CMPT 318 CYBERSECURITY
Term Project

Group 5
Yongzhe Le (301269967)
Darryl Julius Basr (301388030)
Alim Nizari (301328567)

Abstract

The importance of the integration Artificial Intelligence in cybersecurity defense mechanisms has been a growing topic as new advanced threats are being produced rapidly. Anomaly-based intrusion detection systems have the potential to prevent cyberattacks by behavioral analysis of a system or network. We present the performance of anomaly detection systems based on the training and testing of Hidden Markov Models using normal data and comparing these results to anomalous data using the log-likelihood method. Principal Component Analysis is conducted to choose the best suited variables for the experiment. The variables chosen are Voltage, Sub_metering_1 and Sub_metering_2. The results obtained from comparing the training model and the testing model concluded in the testing model being a good fit for the training model as it was neither underift nor overfit. The results obtained from the anomaly detection analysis concluded with the datasets being least anomalous to most anomalous in the following order: Data1, Data3, Data2. Our report shows that the log-likelihood method is a suitable method for anomaly-based intrusion detection.

Table of Contents

Introduction

Anomaly detection is used in an intrusion detection system to detect intrusions in a network by flagging irregular activity in a system. A system must be able to identify both normal activity and anomalous activity for the intrusion detection to function properly. The system must first be capable of identifying normal activity in a network for anomaly detection to work. The system is first trained with network activity (data) that represents normal activity within a network. The next step of anomaly detection requires the system to be trained to correctly identify the anomalies within network traffic (data). The new traffic introduced to the system is filled with anomalous activity for the purpose of training the system. The method of detecting anomalies can differ in each system and the type of model used to detect anomalies varies with each system.

Problem addressed

Due to the increase of automation among critical infrastructure systems, there is also a growing number of threats to the systems. The automation of a system implies the involvement of computers which creates vulnerabilities within the system. The risk involved with attacks on critical infrastructure is immense therefore methods to mitigate the risk are explored in this project. Specifically, the anomaly detection method for intrusion detection systems. The goal of the project is to create and test an anomaly-based intrusion detection system.

Methodology used for solving problem

The model used in this report is a data driven model based on four separate datasets. Using the data, a predictive model for the intrusion detection system is created One data set represents the training data filled with normal observations and the other three datasets represent the testing data which are filled with anomalous observations. A Hidden Markov Model (HMM) is used to build our predictive model. The HMM is based on the idea of the Markov chain. A Markov chain is a model that tells us the probabilities of sequences of variables / states each of which can take on values from some sets. These sets can represent anything, from weather to electricity consumption. A Markov chain gives us a prediction of the sequence's future based on the current state. The states before the current state have no impact on the future except via the current state.

A Markov chain is useful to calculate the probability for a sequence of observable events. However, to compute hidden events, a HMM is useful. A hidden Markov model allows the computation of observable events as well as hidden events that may be a factor in building a probabilistic model. An HMM has 5 major components : 1) a set of n states, 2) a transition probability matrix, 3) the observations, 4) observation likelihoods, and 5) an initial probability distribution over states.

The likelihood of an observation sequence is the multiplication of different probabilities together and their sum. A greater likelihood of an observation sequence means that the current HMM is more likely to generate this sequence compared to other sequences with lower likelihood. A very low likelihood can imply that this sequence does not match with the behaviour that is being captured by our model.

For this report, because the data set is very big, calculating likelihood will require many multiplication and summation of probabilities. Since these probabilities are numbers between 0 to 1, the likelihood will become very small. Therefore, logarithmic likelihood is used (log likelihood) to make computation more convenient.

depmixS4 is a framework for specifying and fitting dependent mixture models, otherwise known as hidden Markov models. Model fitting is done in two steps; first, models are specified through the depmix function (or the mix function for mixture and latent class models), which both use standard glm style arguments to specify the observed distributions; second, the model needs to be fitted by using the fit function. The depmix function that is used in this report takes on 5 arguments; 1) response(s); it's the response or variable that wants to be modelled. It can be univariate (only 1 variable) or multivariate (2 or more variables). 2) data frame; the data that is being analyzed. 3) nstates; the number of states of the model. 4) family; the type of distribution that our response has. This must be a list if it's a multivariate model. 5) ntimes; a vector specifying the lengths of our data. Then calling the print function will show the log likelihood as well as BIC and AIC values.

