Improving Machine Learning based Intrusion and Anomaly Detection on SCADA and DCS using Case Specific Information

Peter Prjevara - Dima van de Wouw

Research Project 1 - System and Network Engineering

February 11, 2018

Abstract

Herein we propose a novel perspective into the workings of Intrusion and Anomaly Detection within SCADA systems. We examine the effectiveness of combining contextual knowledge of the system and Machine Learning to create a Network based Anomaly Detection model. The main focus of this paper is on finding the useful contextual information and improve on current proposals of similar researches. As part of this project, a virtual test environment using Python was developed, allowing for easy reproduction of the tests conducted. The research resulted in the development of a novel model, which enables modelling of more complex processes. This was tested through a Proof of Concept Algorithm. The proposed model also offers efficiency improvements on the researches previous research proposals. The research further hypothesises that better response to anomalies can be achieved by the implementation of the proposed model.

1 Introduction and Problem Statement

Ample amount of research into best practices of Intrusion / Anomaly Detection (IADS) for Supervisor Control and Data Acquisition (SCADA) Systems and Distributed Control Systems (DCS) has been done, or currently ongoing. This is due to the phenomenon that critical infrastructure nowadays is operated through a DCS (examples include power plants or water treatment systems etc.) and that more and more of these systems are getting connected to the internet (for instance to enable remote real-time monitoring, or third party remote access for maintenance). These systems often use well-known protocols, which can make it even more feasible to perform attacks for an outsider. As a result, the number of confirmed attacks against control systems are on the rise [18]. Even though the risk is seemingly rising owners of these systems are increasingly pushing for more interconnection, in an effort to "reduce manufacturing and operational costs, enhance productivity and provide access to real-time information" [20]. Even if systems are not connected to the internet however, they are still vulnerable to attacks: one of the reported successful attacks happened through infecting the system through a USB flash drive (Stuxnet Worm [22]), an other has been conducted by a person with direct access to the system (Maroochy Waste Water System Attack [23]).

There are several attempts to classify Anomaly detection strategies for control systems, for instance based on the way evidence is collected [10]. However, even if the recommendations made by Fovino et al. are followed and Network based Intrusion Detection is used, best practices are still not straightforward to define and so the simple models used in the solutions of the researchers can't offer all encompassing protection. This is because all SCADA systems vary in their structure, and the process behaviour they are supervising is not always linear or sequential. In an effort to alleviate this issue, a form of safety system is normally implemented as part of the the control system. Such safety system however is expensive, and only capable of acting after an anomaly was detected by the standard input devices of the system. They are also knowledge based systems, designed precisely for the particular SCADA environment that they are implemented in, so they are not generic solutions. Depending on their sensitivity they might also decrease system availability, due to downtime introduced by bringing the process to a safe state. These observations underpin the need for developing more effective, generic IADS methods.

Many researchers came up with proposals that take advantage of some form of Machine Learning model, that makes the anomaly detection solutions less exclusive. Researchers who try to take advantage of Machine Learning are using unsupervised algorithms, which rely on the assumption that no information is available on the system itself [10, 2, 3]. This seems unrealistic as the deployment of an Intrusion Detection System would most likely be ordered by the owner of the control system itself - who should be cooperative to provide at least some specific information on the control system's structure, if not all. These proposals also experiment with single Machine Learning models, and so the resulting system is only capable of learning either the sequential / linear features

of the process, or the behaviour of a specific process value. As a result, it is hypothesised in this paper that more complex relationship model between process values and sequential features would be beneficial for the purpose of Intrusion and Anomaly Detection.

2 Related Work

2.1 Intrusion Detection Systems

The field of Intrusion and Anomaly Detection is well established and is ongoing since the mid-eighties [10]. In their work, Fovino et al., produce a state based intrusion detection system where a Virtual System Image is generated based on the detailed design knowledge of the system. This virtual image is then used to compare to the real-time state of the system. Their approach is very accurate, however the effectiveness is highly dependant on the actual data throughput of the system. In their previous work, they also propose interesting methods that can be used to model systems for anomaly detection purposes [8, 9, 16, 17].

In a similar research to the above, Hadeli et al. works with deterministic information retrieved from configuration files to create and effective intrusion detection method. They developed a parsing method for the IEC 61850 "Goose" protocol that is capable of interpreting configuration information to be later used as base for anomaly detection on a power system [12]. They claim that these "formal system descriptions" are "often available", hence underpinning the validity of the research question proposed herein.

Kleinmann and Wool proposed an IADS that leverages Statechart based Deterministic Finite Automata modelling and uses unsupervised learning algorithms to create Discrete-Time Markov Chains. The resulting "healthy" virtual image of the system is then compared to data received from real-time packet capture on the system to find anomalies. The research project also used modbus and S7 protocols to create a test environment, and achieved a 99.6% accuracy rate [15]. In certain cases, their solution ran into a problem of a so called "state explosion", which was eventually prevented by a dual layer statechart builder model. Later in this paper, a differnt method is proposed to achieve the same result, using case specific information.

In their work Caselli et al. created a Sequence-aware IADS. They were using similar methods as proposed by Kleinmann and Wool, and they achieved low false positive rates [3]. Their solution relies on a machine learning model that builds a state-chart and then analyse the "transition probability" between states. In the detection phase, the resulting virtual model and probabilities are compared to the actual captured states of the system. They also note that leveraging the semantics and specific parameters of the DCS can be a powerful way of improving the detection efficiency, however they do not implement or test this.

It has been observed that the all the IADS models proposed by previous researchers are all complementing some form of a conventional Intrusion De-

tection System like a firewall, (normally SNORT) to detect obvious anomalies within the network communication of a SCADA system.

Boukema and Lahaye were investigating which level of a process control system would be most suitable for anomaly detection. The result of their investigation was that this can be done between Levels 0 and 1 most efficiently (between actual devices and PLCs). Their research included implementation of proof of concept, where they used Machine Learning (ML) techniques to create a universal product that is capable of learning the healthy behaviour of the system. In their experiments, they achieved 100% accuracy, however due to the limited amount of experiments the findings are not statistically significant [2].

2.2 Artificial Intelligence and Machine Learning

In their PHD thesis, titled Supervised Sequence Labelling with Recurrent Neural Networks, Kazuya Kawakami and Alex Graves proposed to use a combination of different types of neural networks to create a more effective recognition technology for handwriting and test recognition. He achieved good results throughout his experiments, which suggests that using hybrid machine learning models to solve complex problems can be beneficial [14].

A similar research from 1995 done by Bengio et al also underpins this idea [1]. Their research is focusing handwriting recognition using a combination of hidden Markov Models and Neural Networks, which is a similar model proposed within the final sections of this paper, but within the scope of SCADA system modelling.

2.3 Attack Models

Spennenberg et al, in 2016, created a virus that is capable of infecting Siemens PLCs in a SCADA system. The worm does not delete the original program of the PLCs, but creates an additional function block to be executed by the PLC, making the worm capable of staying hidden once installed. The worm is also capable of spreading over to other PLCs, by simulating the behaviour of the TIA PLC programming environment of Siemens [24]. Their paper proves that attacks are possible on the lowest layers of a SCADA system, and they suggest basic detection methods, that in combination with the proposed models from the previous section, can create effective and all encompassing IADS solutions.

The story behind the Stuxnet virus paper by Bruce Schneier also describe a similar attack surface, with a real life example of the lowest levels of an Iranian Nuclear Power Plants SCADA system being infected by a worm [22]. This attack could have been detected if a Network Based IADS system is in place, by alerting of the suspicious network traffic generated by the worms that spread from the HMI Level to the PLCs of the control system.

Common in these attack models that they both target the PLC (Level 1) layer of the SCADA system, which is the last layer before the actual controller process, as will be visible on Figure 1 in The Taxonomy of a SCADA System section (Section 4).

