Deep Learning for Intrusion Detection on Encrypted Traffic

Christopher Lamprecht*
LMPCHR002myuct.ac.za
University of Cape Town
Cape Town, Western Province, South Africa

Lucas Carr*
CRRLUC003myuct.ac.za
University of Cape Town
Cape Town, Western Province, South Africa

Keywords: Deep Learning, Malware, Intrusion Detection, Networks, Classification

1 Project Description

The last three decades have seen networks and computers become important tools, ubiquitous to modern life. The widespread adoption of these technologies has added a new dimensions of risk, in the form of malicious software, or *malware*. Malware is a term which encompasses several distinct types of software which share a defining feature of installing themselves on a system without permission, with the intention to cause some harm[8] [13]. While there are numerous, important malware sub categories, we have chosen to focus attention on ransomware and botnet malware's. A botnet is defined as a network of hosts - or, *bots*, infected with a malware which gives control of the system to a single operator, the *botmaster* [1]. The botmaster can send instructions to be executed by hosts - enabling attacks such as Denial of Service (DoS), Cryptomining, as well as providing compute resources for other brute-force attacks [1]. Ransomware is a type of malware that locks a user/organisation out of their device(s), demanding a ransom to be paid to the attacker in order to regain access. Since 2018, ransomware has become one of the most disruptive and commonly seen forms of malware [11]. Despite organisations storing valuable information on shared network volumes, to allow for backups, and reduce the effects of ransomware, it is still shown that 65% of these victims lose their data, despite this measure [3]. Hence, it is becoming increasingly more important to combat specifically ransomware, to reduce the affects it has on individuals and organisations.

Historically, networks were secured against malware attacks through the use of a Network Intrusion Detection System (NIDS). A NIDS, functioning correctly, is able to identify and block malware - or, more generally, blacklisted traffic - from entering the network. NIDS were capable of this due to the deployment of a broad range of techniques to identify and classify traffic: specifically, port analysis, deep packet inspection (DPI) and statistical modelling of packet flow [20] [4]. However, more recent standard practice around networking has made it difficult for the aforementioned detection systems to function well. Port numbers have become a less reliable indicator of an application type; moreover, port obfuscation is a technique that masks the destination port of some traffic [20]. Similarly, the adoption of dynamic IP addresses has also made it difficult for NIDS, given that NIDS rely on their ability to associate traffic from specific IPs with devices - a difficult task if the IP address changes frequently. Furthermore, much of modern internet traffic undergoes some encryption protocol; notably, in 2017 $\approx 75\%$ of analyzed malware made use of encryption [18]. Encrypted packets make conventional DPI inspection impossible since they require access to the payloads contained in packets - which are now encrypted [14]. It is important to recognize that these practices are useful, and in many cases, the reason for their adoption is that they offer significant security benefits - port obfuscation, for example, can prevent attackers from targeting specific ports for specially created attacks, since their required ports no longer function as expected.

As a result, novel technologies which perform intrusion detection - capable of working within the previously established networking practices - are required. Machine Learning (ML) applications are examples of such technologies; however, many existing ML-based intrusion detection models require carefully engineered feature selection, which is a slow and difficult task [9] [20] [15]. Deep learning algorithms have been proposed as solutions to this problem - Lotfollahi et al. [9]successfully implemented a convolutional neural network, and stacked autoencoder for the purpose of encrypted traffic classification. The motivation behind these approaches is that deep learning algorithms are well suited to automatic feature extraction, extensible to extracting features from encrypted data [21]. Moreover, deep learning models are capable of reasoning with a much higher dimensionality than shallow ML counterparts - which enables them to learn more complicated patterns [20]. The trade-off of this, however, is that these algorithms can be significantly more expensive to train. Frequently, this is increase in resource requirements is ignored, as measurements such as accuracy are considered more important metrics when evaluating a model.

