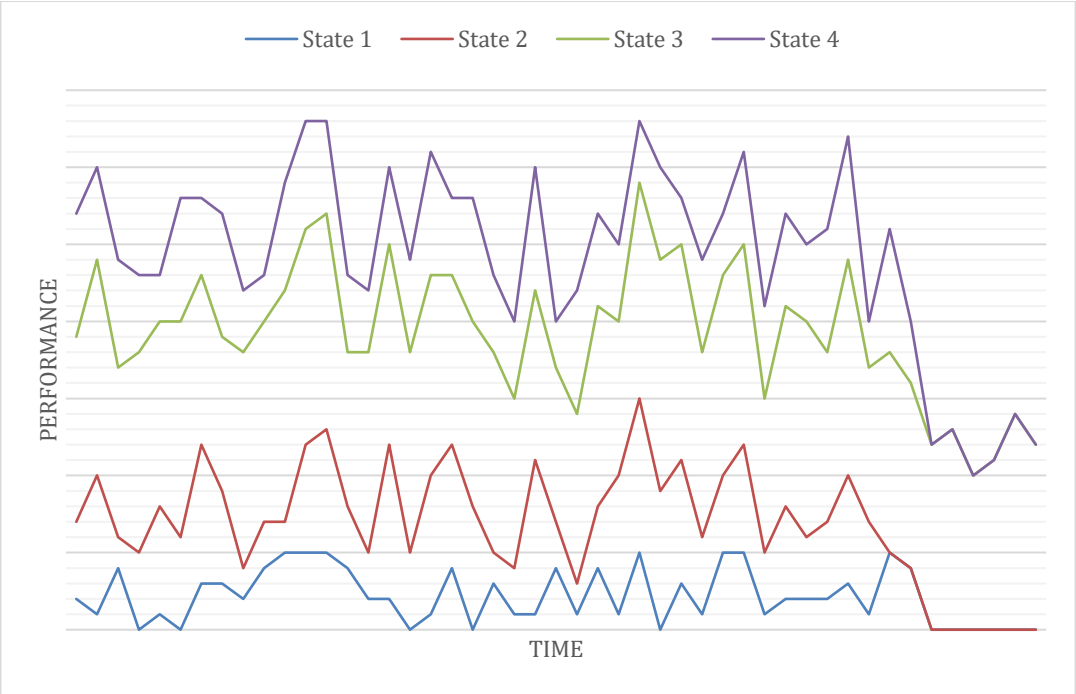

Figure 5: State Performance vs. Time

Modified State Based Machine Design Pattern

Starting with the State Machine Design Pattern discussed in Chapter 2, we can assume that our system can and will only operate in any defined ConcreteState. Expanding upon the basic concepts of the State Machine Design Pattern and including an additional required functionality called *model()* and *detect()*, it is possible to achieve the result depicted in Figure 5. That is isolating system performance characterization based on designed functionality for a given input and output of the state-driven system. It is possible to achieve this enhancement though the use of the *model()* function as it is intended to provide the applicable anomaly-based algorithm model to the associated HIDS for dynamic loading during runtime.

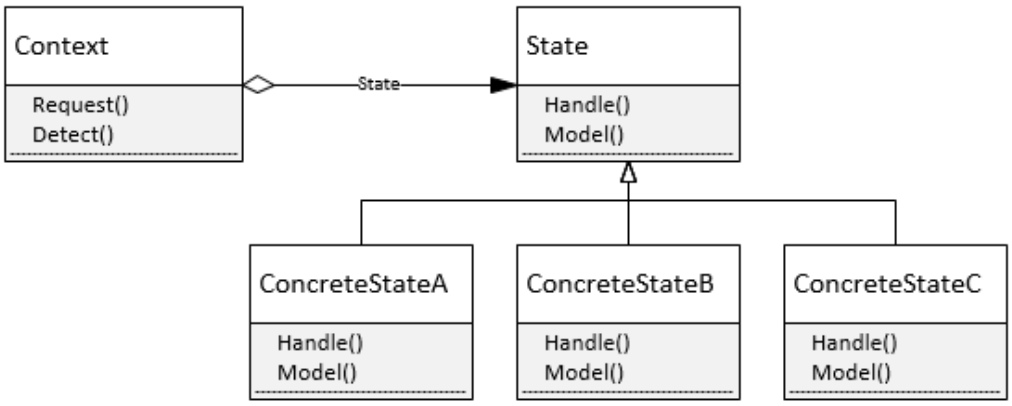

Figure 6: Modified State Based Machine Design Pattern

The machine learning model generation and interpretation were carefully selected to allow for a balance between efficiency and accuracy with modeling unsupervised data, considering the system data used to train the models might not contain any labels.