The list of examined attacks is not exhaustive and there are several other attack models are possible, however the aforementioned considered the most impactful. As it is visible in the Taxanomy of a Scada System section (Section 4, the lowest computerised Level of the system is the PLC level (Level 1), where attacks can cause unnoticeable damage (by outputting healthy values towards the HMI whilst generating unhealthy behaviour within the process), hence considered the Level to be protected. This paper focuses on developing a detection method for Level 1.

3 Research Questions

From the desk study conducted it could be distilled that there are several unsupervised and supervised IADS concepts are available, all of which are effective within its own constraints. The models proposed therein offer high detection rates for within their specific scopes, however they are ineffective for different processes types and hence there is space for improvement.

As these solutions rely either on Single-Model ML algorithms (such as the Discrete Time Markov Model or Neural Networks), or full descriptive knowledge of the system, it is hypothesised within this paper that the combination of Multi-Model ML algorithms combined with some contextual information about the system will result in more generic modelling technique, and a more effective IADS.

In order to be able to develop this novel, hybrid model, the following research questions are proposed and investigated in the further parts of this paper:

"What information can be used to complement the information generated by ML algorithm(s), to improve the efficiency and accuracy of a ML based IADS, and make it useful for both seuqential and self regulating processes?"

"How can this information be best combined with the ML algorithm(s)?"

To provide context for the research questions, the hypothesis and the experiments, the following sections will briefly discuss the structure and common terminology of a SCADA system, and the mathematical models used in Machine Learning.

4 The Taxonomy of a SCADA System

In their whitepaper, the System Administration, Networking, and Security Institute (SANS) provide the following diagram of a control system that depicts the architecture of a control system. They also include recommendations for IADS positioning with the lowest level for IADS proposed being between Level

1 and 0.

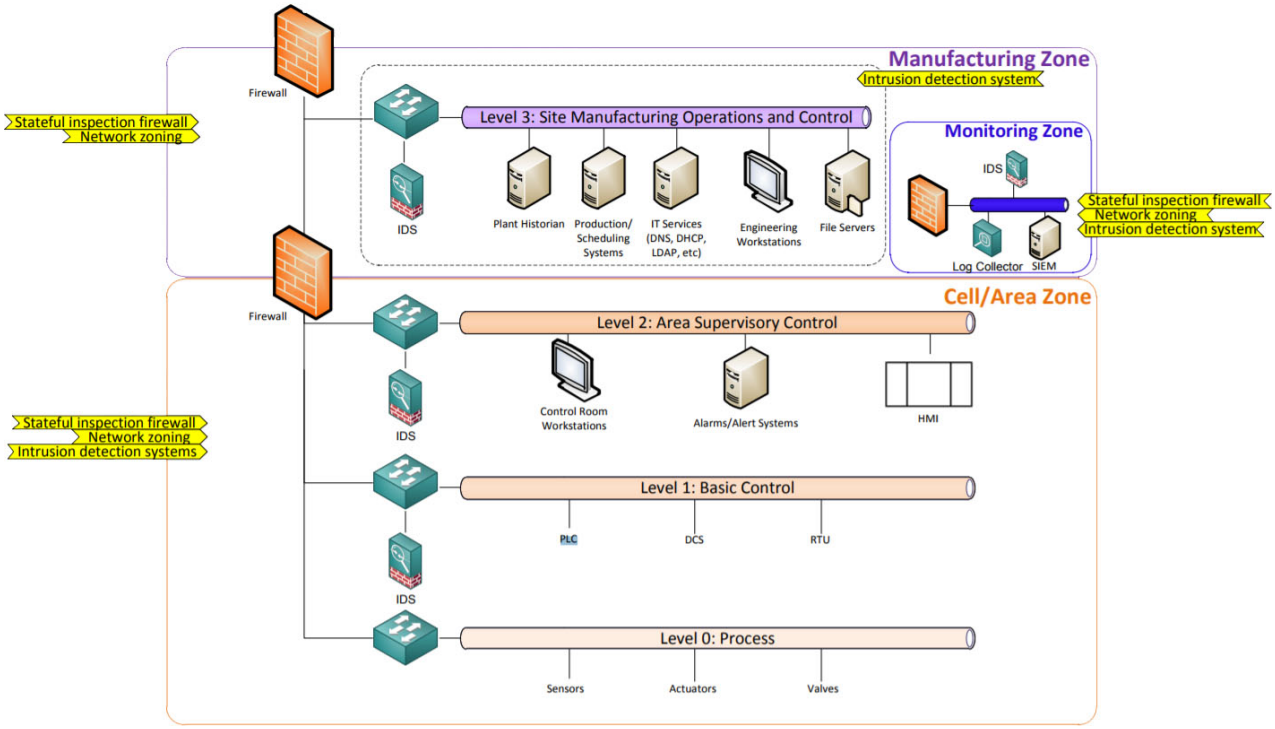

Figure 1: Standard architectural diagram of SCADA with IDS[20]

The control of the real world process is handled within the Cell/Area Zone. On Level 0 (Process) lie the actual devices that interact with the environment, and the *sensors* that provide information on the environment. On the Level 1 (Basic Control) level, one can find the computers that digest and respond to the sensory signals, by commanding the devices (*actuators and valves*). The computers that reside at this level are normally purpose built Single Board Computers (*SBC*, often called Programmable Logic Controller *PLC*). These often use different protocols to communicate with the level above which brings another layer of complexity to the development of an IADS. Above these two levels reside the *Human Machine Interface (HMI)*. The HMI is responsible for providing an "eye" into the process for operators, and consists of components that provide audio-visual feedback and the ability to send commands to or reprogram the PLCs. The level of complexity of information representation decreases as the levels progress from 0 to 2: this is due to the fact that the one of the purposes of PLCs is to structure the data using their respective protocols (for instance *modbus or S7Comms*), to then enable the HMI to display the data in a human comprehensible format.

4.1 The Ideal Level for Anomaly Detection

As it is visible on Figure 1 on Page 6, there isn't a de facto standard for the level on which an IADS system can be implemented. Boukema and Lahaye argued that the best place to implement an IADS would be between levels 0 and 1, because the direct information received from the devices is almost impossible to falsify [2]. That would however require all the individual devices to be monitored separately, and that can be extremely difficult to achieve especially in large systems which might contain hundreds of devices.

In addition, as all of the examined attacks rely on reprogramming or hijacking the PLC level, it is hereby proposed that it is sufficient to implement a network based IADS between Level 1 and 2, which this paper is focusing on, taking advantage of the organised data structure offered by the Level 1 protocols such as modbus or S7 Communication (S7comm).

4.2 Examples of Processes Controlled by SCADA Systems

SCADA systems have a wide range of use cases. The specific use case determines the required Level 0 devices and the implemented program logic. The program logic will depend on the process type controlled by the SCADA system, which can either be a repeating sequence of steps, like in the example of the Bottle Filling Sequence Diagram, or a regulated process such as down regulation of liquid flow by valves. The different process types are depicted on Figure 2. Any combination of the two process types can exist in the same system: as an example, consider the case of the bottle filling sequence, where the flow through the filling hose can be self regulated. This results in a infinite variety of possible system architectures, which adds to the complexity of finding a universal, generic model for an IADS system. Knowing that different process types can coexist within the system allows to understand, that none of the models proposed by researchers in the Related Work section will offer all encompassing IADS solutions.