2 Problem Statement

2.1 Research Problem

There have been demonstrations of using deep learning approaches for both encrypted network traffic classification, and intrusion detection, which have produced encouraging results. However, the literature regarding the union of these two areas, specifically, intrusion detection on encrypted network traffic, is limited. Furthermore, much of the literature on deep learning applications to network classification have not given substantial weight to the computational expense of the models they train - opting to view model accuracy as the most significant evaluation metric. We recognize the importance of considering computational cost as an important metric for evaluating models; in the case of implementing one of these models on a network in an intrusion detection system for real time classification of traffic, a more efficient model would be ideal [20]. This is compounded by the assumption that this software would be deployed on a system without considerable compute power.

We will implement deep learning models as binary classifiers, where the classification task is to recognize encrypted network traffic as either malicious or benign. The scope of malware that will be considered will be limited to instances of ransomware, and botnet sub-types. Three deep learning models will be implemented for both ransomware and botnet traffic. A one-dimensional convolutional neural network (1D-CNN) was chosen due to CNNs effectiveness at recognizing patterns in high dimensional data [12]; additionally, CNNs make use of sparsely connected layers, where the neurons which make up a layer have an imposed limit on how many outgoing connections they can have - this has been seen to significantly reduce the complexity of the model without a large impact on performance [8]. In addition, a stacked autoencoder (SAE) will be implemented. SAEs work by encoding some data into a vector with a lower dimensionality than the input, and then decoding this in order to rebuild a representation of the input data from the encoded data. The reduced dimensionality of the encoded data forces the encoding function to prioritise features important to the input data, acting as a form of dimensionality reduction [20][8]. There is precedent for choosing this model, as successful implementations of SAEs for classifying encrypted traffic can be seen in Zeng et al. [20] and Lotfollahi et al. [9] work on encrypted traffic classification.

The final model we have chosen to implement is a long short-term memory (LSTM) network; we regard LSTMs as a good option for this problem due to their ability to analyze long-term relations in sequential data; this property makes them suited to learning features from network flows, which contain sequential, time-related data [20]. Zeng et al. [20] implemented an LSTM to classify encrypted traffic, which achieved an accuracy of 99.22%.

In addition to this, a simpler multi-layer perceptron will be created to provide a basis for benchmarking these deep learning models against less complex neural networks. The motivation behind the decision to train the models for each malware sub-type separate stems from the recognition that the profiles of ransomware and botnet malware are distinct. Moreover, given the scarcity of literature regarding this problem, breaking the challenge of malware detection into smaller components might be more conducive to successful experimentation.

2.2 Research Questions

This research project aims to determine how effective deep learning algorithms are at detecting encrypted ransomware and botnet traffic - where *effectiveness* is a measure of the accuracy of a model balanced against the computational cost of training that model. It should be noted that this question, as well as the required experimentation steps, will be undertaken for ransomware and botnet traffic individually.

1. How successful - where success is a measure of accuracy and F1-Score - are the implemented deep learning models when classifying encrypted network traffic as either benign or malicious traffic?
2. How successful - where success is a measure of accuracy and F1-Score - is the less complex MLP model when classifying encrypted network traffic as either benign or malicious traffic?
3. How expensive - where expense is a measure of training time, memory usage, and inference time - are the DL models, relative compared to an MLP, for applications of classifying encrypted malware traffic?
4. Which of the implemented deep learning models, CNN, LSTM and SAE produces the highest degree of accuracy?

3 Procedures and Methods

3.1 Dataset Selection

Despite the comprehensive amount of research that has been done on malware detection using DL, the datasets used in the evaluation of the past research are rarely made public. On top of the limited amount of datasets available, the datasets selected need to fulfil the following criteria: they must contain encrypted network traffic, they must contain either ransomware or botnet-specific traffic and they must consist of relatively new (past 5 years) network traffic that resembles the current flow of network traffic and malware. Following these criteria and limitations, the datasets we have found are the Open Access Dataset [2] for ransomware traffic and the dataset provided by Modi et al [10].