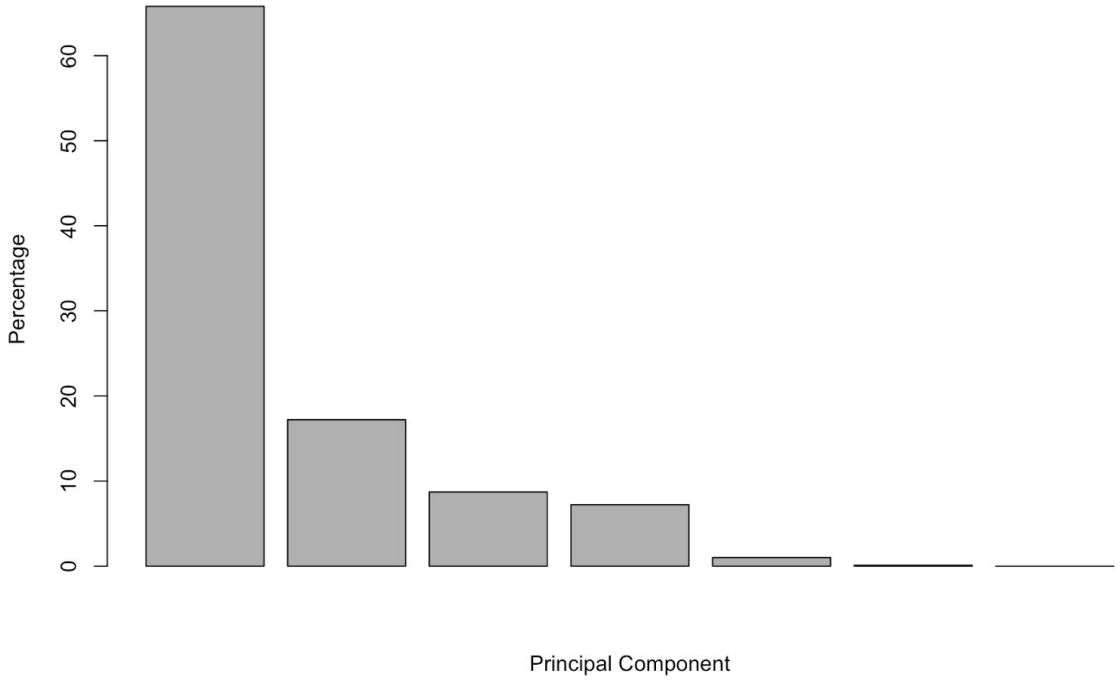
To find the most accurate number of states, log likelihood and Bayesian information criterion (BIC) is important. BIC is formally defined as;

$BIC = \ln(n) * k - 2 * \ln(L)$,

where n is the number of data points in our sample, k is the number of parameters estimated by the model and L is the maximised value of the likelihood function. The best number of state is the one with the biggest log likelihood (keep in mind that log likelihood is a negative number), with the lowest BIC.

To make sure the HMM that is obtained from the train data set does not overfit nor underfit, it is important to test it on a different data set, i.e “test” data set. To do this, first obtain the matrix from our previous model using getpars() function (also from the depmixs4 package) . Then, use setpars() function to set the value of these matrices on our new depmix model (from the test data set). Then normalized the log likelihood if the two data sets have different lengths. For example, if the train data set has 2 times more data than the test data set, divide the log likelihood by 2.