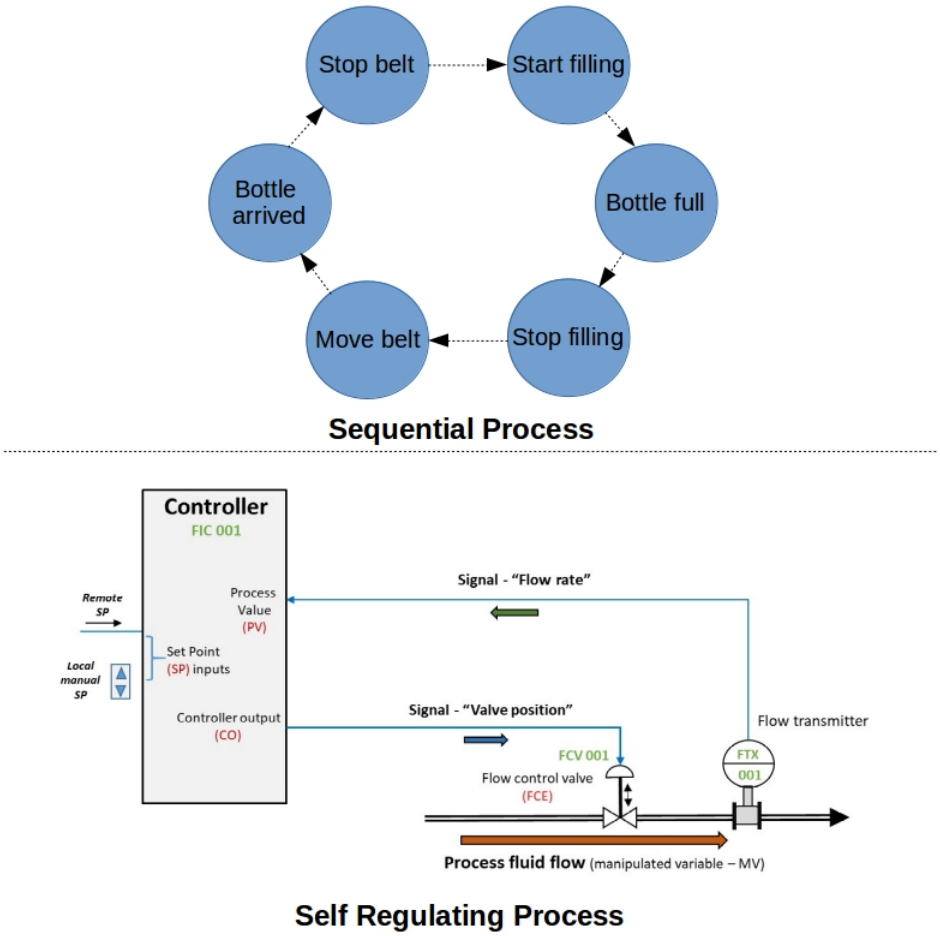

Figure 2: Different Process Types [26]

4.3 Defining the "Full Knowledge" of SCADA

Based on the discussion in the previous section and the conducted desk study, it is possible to define the information that comprehensively describe a SCADA system.

The team of Fovino et al reached very good anomaly detection rates using a knowledge based IADS. Their final artefact relied on the knowledge of the list of devices, their controlling PLC, the modbus addresses used for the device signals on the PLC, and the critical states of the system (defined by analogue value ranges, and unwanted state descriptor functions). [10].

Based on the research of Hadeli et al, discussion with subject matter experts and personal experience, the following list of data points are also considered to be useful to describe a system:

- Device types - analogue or digital
- Communication protocols used within the system
- The different process types that exist within the system
- List of process sequences and their length
- Causal and logical relationship between the devices and sequences
- Age of equipment
- Maximum allowed tolerance for anomalies

In order to be able to answer the first research question effectively, a Proof of Concept (PoC) IADS application is designed and experimented with. During the experimentation with the PoC this list will be reduced to the most necessary information, that can't be generated from the information gathered from capturing packets from the testbed SCADA environment, and is tested using the PoC.

5 Machine Learning Models and Principles

As part of the PoC, a novel Machine Learning based Intrusion and Anomaly Detection concept is developed and experimented with. One goal of the PoC development is to create an artefact, that allows easy implementation and testing of different machine learning models and algorithms. There are several different types of Machine Learning models that are used by researchers, each of which is more effective in detecting trends within data from different contexts. Due to this careful investigation is required before a specific ML model is selected to be implemented for a particular problem. Fortunately, nowadays there are several libraries available online alongside implementation examples, that ease the development of algorithms that take advantage of ML models, some of which is mentioned within this section, and used later in the PoC.

5.1 Neural Networks

One large group of Machine Learning models are Neural Networks. The term Neural Networks is inspired by the workings of the human brain and biological system [25]. The term is used to describe several layers of highly interconnected mathematical functions (so called "neurons"), that work together to create a model of a problem, in order to be able to answer "what if" questions related to the modelled problem. As part of this paper's scope, due to the successes observed in the research project from Boukema and Lahaye, the effectiveness of the LSTM Recurrent Neural Network model was considered. Early on the process of experimentation with the PoC it was discovered that the correlations are better modelled with other types of machine learning algorithms, so the idea of using Neural Networks was dismissed.

5.2 SVM - Support Vector Machine

SVM is a model that can be used to group values into clusters. The mathematical formulation found on the website that document the python API that was used for the experimentation states: "A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. [5]". For the particular problem modelled, this Machine Learning model seemed to be the most suitable, based on comparison of charted data output of the PoC. It will be possible to observe in later sections that the data produced by the PoC resembles the chart depicted on Figure 3 well.

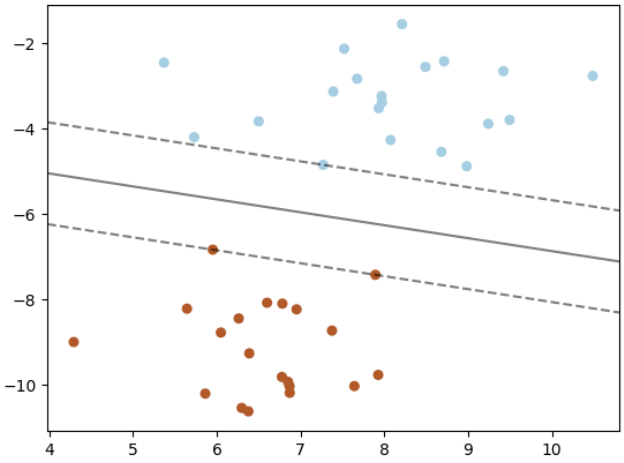

Figure 3: SVM Hyperplane Example[5]

According to the above referenced documentation, SVM is more effective with multi-dimensional models, such as the one produced by the herein used 'RBF kernel'. As we are working with 2 dimensional data only, K-means clustering was also considered to be an option for modelling the data. K-Means clustering can especially be more effective for wider datasets as show on Figure 4.

5.3 K-Means Clustering and Other Clustering Models

The developers of the above mentioned modelling library also created an example performance comparison of two clustering methods, SVM and KRR (Kernel Ridge Regression). As the SVM model was performing better with the random dataset, KRR was considered, but quickly dismissed.[6]

In addition, another clustering technique called K-Means clustering was briefly examined (see example in Figure 4, and should be tested against the results and performance of SVM, as it might be able to provide better predictions for a not so narrowly optimised processes (such as the virtual test environment, described in the later sections).

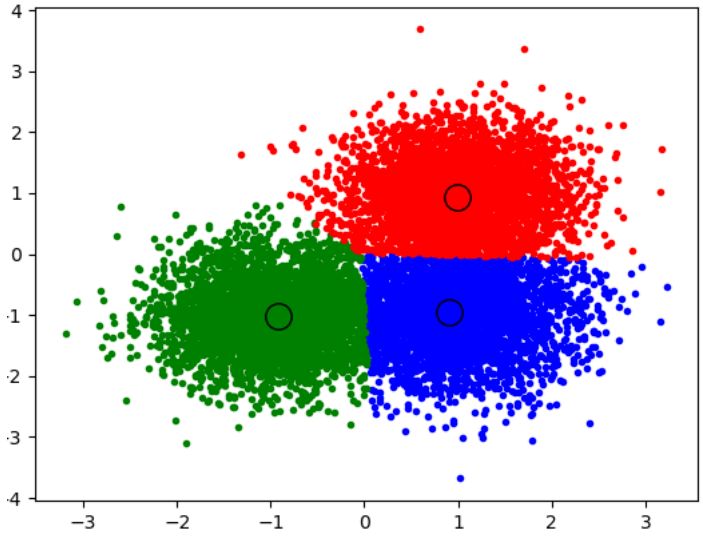

Figure 4: An Example of the K-Means Clustering Modelling Technique[4]

5.4 DTMM - Discrete Time Markov Model (or FSM - Finite State Machines)

The main use case of FSMs are wide ranging: they provide a simple model for describing a sequence of states which take place one after the other, in a repetitive manner. DTMMs are the same, however they focus on measuring the probability of transitions from one state to the other [11]. This model allows the prediction of future states, solely based on the current state of the system. Besides allowing for sequence modelling, due to measuring the probability of state transitions, it is often used to model more complex problems too (for instance in the field of finance). [19] The DTMM modelling technique provides the basis of the herein proposed model, combined with one of the previously mentioned modelling techniques.