The Open Access Dataset contains two types of samples, infected encrypted traffic, captured whilst ransomware encrypts network-shared files and benign traffic, captured by users using benign applications. The infected part of the dataset contains 70 ransomware families, whose traffic was captured over a 50-hour period of ransomware activity. The dataset contains an eclectic variety of ransomware families, so as to gather all the ways different ransomware traffic acts. Additionally, the dataset contains ransomware traffic that has had a high impact in the real world, all taken from the years 2019 and up. The non-infected part of the dataset was collected over a 2527-hour period (7640 files opened every eight hours). This traffic does not contain any identifiable information.

The dataset provided by Modi et al [10] only contains the traffic from 20 ransomware, many of which are already included in the Open Access Dataset [2], hence this dataset will not be used.

For the botnet dataset, the Stratosphere Laboratory, and a 'sister project' called *Malware Capture Facility Project*, provide an extensive repository of traffic capture datasets - Stratosphere Laboratory calls these captures *scenarios*. Each scenario consists of traffic from a specific malware, as well as normal and background traffic, captured over a specified duration - where this duration ranges from a few hours to several weeks. The inclusion of 'normal' traffic is necessary in order to allow the measuring of *False Positives* and *True Negatives* [6]. We selected 25 relevant datasets from the Stratosphere Laboratory's repository to merge into a single dataset for botnet traffic.

The data for each scenario was captured in a .PCAP file, from which bidirectional network flows were extracted and labelled. The labelled bidirectional network flows from each scenario were generated from the .PCAP using the *Argus* network tool, which supports the selection of which fields to be included in the network flow - for example, including the 'pretrans' field will included the percentage of packets retransmitted in the generated network flow. Around twenty five of these 'scenarios' were selected to be merged into a single dataset, representing captures of seventeen different botnet families, amongst other benign traffic. The number of botnet families included was lower than desirable; however, this is a consequence of there not being many publicly available datasets providing encrypted botnet traffic. Moreover, the other qualities of this dataset - specifically, the fact that it balances malware traffic with normal and background traffic, as well as having clearly labelled bidirectional network flows - make it desirable in spite of this shortcoming.

3.2 Pre-processing of Datasets

From the Open Access Dataset [2], three subsets will be made, training, testing and unseen subsets. The training and testing subsets will be made up of 50 hours of ransomware traffic and 50 hours of benign traffic. Out of this, 80% will be used for the training subset and 20% for the testing subset. Ransomware traffic from 7 of the more recent ransomware families will be used to create the unseen dataset, which will be used to test the DL models on unseen ransomware traffic. Lastly, a dataset, made up of 2477 hours of benign traffic will be created that can be used to evaluate the number of false positives that the models return. All the network traffic has been pre-processed into the following format: each line is one sample, each sample has 30 features, the label (1 if it is an 'infected' sample and 0 if it is not) and the timestamp of the last sample interval in the trace (in seconds since the beginning of the trace). The features are separated by ',' because it is a .csv file. The models will use IP payload bytes from each packet as input (not including the label). This method allows for analyzing both encrypted and unencrypted packets and has been shown to improve traffic classification accuracy with deep learning models [9]. Shorter packets will be padded with zeros to ensure they have the same number of features. Normalizing the data will be done to algorithm make the algorithms more robust and reduce training time, which could save valuable resources when working with low-resource networks. The first few bytes of the vector will be masked to ensure that the models do not learn information related to the port numbers. This will also save more resources.