Embedded System Simulator

The embedded system simulator was modeled after a typical IoT device, a smart WiFi-Enabled camera, incorporating the modified State Design Pattern. Each state provided a means to interact with the Host Based Intrusion Detection System through the use of the *model()* and *detect()* function implementations within each state. Additionally, functionality was incorporated to perform a Denial of Service attack on the simulator to test the anomaly detection software within the HIDS.

The State Base Machine Design pattern incorporated in the embedded system simulation consisted of the following states:

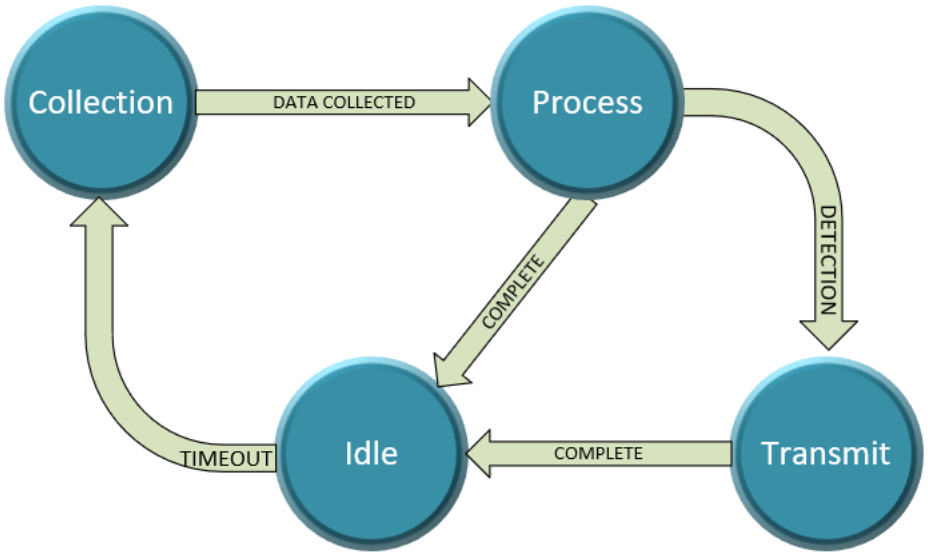

Figure 7: Simulate State Diagram

State functions and transitions in detail, as Figure 7 depicts:

- Idle
    - Device performs no immediate actions or processing
    - Transitions:
        - Processing of input data completed, with no motion detected.
        - Transmission of motion detected alert completed.
- Collection
    - Device collects data from input devices, in this case from the attached camera
    - Transitions:
        - Internal device interrupt triggered to collect data
- Process
    - Device batch processes collected data, performing basic motion detection algorithms
    - Transitions:
        - Device completion of data collection from input devices
- Transmit
    - Device establishes a network connection with the remote server providing notification of motion and collected data.
    - Transition:
        - Motion detected during the processing of collected data.

Host-Based Intrusion Detection System

The HIDS for this study was running on the same computer running the embedded system simulator. The primary function of the HIDS was to dynamically load the Isolation Forest model generated by the simulator and pass the runtime data through the loaded model for the detection of anomalies. This model transfer was accomplished through an inter-process communication method with the use of shared memory. The simulator upon changing states would load the applicable model into a shared memory location that the HIDS would monitor for changes. Once a state change was detected, the applicable state model would be loaded and used for anomaly detection. As the detection system and simulator were both applications on a general-purpose computer, system performance characteristics were isolated only to collect and analyze characteristics associated with the running simulation.