6 Method of Research

6.1 The Virtual Test Environment

As the availability of process control systems is critical, they are rarely made available to researchers and developers to run tests on. As a result, most academic researchers build test environments to prove their hypothesis, making the results hard to reproduce. Every time a concept needs to be re-tested, the test environment also has to be re-built. One aim of this research project is to make the results more reproducible by creating a *deterministic* virtual test environment based on the *virtuaplant* python library [13]. The library can be used to create process control simulation environments, based on realistic physics using computer game physics engines. The final simulation produced by the library consists of two applications, in between which the communication is done using a real-life SCADA protocol. In the original software, only modbus TCP is implemented. To prove the portability of the virtual environment, its sequential were ported to a Siemens S7 PLC (as depicted on Figure 5).

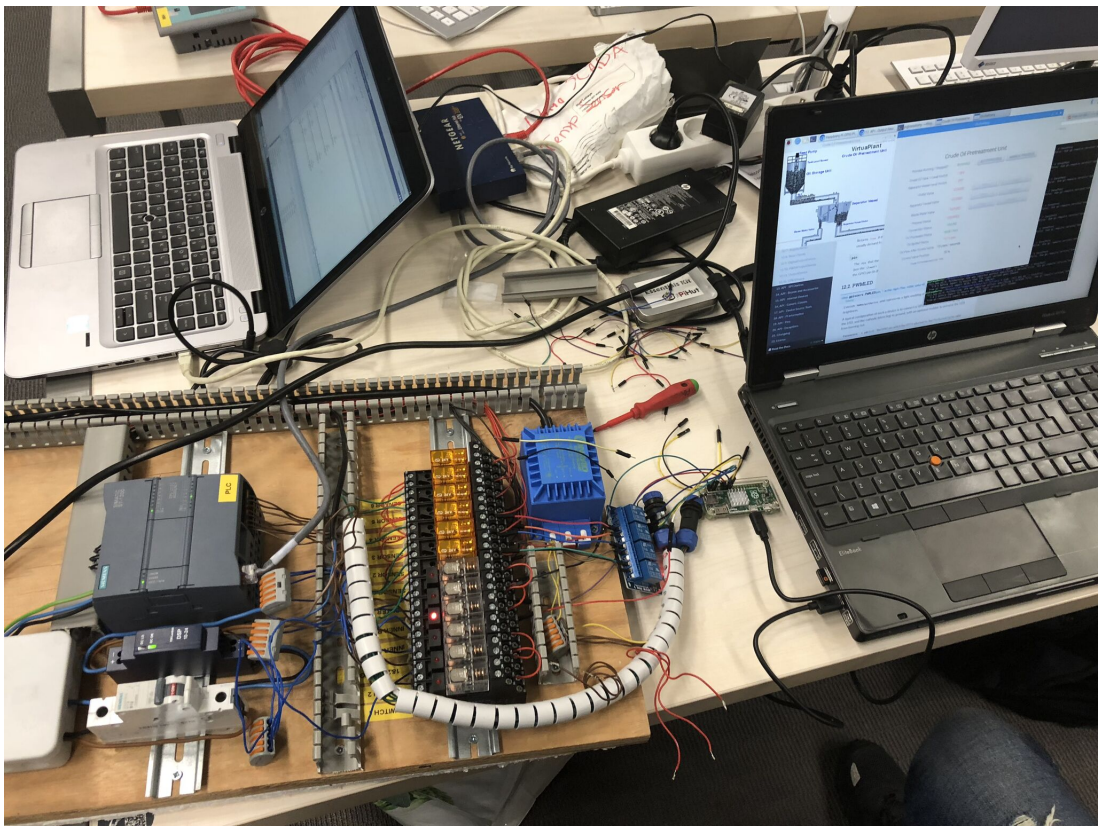

Figure 5: The Siemens S7 PLC and the *virtuaplant* Simulation

6.1.1 The architecture of the environment

There are two separate applications are present in the system: One application serves as the HMI (Level 2 from Figure 1 displayed in Section 4) of a SCADA system, the other is the PLC (Level 1) and the virtual world (Level 0) merged. The merged PLC / world level is depicted on Figure 6, and it will be referred to as virtual world from this point.

In the *oil-refinery* example, a simple form of liquid flow control is simulated. In it's original form the library is capable of simulating 3 digital valves, that can be commanded to open or close through the HMI and a tank level sensor that stops the pump when the tank is full. Besides the digital valves an analogue valve was implemented as part of this project. The analogue valve is attached to a PID controller, and behaves as depicted on Figure 6: when the flow after the valve is higher than the setpoint in the PID, the valve closes to regulate the flow of particles. An additional tank sensor was also added to the Separator, and the following sequence has been implemented.

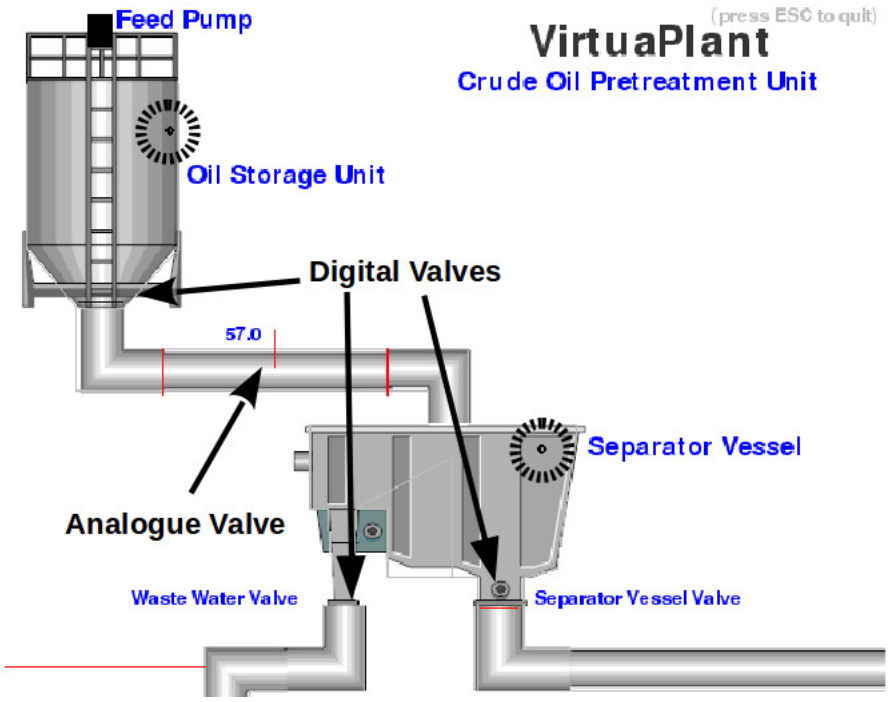

Figure 6: The Virtual World Application

The *virtuaplant* also contains sensors that produce signals to provide information on the current state of the environment. These sensors are highlighted with dashed circles on the above diagram, and there are also flow sensors after the Main Tank Outlet Valve, the analogue valve and the Separator Vessel Valve. The signals from these sensors are processed in the virtual world application,

by a PLC like algorithm, which then sends the information to Level 2, to the HMI, using the modbus protocol.