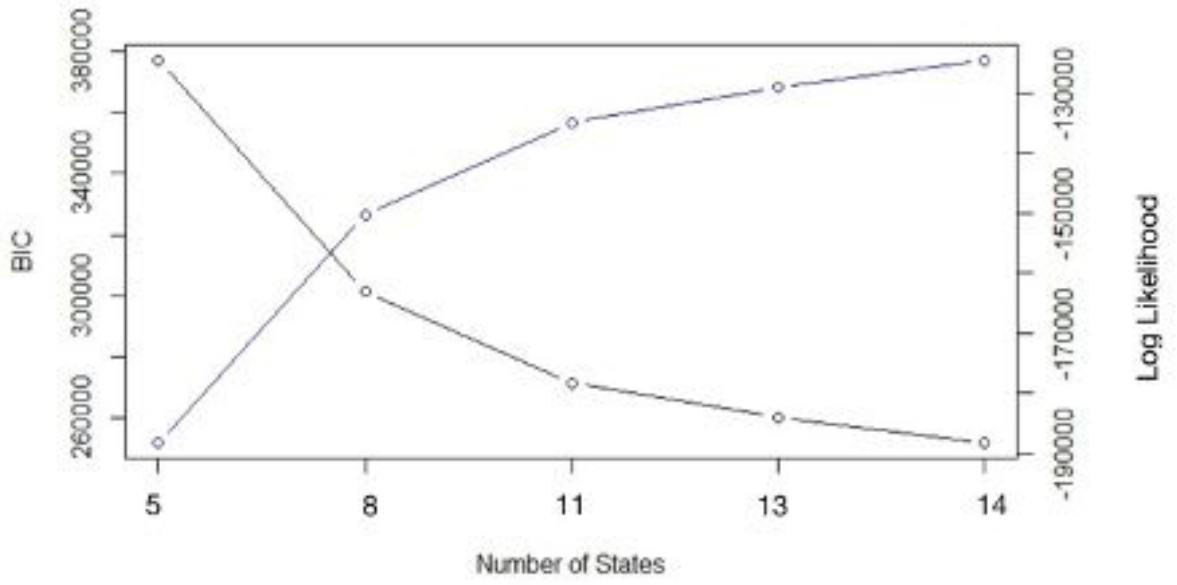
In this report, the multivariate model uses 2 responses. The number of responses used was decided by analyzing Principal Component Analysis (PCA). The PCA showed the top 2 Principal Components (PCs) and top 3 PCs represent 83% and 91.7% of the variance in the data respectively. The following graph is the variance each PC is responsible for represented in Percentage.



Top 2 PCs were chosen because using only responses can greatly simplify the HMM but still covers a large percentage of the variance in the data. Voltage and Sub_metering_1 were . These variables are chosen from analyzing the Principal Component Analysis (PCA). The PCA While the number of states ranges from 4 to 20 and the best state is chosen. The family lists are gaussian distribution and Poisson distribution. The first response which is Voltage requires gaussian distribution because it's a continuous type of response which has real numbers. While the second response, Sub_metering_1, is a count type of response which has integer numbers, so it uses Poisson distribution. It's also possible to use multinomial distribution for the second response, but computation is more expensive.

The ntimes vector that is used in this report is (240 * number of weekdays in the data set). Because the data in this report is recorded every minute from 2006 to 2009, and the time window that is chosen is from 5 am to 9 am. This time window being choose is because most people wake up and go to work during this time period. There is a clear trend in electricity consumption in that time period throughout the yearw. Also note that it is possible for the data to have NaN values, so it is important to modify the ntimes such that these NaN values are not being included when making the model.