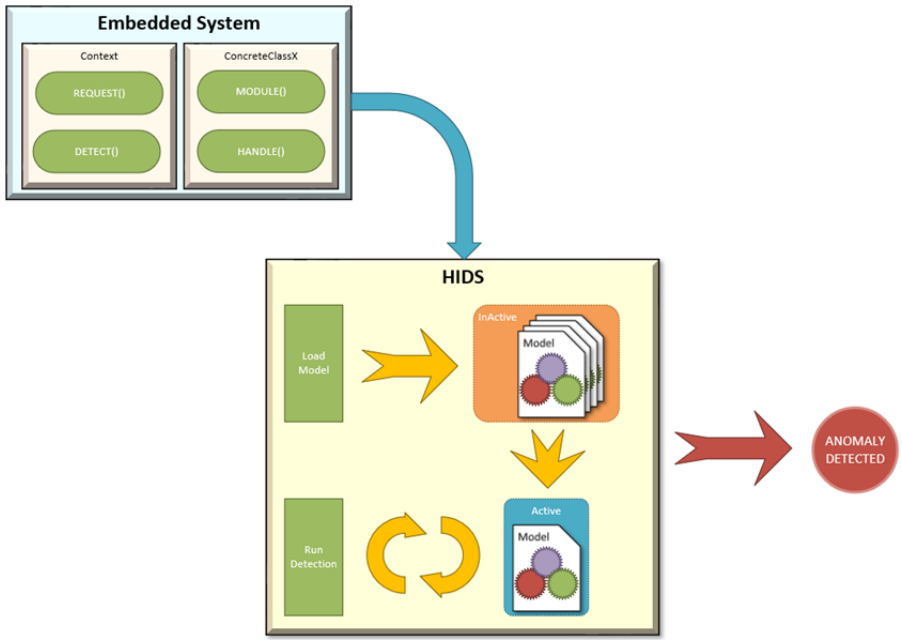

Figure 8: Simulation Setup

Chapter 4
Results and Analysis

In this chapter, the results obtained from experimentation with an embedded system utilizing the technique discussed in Chapter 3, a modified state design pattern infused with machine learning. We review the results from this new approach as compared to a control system running the classical state machine design pattern.

Training Data

The chosen performance metrics utilized to characterize the system operation at any given moment in time were CPU and RAM utilization in terms of overall usage. These chosen metrics are easy to acquire as the native capture of these characteristics is within the operating system. For the ease of reporting and analysis, the data used by the system to train each state's model is captured and presented in Figures 9 and 10. These two figures represent system metrics with respect to time throughout regular system operation with the system cycling the system through all four states; capture, process, transmit, and idle. Initial observations of this data suggest that system operation differs significantly at periods, which correlates directly with the current state the system is in concerning the state machine design pattern. However, it is difficult to characterize the performance of the system given the state.

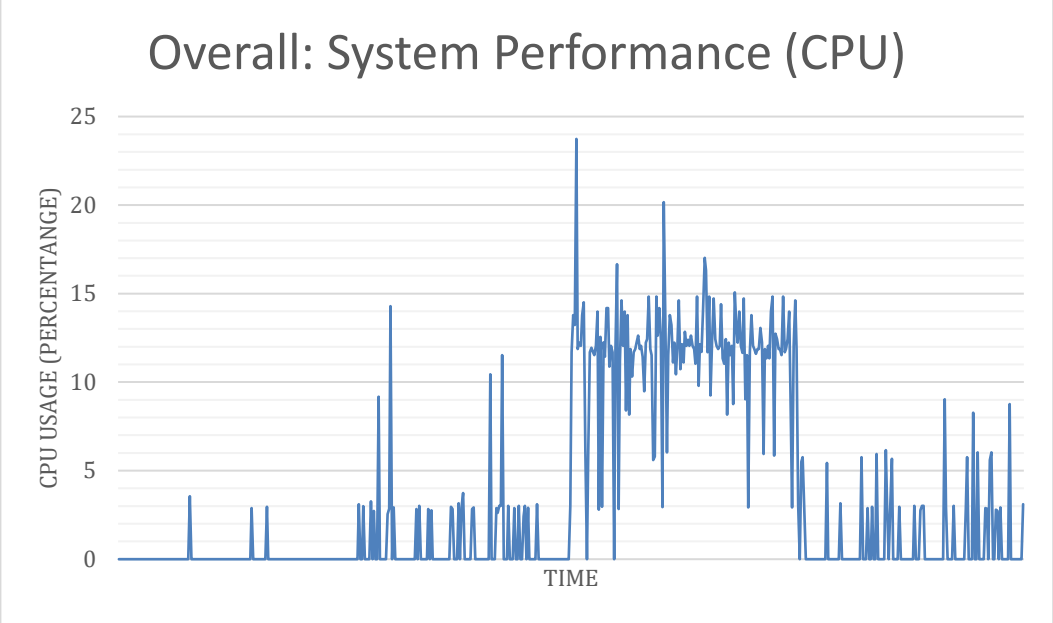

Figure 9: Training System Performance (CPU)