6.1.2 The characteristics of modbus

Modbus operates using memory space mapped to registers. Each signal normally has it's own register assigned to it. These registers are accessible for reading and writing through specific modbus function codes. The function codes depicted in Table 1 were implemented within the virtual environment:

Function Code	Purpose in modbus	Purpose in *virtuaplant*
03	Read Holding Registers	Get status of all devices
06	Write Single Register	Command single device

Table 1: Data passed to the IADS in a CSV format

The register addresses used are listed in the description of the Parsing Phase of the PoC description in Section 7.1. The virtual environment is used to aid the development of a modbus packet dissector algorithm, that is the basis of the statechart model building algorithm. The extracted information from the packets sent within the virtual environment is used to test and refine a new model, that is built using multiple machine learning techniques and describes the behaviour of the vitrual system. The architectural decisions related to the model is described within the next section.

7 The Proposed Model

Through the desk research conducted and described within the Related Work section (Section 2), it became clear that the current models used by researchers to create IADS systems are effective, but only within their own constraints. Since there are modelling solutions available for both self-regulating (analogue) and sequential processes, a combination of these models could enable the creation of an all encompassing anomaly detection solution that is capable of modelling a complex system. In order to be able to select the correct models would require some pre-existing contextual information about the processes fed to the modelling algorithm. Other researchers have already also hinted on this idea: besides creating an effective anomaly detection solution, Caselli et al proposes that "leveraging semantic of ICS communications and parameters is a powerful way to enhance security tools knowledge of the environment". The great results achieved by the fully knowledge based system proposed by Fovino et al underpin the validity of this suggestion.

7.1 Selecting the Information

Through analysing the TCP packets captured and dissected, it can be concluded that the following information can be learned with certainty by simple methods:

- Whether a signal value is digital or analogue
- The protocol used for the communication (protocols normally use a dedicated port)
- The sequence of events using the DTMM modelling technique proposed by Caselli et al and Kleinmann and Wool
- Existing process types - if only digital values are present in a system, the process can only be sequential

From the full knowledge section, the following datapoints are still not known, and it is proposed within this paper that they are fed to the IADS sytem algorithm, in form of a CSV file. The signal and device list is normally available in a CSV format (regularly referred to as an I/O list), and hence thought of as a simpler way of inputting the information to the system, than the one proposed by Fovino et al.

- Signal / Device names and the addresses used by the protocol
- Maximum error tolerance
- The age of equipment
- Logical groups and correlations within the processes
- Irrelevant information, that should be ignored within the calculations

If the signal / device names are fed to the algorithm, the state space explosion experience by Kleinmann and Wool can be avoided due to the existing knowledge of the devices and addresses to be monitored. The above list is an extended one compared to the list proposed by Fovino et al. If logical groups and irrelevant information are fed to the algorithm, performance improvements become possible. It is hypothesised herein that besides focusing on using proposed models of researchers, new correlations between data points can also be found, and fed to clustering algorithms such as the SVM.

This section partially concludes the answer for the first research question, which result will be further emphasised through the following parts of the paper.

7.2 Proposed Correlations

At the beginning of the development of the PoC application and the Virtual Test Environment, it was hyphotesised that the following data points generated could create added value, if correlated using statistical clustering algorithms, or neural networks:

- The time difference measured between state changes
- The time the system spends in a certain state

In the case of the virtuaplant environment, these values determine the state of analogue values. For example the time spent in a certain state determines the frequency of how often a state appears within a certain timeframe. A higher frequency will potentially result in a more volatile analogue value measurement within the same timeframe in the same logical group, or within other logical groups (and vice versa). Figure 7 highlights the logical groups present within the *virtuaplant* test environment.

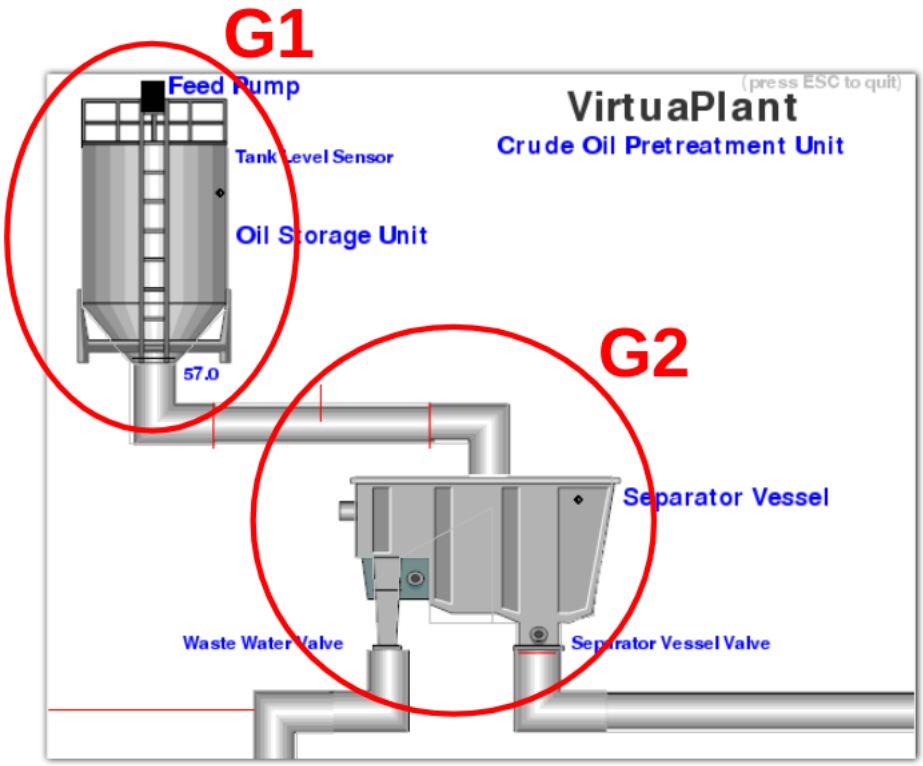

Figure 7: Logical Groups within the Simulation Environment

Within the PoC algorithm, the focus was based on on Logical Group 2. The frequency of the digital values changing within the group are defined by the flow rate of the control valve in the middle of Figure 7 (open 57% at the time this photo was taken). After the desk study it was thought that the time that it takes for changes to occur (Time Delta - depicted as Td on Figure 8) should be related to the actual flow rate. During the testing of the PoC it was discovered that the Time Delta values don't change at all. As a result of this discovery it was concluded that the time spent in a particular state (depicted as Tl on Figure 8) should be measured instead.

7.3 Summary of the Model

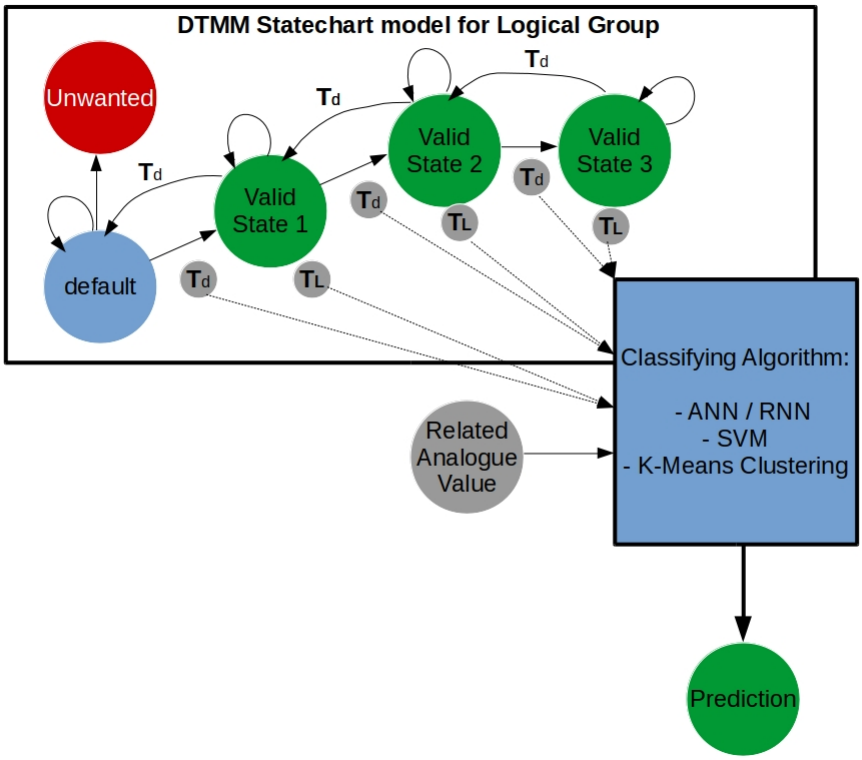

Figure 8: The proposed combination of Machine Learning Algorithms and Information