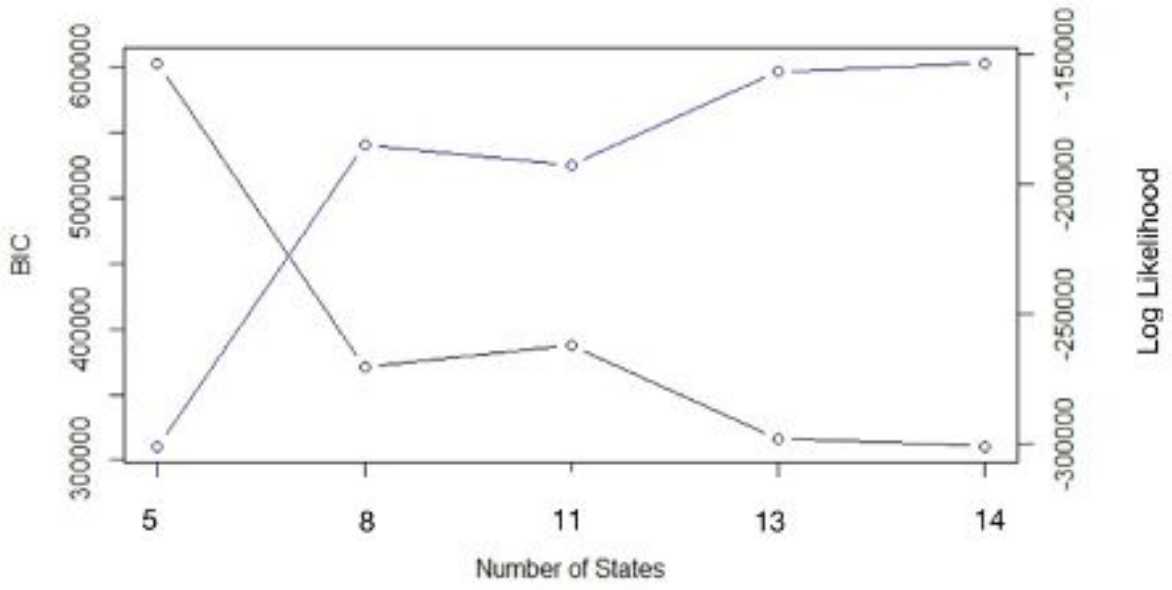
Univariate Model



The graph above uses the training dataset, where it is subsetted into weekdays and a time window from 05:00 to 09:00. It shows the log likelihood and BIC from computing the hidden Markov model for the Voltage response.

From the graph above, the number of states (nstates) 5, 8, 11,13 15 gives us the most interesting cases. As can be seen from the graph, state 5 has the highest BIC value while also has the lowest log likelihood, 376889 and -188244.7, respectively. When nstates is equal to 8, the log likelihood is -150322.4 with a BIC of 301573.4 , which shows the improvement when increasing nstates. nstates equal to 11, gives an even better result where the log likelihood is -137382.3 and a BIC of 276433.7. For states above 11, i.e 13 and 14, the fit function stopped because it reached the maximum number of iterations and still wasn't able to converge. This shows that the function is not able to create a model that fits the data sets given with that number of states.

The Graph below shows the relation between Number of States and the Log Likelihood in our Multivariate Model



When the state number is 5, it gives a log likelihood of -300857.5 with a BIC of 602173.4. In state number 8, it gives a much better result; log likelihood is -185197.4 and BIC is equal to 371417.4. However, on state 11, the log likelihood decreases to -192945.2 and BIC increases to 387688.7. This shows that it's harder to match the behaviour captured by the model when the state number is 11. On state 13, the log likelihood gives a result of -156875.3 and the BIC of 316183.6. On state 14, again, the fit function is unable to converge when it reaches the maximum number of iterations. Therefore nstates=13 is chosen to be the best number of states for this model.

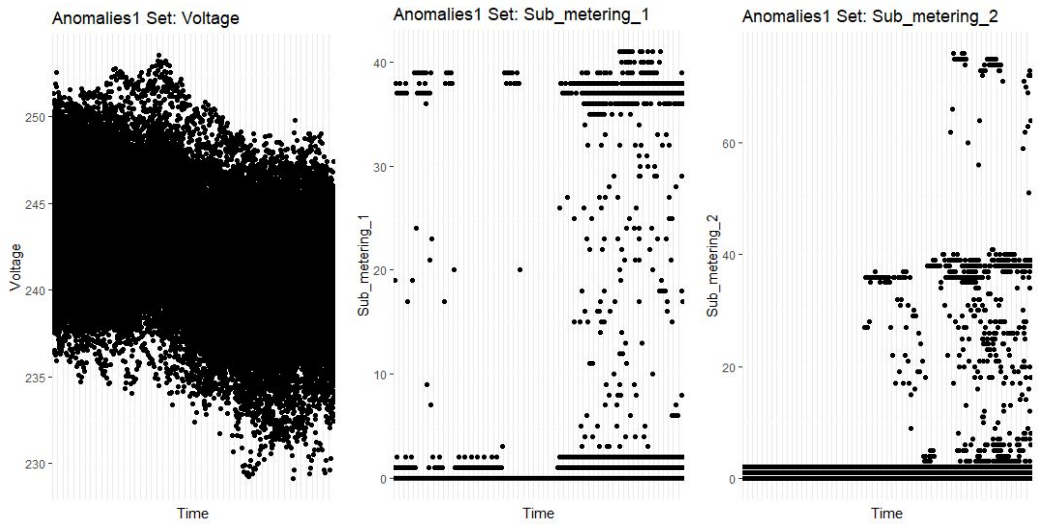
Then the next step is to test the model in a different data set to see if its underfit or overfit or just right. The result shows that the model is neither underfit nor overfit.

The test sequence gives a log likelihood of -77614.19. Compare to the train dataset, when the log likelihood is normalized to -78437.65.