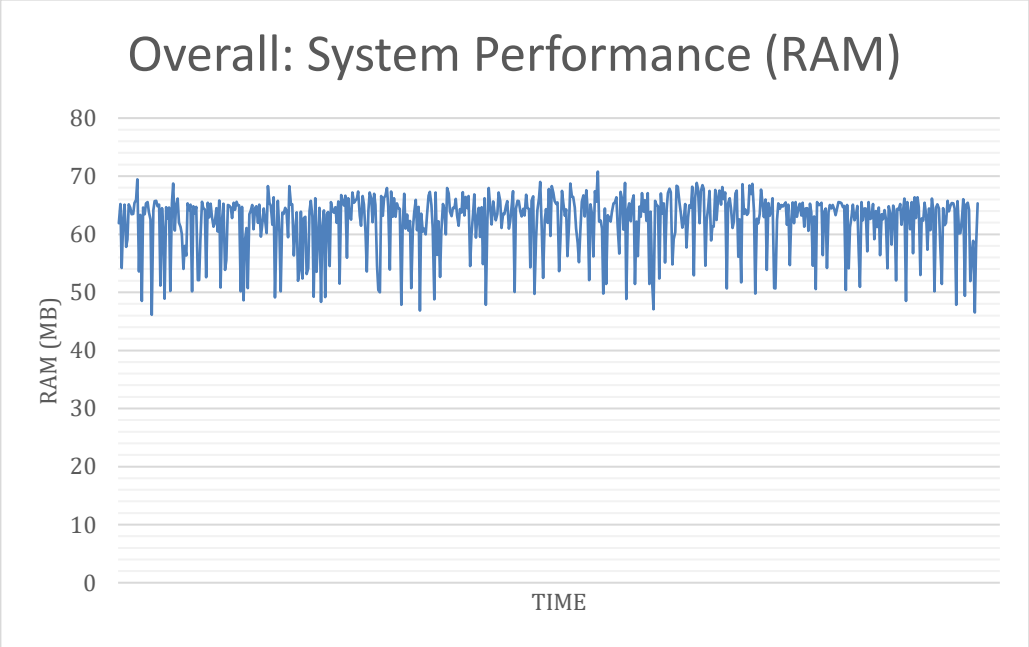

Figure 10: Training System Performance (RAM)

By attaching the state context to the system characteristics for a given moment in time, determining the normal operational conditions become a trivial task. Isolation forests perform best under smaller datasets, and the training sets consist of a subset of 150 samples per state. State operating characteristics depicted in Figure 11 lead to a clear depiction of clusters of typical operating performance.