8 The Architecture of the Proof of Concept IADS

The PoC is designed with a modular mindset based on open source libraries, in order to ensure possibility for expansion. The following modules were implemented within the current version:

- I/O Parser
- Data Objects: ModbusObject, System State
- Modbus Packet Dissector: based on pyshark / wireshark python libraries
- Statechart Builder with Logical Groups
- Statechart Time Analytics

The modules are run in 4 sequential phases: the default learning phase, the digital statechart building phase, the model training phase and the model enforcement phases. This provides an area of a clear improvement area: as multiple core machines are getting cheaper and cheaper, implementing multiple threads instead of the current sequential structure should be possible. The following sections will explain the step of sequences in detail and will provide some pseudocode for representation, however all the working code is accessible on the Github repository of the project [21].

8.1 The Parsing Phase

When the IADS starts up, it searches for the I/O list. During the development of the application it has been observed that the learning logic can get significantly more complicated if all the items proposed in the previous section are learned dynamically. As a result, in the current version of the software the data described in Table 2 (displayed in the Appendix) was passed to parser within the CSV file, to ensure focus can be placed on model concept development. The elements of this table are parsed into modbus objects.

Definition 8.1. Modbus Object

A Modbus Object is a python data object. It is a *tuple* with 5 elements in the format (Tag Name, Address, State, Digital, Logical Group) with Tag Name being of type string, Address, State, Logical Group being of type Integer, and Digital being of type Boolean.

8.2 Learning the Default, Starting State

In the virtual environment the HMI requests the full state of the PLC and virtual world every second using Modbus function code 3. The starting state is defined to be the state when the system is in a settled state. If this state is recorded first, it can be avoided that the IADS raises a false positive, when the system is shut down for maintenance. The default values for each device are recorded per logical group, and converted into a string.

8.2.1 A note on Scalability

The resulting string is then hashed using the SHA256 algorithm using the python *hashlib* library, for so it can be easily compared with newly captured states. This supports scalability of the proposed solution, as the size of the output of the hash algorithm will be the same, regardless the amount of objects are in the system. Expressed in terms of the Big-O notation, the resulting comparison algorithm is O(1), as it is possible to store the states in a hash table.

Definition 8.2. System Statechart

The System Statechart is a python data type of *dictionary*, with n *keys* (where n equals to the number of logical groups given at the Parsing Phase) in the format of { 0:{}, 1:{} ... n:{}}. Each key block itself is pointing to a dictionary, which contains the hashed states as dictionary keys. Each of these keys point to another dictionary, where the hash of the possible state successors and the interesting parameters of a state and state transitions are stored. Figure 9 on the next page and Listing 1 below describe the structure of the System Statechart.

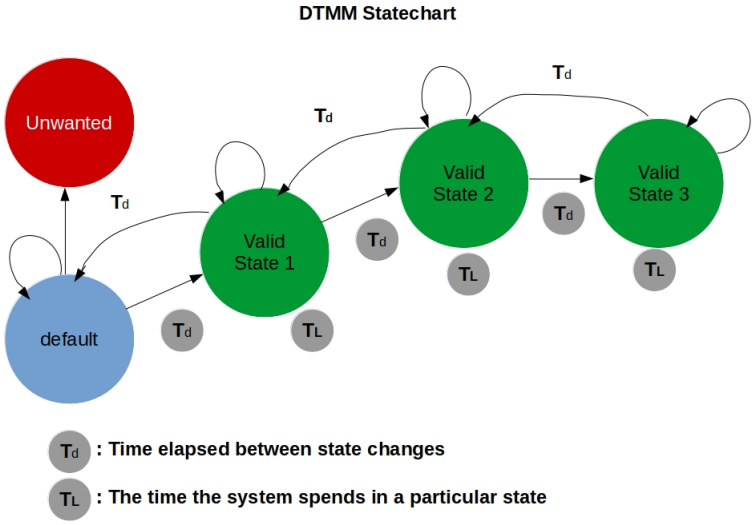

Figure 9: The Resulting Statechart Model of the Valid States of the System for a Single Logical Block

```
{0: {'fa7b6d0b65...84c72ff215': {'successors': ['fa7b6d0b65...84
    c72ff215'], 'prob': [0], 'time_delta': [0], 'time_in_state':
    [0]}},
1: {'b29d59cb44...f1e03e3302': {'successors': ['b29d59cb44...
    f1e03e3302'], 'prob': [0], 'time_delta': [0], 'time_in_state':
    [0]}},
2: {'7a31010015...cef914ecb2': {'successors': ['7a31010015...
    cef914ecb2'], 'prob': [0], 'time_delta': [0], 'time_in_state':
    [0]}}}
```

Listing 1: Example of a System Statechart with the Default States Recorded

The statechart building algorithm takes note of possible successors, the number of occurrences of a particular successor. This way it is easy to calculate Time Delta (the time it takes to advance from current state to new state), the time spent in previous state, and the probability of the state occuring for the period of training. The states are separately recorded for each of the digital values within each logical group. When a state is recorded, it becomes its own first successor immediately, with all the interesting values of the state transition initialised to 0.

In order to make sure that the default state is recorded, the IADS compares the hash of consecutive state captures for a certain period of time given by the user. If more than one hash is found per logical group, the system changed state, and the algorithm terminates and ask the user to bring the system back to stable, default state. The Algorithm 1 depicts the default statechart building prorcess.

8.3 Ignoring the Irrelevant Information

It is proposed herein that in order to ignore irrelevant information, they should be organised into a specific logical group (for instance group 0 in our case). This group can then be easily excluded from the calculations, improving overall running speed of the algorithm.

8.4 Learning the Possible Digital States and Analogue Ranges

When the timer expires for recording the default state, phase advancement occurs and the IADS allows for state changes to happen. In this phase, it is assumed that the system starts normal, anomaly free operations. If the same state occurs as previously, the occurrence counter is increased for the current state. When a different state from the previously recorded occurs, it is added to the list of states, and to the list of successors of the already recorded states, with occurrence counter initialised to 1. The current time is recorded, so "in state time" in a certain state and "time delta" between state changes can be measured. This process runs for a given amount of time. The length should be more than the maximum length of a full process cycle of a logical group from all logical groups. When the learning is finished, the probability values can

```
i = 0
defaultLearningPhase = 100 #LENGTH OF PHASE (s)
startTime = now()
for Each captured packet do
    if now() - startTime >defaultLearningPhase then
        advanceState()
    if FuncCode == 3 then
        i++
        for Each Register in FuncCode 3 Modbus Block do
            if Register is in AllModbusAddresses and
             ModbusObject.logicalBlock != 0 and ModbusObject.digital
             then
                currentState = registerState()
                currentState = SHA256(currentState)
                if i == 1 then
                    addStateToDefaultStateChart(currentState)
                else if currentState not in defaultStatechart then
                    exitWithError(state_changed)
            exitWithError(unknown_device)
        end
end
```

Algorithm 1: Recording the Default State

be calculated for the Discrete Time Markov Chain, and stored within the data structure. This is a similar approach to what was recommended by the research of Caselli et al and Kleinmann and Wool, and results high detection efficiency against replay attacks, or introducing unknown states within the system. Due to the fact that the device list and possible addresses are recorded, the state space explosion can be avoided at this stage of the process. The resulting statechart is stored for each logical block existing within the system, in the datasrtucture described previously. Irrelevant values should also be ignored in this phase.