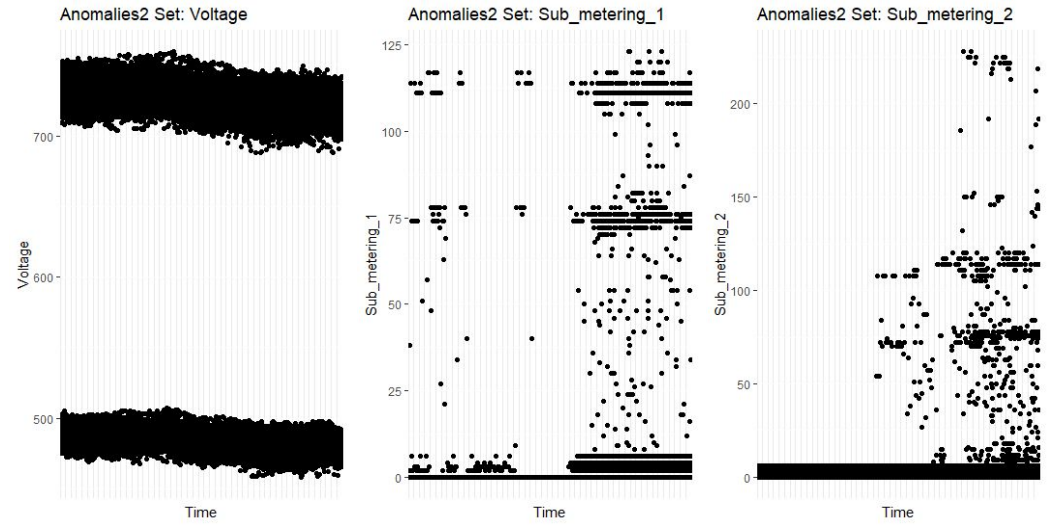
The multivariable HMM generated from the training data was applied to the The three variables used are Voltage, Sub_metering_1, and Sub_metering_2. A single time window of observations is used from all the datasets. The time window used is 05:00:00 to 09:00:00 (24-hour format). Subsets of each dataset are created to include only the necessary variables and observations to allow for an easier process when using the RSTUDIO software. The training dataset over one full year contains 87965 observations and the anomaly datasets each contain 86760 observations. The little difference in the length of the data for the training dataset and the anomaly datasets disregards the need for normalizing the log-likelihood of each dataset. The depmixS4 package in RSTUDIO is used to achieve the tasks for computing log-likelihood. Specifically, the depmix() function within the package is used for the three chosen variables. The gaussian() method is used for Voltage and the poisson() method is used for both Sub_metering_1 and Sub_metering_2.

The following diagrams display the scatterplots for each dataset:

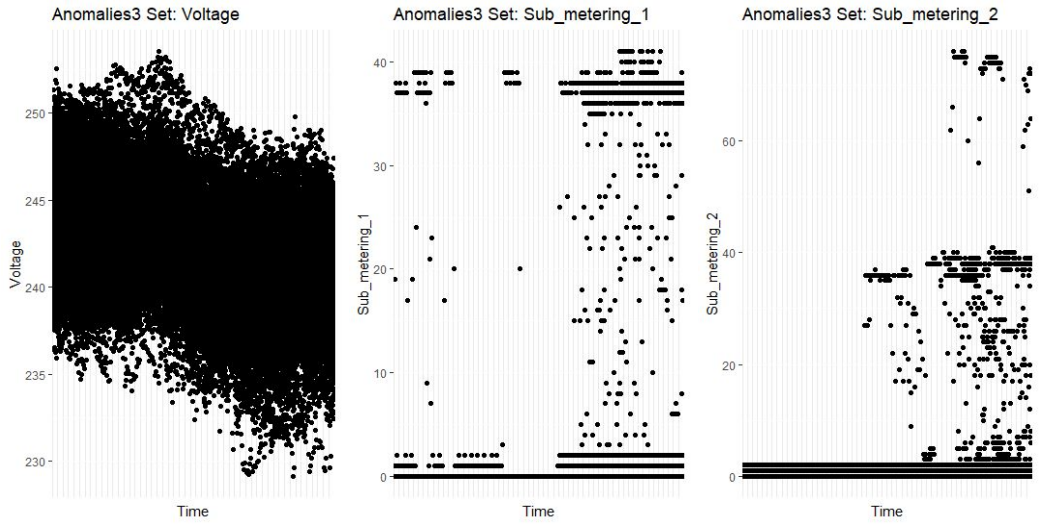
Data1 with injected anomalies:



Data2 with injected anomalies:



Data3 with injected anomalies:



Log-likelihood is a value that produces a prediction on how well the model fits the sample data. The training data that contains no injected anomalies represents a model with normal activity. The testing data with injected anomalies is used to compare against the normal model. Given the three datasets with anomalies, the log-likelihoods of the datasets are computed with a chosen number of states for each dataset. The previous section uses five different states to identify an optimal amount that will satisfy the following conditions: obtaining the lowest log-likelihood and function converging at a finite iteration. The number of states that is chosen to be optimal is 13 states.

The following table displays the results of the program:

	Data1	Data2	Data3
Log-likelihood	-136635.4	-532375.9	-136637.1

Iteration of convergence	167	11	101

The results show Data1 and Data3 having little difference in the log-likelihoods whereas the Data2 dataset differs by a large amount. The dataset with the lowest value for log-likelihood suggests that the data contains more irregular observations. Comparing all three datasets, the log-likelihood of the second dataset, Data2, is the most anomalous dataset. The second most anomalous dataset is Data3 and the least anomalous is Data1.

Comparing the previous HMM section's log-likelihood of -77614.19 for the testing model and -78437.65 for the training model to the log-likelihoods of the anomalous datasets shows a large difference in value. The results from the multivariate HMM shows that the testing set fits well with the training set therefore the training log-likelihood will be used for comparison to the anomalous data. The log-likelihoods of the anomalous datasets are much lower than the training set. The results are expected as the training data does not contain injected anomalies and is much less anomalous than the three datasets with injected anomalies. The log-likelihood method here provides support to anomaly detection systems working as an intrusion detection method for the given data. The outcome shows that the model can accurately determine which dataset is less anomalous or more normal than the other. Giving the predictive model the normal training data succeeded in concluding the training dataset is both a better predictor for real-world models as well as less anomalous than the datasets with injected anomalies.

Problems encountered

The Sub_metering variables are distributed among only a certain set of values which makes it difficult to observe visually. The anomalous datasets contained many zero values for both Sub_metering_1 and Sub_metering_2. This led to some problems with the code and required using a different method afterwards. Due to the slow speed of the depmix function, the poisson() method was used in the depmix(family) function to account for the size of the data. The depmix function is heavily reliant on the system's hardware to run at an acceptable speed. Using this function can take up a large amount of time and can produce unusual results with larger datasets. Therefore, the log-likelihood was calculated only using the Voltage and Sub_metering_1 variables.

Lessons learned

The first lesson learned in this project is using HMM to perform intrusion detection is very resource intensive. Each response variable has to be chosen carefully and the HMM should be as simple as possible. The second lesson learned is HMM does not offer a great way to identify individual anomalies. An alternative method has to be used to identify specific anomalies (i.e. Rolling average). And a subject expert is needed to set thresholds for identifying anomalies. And there is always a trade off between the amount of false positive and false negative. The third lesson learned is the training data should not contain anomalies and has to represent the real world condition in order to have an effective HMM.

Conclusion

This project shows the Hidden Markov model is a very powerful and effective method for performing intrusion detection. However, it is very challenging and labour intensive to build an effective and efficient HMM and have the ability to identify each anomaly. Especially in today's ever changing cyber landscape. Reinforcement learning is a possible way to address some of these issues and keep up with new cyber threats. Following is an essay that briefly introduces this concept.