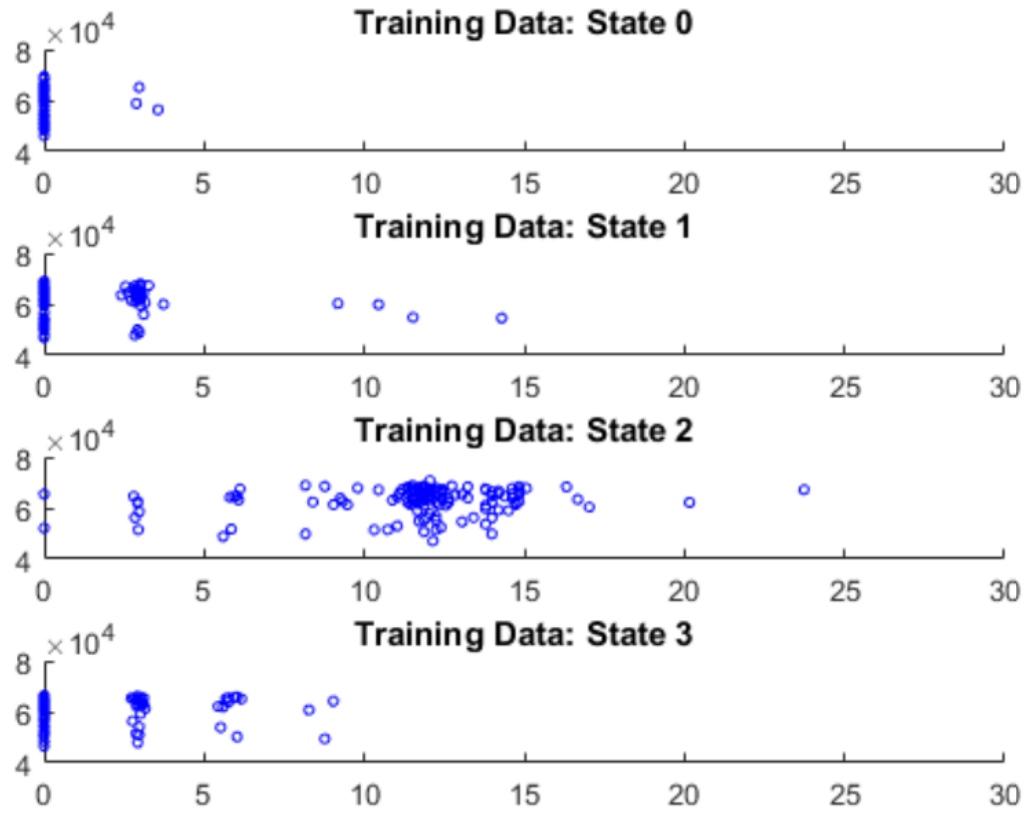

Figure 11: Training Data with States

Testing Data

The embedded system simulator, discussed in Chapter 3, was injected with a denial of service type attack. The goal was to provide clear indications of excess hardware usage well beyond the typical operational characteristics to highlight the downfalls of modeling systems as a whole instead of by the state of operation. The testing data was gathered through the HIDS for post runtime analysis and shown in Figure 12. Initial observations identify prominent clusters of operation that differ from the normal state of operation, as presented in Figure 11.