Besides building the healthy digital statechart of the system, within this phase, the healthy analogue state ranges for the devices are also recorded and analogue values within the same logical group can be correlated with each other, and time using one form of a clustering method such as what was recommended by Boukema and Lahaye. According to their research, this method can effectively predict behaviours of analogue values.

The optimal results described later in the paper were achieved through a 300 second training period. Some experiments were conducted with 150 second and 600 second intervals, but they yielded worse results and hence are excluded

from the paper. This time of course is specific to the virtual environment, and should be adjusted for different environments.

8.5 Correlating Statechart Parameters with Analogue States

Next, the prediction of the correlations happening, using the selected algorithm. Depending on the length of the learning phase, and the resulting data generated, this phase might take longer. Through the development the testing phase have never been observed to take longer than 0.9 seconds. This number however will increase proportional to the amount of correlations examined. The learning phase should be chosen based on the length of logical sequences as overfitting can occur, as the virtual environment is deterministic. If the model is overfitted, that results in too strict predictions, which can result in wrong predictions [7].

Figure 10 depicts the prediction of Time in State for Logical Block 2 in relation to analogue valve behaviour, generated by the PoC algorithm using SVM clustering. The line is the prediction, which is generated by the support vector described using the datapoints within the next diagram.

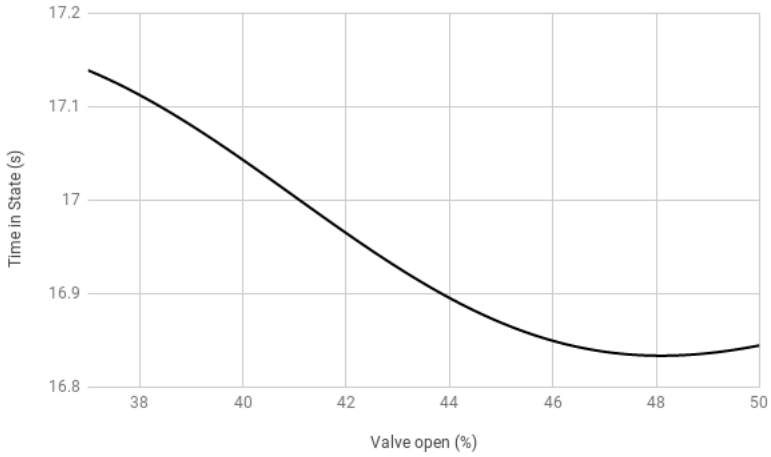

Figure 10: Prediction of Healthy Behaviour of Statechart based on Valve Behaviour

The chart matches the expectations: it can be clearly seen that if the valve is open wider, the time spent in a certain state decreases (this state is when the digital valves of Group 2 are closed).

For this reason it is concluded that this is an effective model to combine the contextual information with information produced by Machine Learning, settling the second research question, and emphasising the validity of the answer to the first research question (detailed in Section 7.1 Selecting the Information).

Listing 2 details the technique used to receive this result. Using the *GridSearchCV* function allows to select the best parameters of *Gamma* and *C*, and promises optimum prediction [6].

```
svr = GridSearchCV(SVR(kernel='rbf', gamma=0.1), cv=3, param_grid
    ={"C": [1e0, 1e1, 1e2, 1e3], "gamma": np.logspace(-2, 2, 5)})
```

Listing 2: Parameters of the SVM Modeller

8.5.1 A note on testing other models

Due to receiving promising results with the SVM modelling technique and the short time allowed for the project, the use of the more complicated Nerual Network models was dismissed, but the use of K-Means Clustering technique should be implemented and results should be compared to the test results achieved from the SVM technique.

8.6 The Enforcement Phase

After the predictions and models are built an enforcement phase follows, when the actual state and previously registered system state can be compared to the modelled values also considering the allowed maximum error tolerance. The enforcement state was not implemented, however it can be seen from here that using a similar pseudocode as used in the learning phase, the current and previous system states can be extracted from the packets. When predictions of the models do not meet the virtual system model, the "anomaly counter" can be increased and alarms can be raised. The same thing can happen when the state observed is not in the statechart, or particular transition is happening more often than the measured probability. Due to the correlations drawn up between analogue states and parameters of the state chart, it is possible to detect attacks that take over the PLC level and obfuscate unhealthy analogue values with healthy ones.

In short, the following detection algorithms can be implemented based on the above described system model:

- Unknown States
- Unknonwn Transitions
- Anomalious Frequency of Transitions
- Anomalious Analogue Value Behaviour
- Anomalious Behaviour in one logical group, based on values measured in another logical group

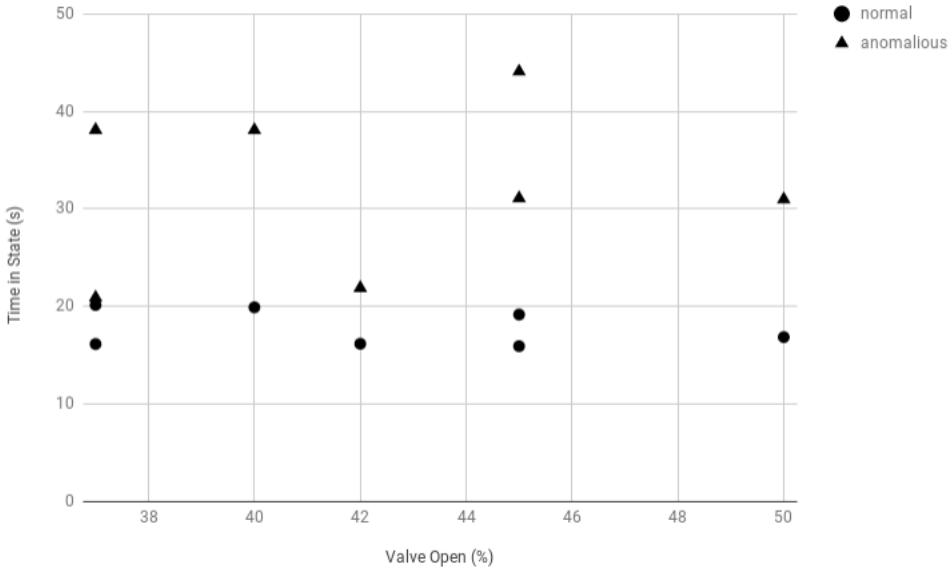

Figure 11: Normal States / Possible Anomalous States

Figure 11 shows an example of anomalious behaviour plotted against normal behaviour. The data points of the normal behaviour produced the prediction depicted on Figure 10. It is visible that due to the existing model of the healthy working range depicted on Figure 10, the anomalious values would be highlighted. The particular attack surface simulated therein is similar to the one used in the Stuxnet attack: the PLC, responsible for the control of the analogue valve is hijacked to output values in the normal range, but the lower than normal flow rate is visible through the plotted correlation between the valve opening and the time spent in state. Due to the lower flow rate, the time spent in state increases. Even though anomalious behaviour can't be observed at the analogue signals received from the valve, the anomaly can detected through observing the anomalious behaviour in the sequence parameters.

9 Results

Through the experiments conducted with the *virtuaplant* environment, it has been observed that the correlation between the Time spent in state property of the digital state chart and the valve opening time can be classified effectively using the SVM modelling method. This has several benefits, such as better ability to pinpoint the part of the system the attack targets, and potentially decreasing rate of false positives.

Due to the fact that some contextual information is used within the system, a reduction in false positive rates can be expected. This is due to the elimi-

nation of the state space explosion, by limiting state storage to logical groups, and ignoring irrelevant data. In addition, logical groups allow for distributed processing of signals (for instance one computer per logical group), and better ability to respond to detected anomalies (by allowing for group specific anomaly response).

The research questions are specifically answered in Sections 7.1 and 8.5.

9.1 Limitations of the Results

The results are specific to the virtual environment, and further testing should be conducted in different environments. Due to the short timeframe allowed for this project, the analogue valve that was implemented in the *virtuaplant* simulation as part of this project is not very well optimised. This results in a "wide" process value, which is visible on the previously shown graphs.