Reinforcement Learning

Today, there are 4.13 billion internet users globally who are vulnerable to cyber-attacks.[1] Technology encapsulates the world and will only grow further with time. Technology's benefits to humankind are abundant and the quality of life is improving with the growing incorporation of technology in human life. Computers can automate processes that were once done by humans which reduces cost, improves quality assurance, and simplifies tasks. In addition to technology in our daily lives, computers and automation play a huge role in the functioning of the world's critical infrastructures. Critical infrastructures are essential systems, facilities, services, etc. that maintain the functions of society and the economy.[2] The failure of any critical infrastructure could lead to a catastrophic loss and harm to society. Essential systems and infrastructures that were once isolated from technology have now become reliant on it and this has led to a huge growth in the attack surface of our computer networks. The attack surface is the total amount of exploitable vulnerabilities that a system contains.[5] Technology has its benefits, however, the potential of harm done by criminals utilizing cyber attacks is a growing problem in today's world. Humans are currently more vulnerable than ever due to the implementation of computer systems in almost every aspect of human life. The world's cybersecurity is improving however, the number of threats that appearing is increasing at an alarming rate. Security experts are constantly trying to create and develop quality defence mechanisms to render criminals useless. The next section will discuss the integration of artificial intelligence in cybersecurity and how it could potentially prevent any threat from conducting cyber attacks.

Artificial intelligence is quickly becoming the norm within technology and has the potential to create both extremely beneficial and dangerous software. Criminals can utilize AI in cyber-attacks and as the use of AI grows, so do the risks. The development of AI-based cyberattacks suggests that AI-based cybersecurity methods must also be developed to mitigate these attacks. Anomaly detection is one method of defence that could help monitor system behaviour and analyze irregular activity within a system or network. Network or system activity can be translated into a database where datasets are extracted and analyzed by anomaly detection systems. Anomaly detection systems will find patterns in the data that do not align with normal behaviour and flag the data as an anomaly. Unusual activity within a system, otherwise known as anomalies, can be malicious but can also classify as normal activity. Anomalies are not always malicious and finding a method to accurately classify normal observations from malicious ones is difficult. There are issues with the accuracy of anomaly detection systems, specifically the high number of false positives when flagging malicious anomalies. Combining artificial intelligence with anomaly detections allows the exploration of reinforced learning algorithms' abilities to accurately identify intrusions. [3], [4]

The importance of having reliable cybersecurity measures is crucial to the operation and maintenance of human society and the economy. Research on anomaly detection through reinforced learning can help improve current intrusion detection systems in terms of accuracy and response times. In the research paper “Application of deep reinforcement learning to intrusion detection for supervised problems,” the authors suggest that Deep Reinforced Learning algorithms can provide improved performance of intrusion detection over other common Machine Learning models. The researchers compared the results from several Machine Learning algorithms including Support Vector Machine, Logistic Regression, Naïve Bayes, K-Nearest Neighbors, Random Forest, Gradient Boosting Machine, AdaBoost, Multilayer Perceptron, and Conventional Neural Network as well as four Deep Reinforced Learning models: Deep Q-Network, Double Deep Q-Network, Policy Gradient, and Actor-Critic. Two standard intrusion detection datasets were used for the models, NSL-KDD and AWID. The reward function used in this research is 1/0 for correct and incorrect predictions respectively. Results show that the Double Deep Q-Network model had the best performance among all models in terms of accuracy, speed, and efficiency. Researchers emphasize the Double Deep Q-Network model’s ability to reduce the number of false negatives the algorithm detects. Furthermore, the general performance of the Deep Reinforced Learning algorithms provides faster prediction times which is essential for large networks and online intrusion detection. The outcome of the research supports the hypothesis that Deep Reinforced Learning algorithms have the potential to outperform the current standards of Machine Learning. [6]

The key difference between a traditional Machine Learning algorithm and Reinforced Learning algorithm is in the Reinforced Learning algorithm humans no longer tell the machine what is the desirable state. Instead the machine will only receive a reward or a punishment based on the outcome.[7] And it is up to the machine itself to navigate the environment and find the optimal outcome. [7] It is crucial for an effective Reinforced Learning algorithm to have a sensible reward system and an acurite environment that represents the real world. The Markov decision process is an accurate and effective way to represent an environment. [7]

A Markov decision process (MDP) can be formally defined as m= (s,a,p,r,gamma), where; s represents the state of all states, a represents the set of all possible actions, p represents the transition probabilities, r represents the rewards, and gamma is the discount factor. The MDP m will find the best policy, often called a pi, that will come up with the best sets of actions in each state that will yield maximum rewards over a long period of time. As the name suggests, MDP is related to the Markov property, which states that the next state is solely dependent on the current state. This applies to how the agent traverses through the MDP with no memory necessary. One of the most known methods for solving this problem is through what