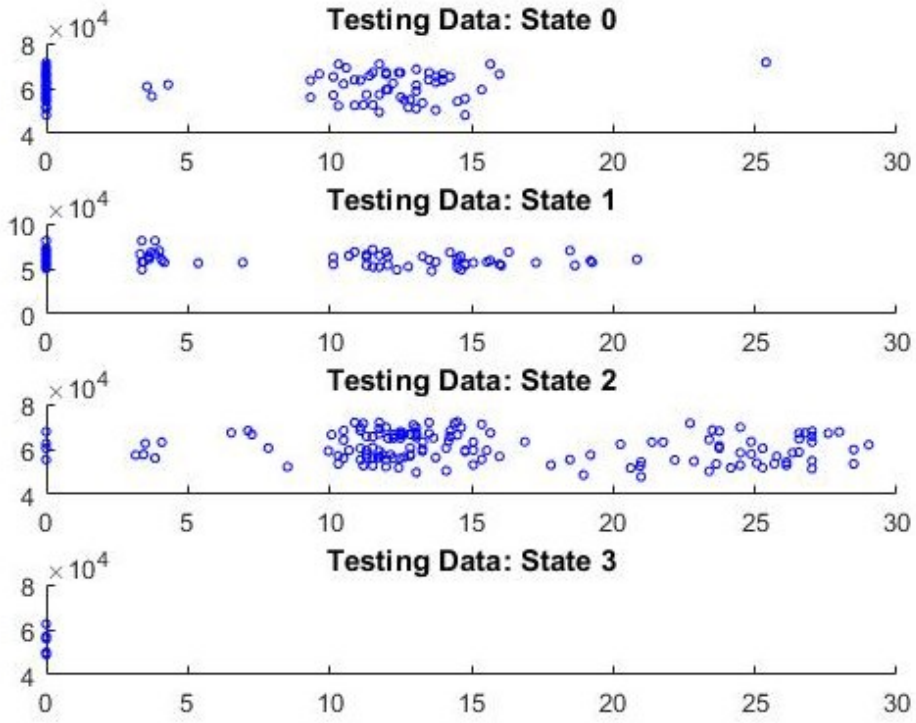

Figure 12: Test Data with States

Analysis

For the experimental system utilizing the modified state design pattern; models of the system performance characteristics were generated for each state using the Isolation Forest algorithm. Parameters for each model were identical as to remove the variance of model accuracy effects due to parameter changes. Due to the nature of unsupervised learning and without labels for normal and abnormal data points, there is not a decisive method to validate models other than performing a visual analysis. This limitation is offset by use of a smaller test set as to not clutter the visual analysis.

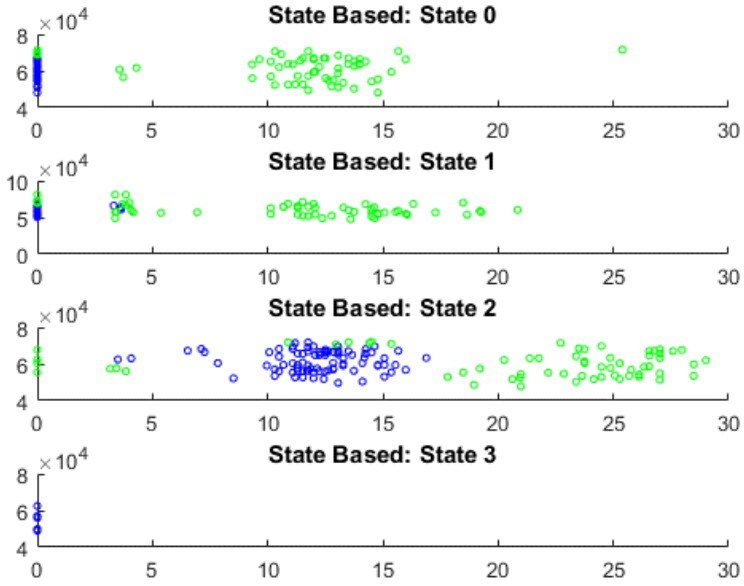

Figure 13: Analysis Modified State Design

Performing the anomaly detection on each state with the test data, which is intermixed with both normal and abnormal readings, the result is clear. The new approach was able to identify nearly all anomalies in each state with minimal error.

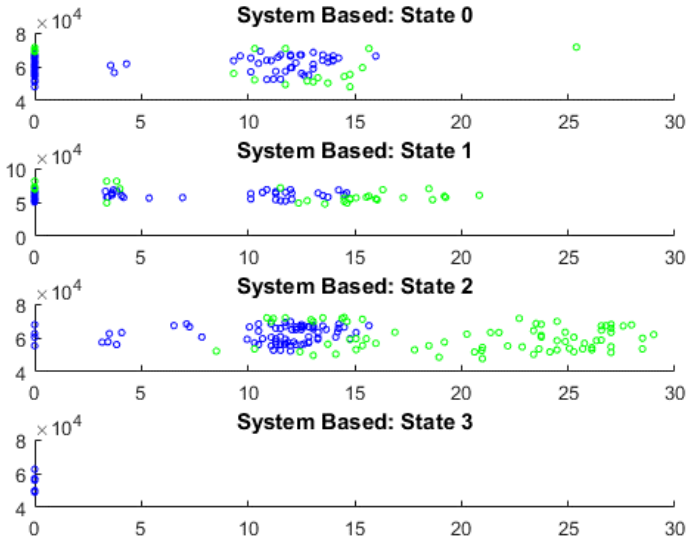

Figure 14: Analysis Control State Design

Consequently, performing the anomaly detection on the system operation in its entirety yields a vastly different result than the system performing state-based anomaly detection. The accuracy of modeling a system as a whole can be seen as far weaker than the former as system characteristics are incorrectly classified.

Reviewing these results side by side creates a clearer analysis, unveiling the benefits of this approach. That is that either approach can easily identify anomalies for system operation that is far outside of the normal operational characteristic. However, anomalies that are within normal system operation as a whole will remain obscured. This observation is clearly evident when reviewing State 0 and State 1, as these states have much lower resource utilization than State 2.