The achieved results are also only an improvement on the proposed model in that they are capable of relating an anomaly to a potential cause: by finding anomalious behaviour through correlations, it becomes possible to tell where the anomaly roots from.

10 Conclusion

Within this paper, besides combining 3 anomaly detection models, we offer a new perspective into thinking about anomaly detection within SCADA systems. The research resulted in a model that is novel within the area of IADS systems. The proposed model is leveraging multiple forms of Machine Learning and it enables correlating parameters of digital statecharts and analogue values that determine the behaviour of the states within the system. This result have been achieved by answering the questions:

"What information can be used to complement the information generated by ML algorithm(s), to improve the efficiency and accuracy of a ML based IADS, and make it useful for both seuqential and self regulating processes?"

"How can this information be best combined with the ML algorithm(s)?"

The research was a combination of desk study, discussion with subject matter experts and the creation of a Proof of Concept algorithm. Besides the useful correlations found, the observed results are easily reproducible using the simulation environment and the freely available code of the PoC. This allows for the search for further correlations within this environment. As the simulation environment is written in python, it is also fairly easy to modify it to simulate other, larger scale / different environments too, allowing for finding other correlations. The digital signals of the virtual environment were ported to a Siemens S7 PLC,

and that allows for testing and developing solutions for a hybrid environment. As the PoC is modular, it easily allows for implementation of new protocols, and the plugging in of new Machine Learning models.

11 Future Work

11.1 Test in Non-Localhost Environments

The tests need to be repeated in a non-localhost environment, with a separate device for the two python applications. The Proof of Concept IADS can be then further optimised to work well within realistic network structures.

11.2 Improvements to the Virtual Environment

It would be desirable for the *virtuaplant* environment to be optimised: the environment physics could be improved, the PID controller could be made to keep the process within a narrower range and further modbus function codes would be desired. The environment should be extended, with further sequences, and different process simulations. Separation of the PLC layer from the virtual world would also be desired, to make the environment more realistic.

11.3 Find Further Correlations, Test with Attack Models and Compare Efficiency

Even thought the time to advance from state to state (Time Delta) hasn't been used to draw up any correlations, it is believed that it could be usefully correlated with the age of equipment. The reason being is that the time it takes to advance from one digital state to another is only dependant on the equipment: for instance the time it would it take for a valve to open or close would mostly be defined by the workings of the equipment. This theory should be tested on real equipment.

Within the *virtuaplant* environment, further correlations should be searched and tested. A couple of proposed correlations are:

- Flow rate compared to In State Time (within Group 2)
- In State time of Group 1 compare to Flow rate / valve open

Using the same environment attacks could be simulated, and false positive rates measured. Comparison to individual models and different model building algorithms could be conducted.

11.4 Optimisation of Code

The code of the IADS proof of concept should be improved. Better organising of code should be prioritised, but further code analysis based on the Big-O

notation principles, threading of the phases and data recording and distribution of signal processing should also be considered.

11.5 Examine Possible Responses to Anomalies

Currently, based on the models, anomaly detection is possible. It would require additional research to see whether it is possible to prevent critical states from happening, or bringing back the system to an ideal state, if anomalies are detected.

11.6 Development of Additional Features

As the Siemens S7 environment is easy to setup, the Siemens S7 packet dissector could be implemented.

Appendix

Tag Name	Protocol	Address	I/O Type	Logical Group
PUMP COM	modbus	1	DI	1
OUTLET VALVE	modbus	3	DI	1
SEP VALVE	modbus	4	DI	2
FLOW SENS	modbus	13	AI	0
WASTE VALVE	modbus	8	DI	2
CALC FLOW	modbus	10	AI	2
TANK1 LEVEL	modbus	2	DO	1
TANK2 LEVEL	modbus	5	DO	2
OIL SPILL	modbus	6	AO	0
OIL PROC	modbus	7	AO	0
OIL UPPER	modbus	9	AO	0
FLOW AFTER	modbus	11	AO	0
CONTROL POS	modbus	12	AO	2

Table 2: Data passed to the IADS in a CSV format

References

[1] Yoshua Bengio et al. "LeRec: A NN/HMM hybrid for on-line handwriting recognition". In: *Neural Computation* 7.6 (1995), pp. 1289–1303.

[2] Anouk Boukema and Rick Lahaye. "Advantages of Anomaly Detection Between the Controlling Unit and the Process Devices of an Industrial Control System". In: (2017).

[3] Marco Caselli, Emmanuele Zambon, and Frank Kargl. "Sequence-aware intrusion detection in industrial control systems". In: *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM. 2015, pp. 13–24.

[4] scikit-learn developers. *K-Means Clustering*. 2018. URL: `http://scikit-learn.org/stable/auto_examples/cluster/plot_mean_shift.html`.

[5] scikit-learn developers. *Support Vector Machines*. 2018. URL: `http://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation`.

[6] scikit-learn developers. *SVR / KRR Performance Comparsion*. 2018. URL: `http://scikit-learn.org/stable/auto_examples/plot_kernel_ridge_regression.html`.

[7] Oxford English Dictionary. "Oxford English dictionary online". In: *Mount Royal College Lib., Calgary* 14 (2004).

[8] I Nai Fovino and Marcelo Masera. "A service oriented approach to the assessment of Infrastructure Security". In: *Proceeding of the first annual IFIP working group* 11 (2007), pp. 19–21.

[9] Igor Nai Fovino and Marcelo Masera. "Emergent disservices in interdependent systems and system-of-systems". In: *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*. Vol. 1. IEEE. 2006, pp. 590–595.

[10] Igor Nai Fovino et al. "Modbus/DNP3 state-based intrusion detection system". In: *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. IEEE. 2010, pp. 729–736.

[11] Paul A Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation*. John Wiley & Sons, 2017.

[12] Hadeli Hadeli et al. "Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration". In: *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*. IEEE. 2009, pp. 1–8.

[13] matlink Peter Prjevara Ike-Clinton jseidl. *Virtuaplant Fork*. 2018. URL: `https://github.com/prjpet/virtuaplant`.

[14] Kazuya Kawakami. "Supervised Sequence Labelling with Recurrent Neural Networks". PhD thesis. Ph. D. thesis, Technical University of Munich, 2008.

[15] Amit Kleinmann and Avishai Wool. "Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.4 (2017), p. 55.

[16] M Masera and I Nai Fovino. "Models for security assessment and management". In: *Proceedings of the International Workshop on Complex Network and Infrastructure Protection*. 2006.

[17] Marcelo Masera and I Nai Fovino. "Modelling information assets for security risk assessment in industrial settings". In: *15th EICAR Annual Conference*. 2006.

[18] Dave McMillen. *Attacks Targeting Industrial Control Systems (ICS) Up 110 Percent*. Article. 2016. URL: `https://securityintelligence.com/attacks-targeting-industrial-control-systems-ics-up-110-percent/`.

[19] S Meyn and RL Tweedie. *Markov Chains and Stochastic Stability, Cambridge*. 2009.

[20] L Obregon. "Secure architecture for industrial control systems". In: *SANS Institute InfoSec Reading Room* (2015).

[21] Peter Prjevara. *Python IADS*. 2018. URL: `https://github.com/prjpet/python_ids`.

[22] Bruce Schneier. "The story behind the Stuxnet virus". In: *Forbes. com* (2010).

[23] Jill Slay and Michael Miller. "Lessons learned from the maroochy water breach". In: *International Conference on Critical Infrastructure Protection*. Springer. 2007, pp. 73–82.

[24] Ralf Spenneberg, Maik Brüggemann, and Hendrik Schwartke. "Plc-blaster: A worm living solely in the plc". In: *Black Hat Asia (p. N/A)* (2016).

[25] Christos Stergiou and Dimitrios Siganos. *Neural Networks. 1996*. 2010.

[26] Wikipedia. *Control theory*. 2017. URL: `https://en.wikipedia.org/wiki/Control_theory`.