is called the Q-learning. Q-learning is perfect to use when no explicit probabilities or values are given, so our model needs to learn this by themselves. The model will update its learnings on the Q-table. At first, each cell on the Q-table will be 0, hence the model hasn't learned anything. Then as the model begins to do some actions, each cell will contain Q-values. These Q-values are obtained via what is known as the Bellman equation; Let Q(s,a) be the Q-values that give the maximum total reward by doing actions from states. Then $Q(s,a)= r + \gamma \max_a Q(s', a')$, which states that the total reward by doing actions a from state s is the expected reward for taking action a from state s, plus the discount factor gamma multiplied by the maximum value of the next state. Since gamma is in the range of between 0 and 1(inclusive), it determines how much weight does the value of the next state play in determining the optimal reward. If gamma is set to 0, the model will only focus on the immediate reward and cancel out the Q(s', a') term. On the other hand, if gamma is equal to 1, it means that the model weights potential reward just as much as it weights immediate reward. In practice, gamme is usually somewhere between 0 and 1, where the potential reward has diminishing effects. Since an environment is stochastic in nature, the reward that our model gets on a certain action might be different from an earlier observation. So the Q(s,a) will need to be recalculated with the same formula and subtracting the previously known Q(s,a). This is known as the Temporal Difference. Which have the following equation; $Q(s, a) = (1 – \alpha) Q(s, a) + \alpha (r + \gamma \max_a Q(s', a'))$. Where α is the learning rate of our model, which controls how our model adopts to the changes from the environment. As can be seen, this equation is recursive, but inevitably Q(s,a) will converge to an optimal quality function Q*(s,a), which then can produce the optimal policy, pi, where $\Pi^*(s) = \text{argmax } Q^*(s, a)$. [8]

An intrusion detection system needs to be run 24/7 and it has to adapt to an ever changing threat environment. Reinforcement learning lands very well in such a system due to its high autonomy and the ability to adapt its behaviour to changes and always produce a desirable outcome.

References

[1] J. Clement, *“Number of internet users worldwide,”* Statista, 07-Jan-2020. [Online]. Available: https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/. [Accessed: 04-Dec-2020]. Retrieved December 03, 2020

[2] “Critical infrastructure,” *Wikipedia*, 06-Jan-2020. [Online]. Available: https://en.wikipedia.org/wiki/Critical_infrastructure. [Accessed: 04-Dec-2020].

[3] “Cyber attacks on critical infrastructure.” [Online]. Available: https://www.agcs.allianz.com/news-and-insights/expert-risk-articles/cyber-attacks-on-critical-infrastructure.html. [Accessed: 04-Dec-2020].

[4] D. H. Koduvely, *“Anomaly Detection through Reinforcement Learning,”* Zighra, 07-Feb-2019. [Online]. Available: https://zighra.com/blogs/anomaly-detection-through-reinforcement-learning/. [Accessed: 04-Dec-2020].

[5] “*What is an attack surface – Reducing it and what it is*,” Avast. [Online]. Available: https://www.avast.com/en-ca/business/resources/what-is-attack-surface. [Accessed: 04-Dec-2020].

[6] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, *“Application of deep reinforcement learning to intrusion detection for supervised problems,”* Expert Systems with Applications, 18-Sep-2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419306815. [Accessed: 04-Dec-2020].

[7] *5 Things You Need to Know about Reinforcement Learning*. (n.d.). Retrieved December 04, 2020, from https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html

[8] Andrew Ye Cofounder at Critiq | Editor & Top Writer at Medium, A. Ye, and Cofounder at Critiq | Editor & Top Writer at Medium, *"Markov Decision Process in Reinforcement Learning: Everything You Need to Know,"* neptune.ai, 01-Dec-2020. [Online]. Available: https://neptune.ai/blog/markov-decision-process-in-reinforcement-learning. [Accessed: 04-Dec-2020].

[9] Wikipedia contributors. (2020, November 15). *Anomaly-based intrusion detection system*. In Wikipedia, The Free Encyclopedia. Retrieved 04:59, December 3, 2020, from https://en.wikipedia.org/w/index.php?title=Anomaly-based_intrusion_detection_system&oldid=988901871

[10] *Discover how to build anomaly detection systems with Bayesian networks.* Learn about supervised and unsupervised techniques, predictive maintenance and time series anomaly detection. (2018, September 18). Retrieved December 03, 2020, from https://www.bayesserver.com/docs/techniques/anomaly-detection