Combining the raw data from both the control and experimental solutions we can perform a rough numerical analysis of the results. Metrics in the numerical analysis are below:

- False Positive: Experimental model identified as normal, control as abnormal.
- False Negative: Experimental model identified as abnormal, control as normal.
- Percentage: Number of misclassifications overall, in percentage.

Table 1: Numerical Analysis

	Overall		0		1		2		3	
	No State	State	No State	State	No State	State	No State	State	No State	State
Inlier:	272	247	125	99	68	58	74	85	5	5
Outlier:	134	159	25	51	33	43	76	65	0	0
False +:	24		2		8		14		0	
False -:	49		28		18		3		0	
Samples:	406		150		101		150		5	
Total Misclass:	73		30		26		17		0	
Percent:	45.91%		58.82%		60.47%		26.15%		0	

We can see that there is large overall gain in anomaly detection accuracy. However, this is seen predominately in states that differ from the dominate or average state of the system overall.

Chapter 5
Conclusion

Overall

Concluding the research and analysis performed to validate the potential impacts of securing our embedded devices at the design stage through the use of machine learning in the state design pattern, the results indicate that this area still requires additional review. However, the results also indicate there being a noticeable improvement over the current methods of anomaly detection in embedded system and perhaps this work could be applicable in a similar fashion on general-purpose systems.

Future Work

Expanding upon the results from this research can continue in several diverging directions. If further research proves this concept outside of the simulation environment there is potential that we could see a change in how we secure our Embedded Devices through the use of design at the development stage.

Utilizing our modified State Based design pattern on actual hardware would be the first step in industry acceptance and use. With actual hardware in the testing environment various performance metrics can be researched and costs evaluated. Additionally, several machine learning algorithms would need evaluation; again, with performance costs and accuracy evaluated. Once numerous potential solutions are evaluated a down selection would occur, and the result implemented through the use of a system service or deamon.

This robust implementation would hypothetically provide protection from power-on to power-off of the device and could potentially evolve to an external device onboard future embedded system.

References

[1] Statista Research Department, "Internet of Things - number of connected devices worldwide," Statista, 27 November 2016. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[2] E. Duffin, "Forecase about the development of the world population from 2015 to 2100," Statista, 19 July 2019. [Online]. Available: https://www.statista.com/statistics/262618/forecast-about-the-development-of-the-world-population/.

[3] R. Kozik, M. Choras, M. Ficco and F. Palmieri, "A scalable distributed machine learning approach for attack detection in edge computing environments," *Journal of Parallel and Distributed Computing,* vol. 119, pp. 18-26, 2018.

[4] M. Ficco, "Internet-of-Things and fog-computing as enables for new security and privacy threats," *Internet of Things,* vol. 8, 2019.

[5] J. Clement, "U.S. cyber crime response lifecycle 2018," Statista, 16 May 2019. [Online]. Available: https://www.statista.com/statistics/194119/average-time-span-until-a-cybercrime-incident-is-resolved/.

[6] J. Clement, "Cyber crime: number of breaches and records exposed 2005-2018," Statista, 2019 August 5. [Online]. Available: https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/.

[7] Cybint Cyber Solutions, "15 Alarming Cyber Security Facts and Stats," Cybint, [Online]. Available: https://www.cybintsolutions.com/cyber-security-facts-stats/. [Accessed 10 August 2019].

[8] K. G. Mehrotra, K. C. Mohan and H. Huang, Anomaly Detection Principles and Algorithms, New York: Springer, 2017.

[9] F. T. Liu, K. M. Ting and Z.-H. Zhou, "Isolation-Based Anomaly Detection," *ACM Transactions on Knowledge Discovery from Data,* vol. 6, no. 1, pp. 1-39, 2012.

[10] M. Rouse, "IoT Agenda," November 2018. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/embedded-system.

[11] M. Samek, "State Machines for Event-Driven System," 4 May 2016. [Online]. Available: https://barrgroup.com/Embedded-Systems/How-To/State-Machines-Event-Driven-Systems. [Accessed August 2019].

[12] E. Gamma, Design Patterns Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1994.

[13] C. P. Kruger and G. P. Hancke, "Benchmarking Internet of Things devices," in *Industrial Informatics (INDIN), IEEE International Conference*, Porto Alegre, 2014.