Each dataset from the Stratosphere Laboratory contains the traffic capture in .PCAP files, as well as a labelled, bidirectional network flow. The network flows were generated with the *Argus* tool, and contain 100 fields. These fields capture measurements of a network traffic flow, and provide information corresponding to features such as the duration of the flow, source/destination port numbers and IP addresses, network protocol, and a label of whether the flow represents benign or malicious traffic. The selected datasets - which each contain a network flow, and a representation of this network flow in a .csv format - will be amalgamated into a single dataset in the form of another .csv file. The final dataset will be read in as a dataframe, where fields not considered to be important will be dropped. After this process, the data will be sectioned into *x_train, y_train, x_test, y_test, x_validate, y_validate*. The purpose of this sectioning of the data is for both cross-validation, as well as separating the features of a flow from the label of the flow - for example, *x_train* will contain all the fields of a network flow, while *y_train* contains the label. Together, *x_train* and *y_train* make up a single sample in the dataset. The training set will consist of 80% of the original dataset, while testing and validation sets will be 10% each - this ratio was inspired by the finding of Gholamy et al. [7].

For both the ransomware and botnet dataframes, after the pre-processing steps have been completed, the training, testing, and validation sets will be reshaped into tensors of appropriate dimension for each respective model; for example, a 2D-CNN requires a 4D tensor.

3.3 Benchmarks and Evaluation

To determine whether using DL models is justified, a Multi-Layer Perceptron (MLP) will be utilized as a benchmark to establish the minimum level of performance required. The MLP is chosen due to its simple structure and can be used as a good starting point to establish a baseline performance. Firdausi et al. [5] demonstrated the success of using an MLP for malware detection (91% accuracy), further showing its usefulness as a benchmark model. Additionally, the MLP can be trained and tested using the same input data that is given to the DL models. In contrast, other ML models might require a different format of input data, and not be able to work with the high dimensionality of the network traffic.

Due to the DL models having more parameters than traditional ML models, they can be more computationally expensive. The trade-off between computational complexity and improved ability to learn patterns in the data will also be tested using the benchmark established by the MLP architecture. If the DL models attain higher accuracies but are too computationally expensive, it may be argued that the ML model is a better option for resource-constrained areas such as community networks.

In terms of performance, the ML and DL models will be evaluated based on predictive performance and computational efficiency.

The metrics that will be looked at to test the predictive performance are accuracy, precision, recall and F1 score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precission = \frac{TP}{TP + FP}$$

$$F1score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

The metrics that will be looked at for computational efficiency will be inference time, memory requirements and training time. Since the aim of the system is to detect malware in real-time, it is important that the system can do this quickly. Hence, the time taken to make predictions on new data will be used to measure computational efficiency. The models need to minimise the amount of memory needed to make these predictions. Therefore, the memory usage of the models will be evaluated. This will be achieved by measuring the memory that each model implementation uses. Additionally, the training time of the models will be measured using wall-clock time and batch time.

3.4 Implementation of ML and DL models

Our ML and DL models will be implemented using Python 3. Python was chosen due to its simplicity, flexibility and most of all its powerful libraries that can be used to implement the models efficiently. *TensorFlow* will be used with the *Keras API* running on top of it. These tools provide an easy, yet efficient way of building and testing the models, without sacrificing flexibility and allow for improved memory requirements [19].

Pandas will be used for data manipulation and analysis, providing easy-to-use data structures and analysis tools, and improving efficiency when working with large datasets. *Numpy* will add support for large, multi-dimensional arrays and matrices by providing mathematical functions to operate on these data structures.

The memory used by the models will be tested using Python's trace malloc.

Lastly, *Matplotlib* will be used as a plotting library, providing helpful visualisations for the data analysis.

3.5 Hyperparameter Tuning

Hyperparameters are modifiable parameters that are not learned from the data, but set by the user, before training. The values of the hyperparameters control the learning process, and selecting the right ones is essential to improve performance. Some of these hyperparameters are learning rate, epoch size, number of hidden layers, number of neurons in each layer, etc.

We will choose a random search to find the best values for the hyperparameters. This involves randomly selecting hyperparameter values from a predefined range in order to find the best combination of hyperparameter values. This technique will allow us to explore a wide range of values. The random search will be performed on Python using the *Sci-kit* machine learning framework.

4 Ethical, Professional and Legal Issues

With regards to the datasets, they can contain sensitive user network information, making consent to use the data essential. Seeing as the datasets we will be using are made public, this should not be an issue but must be confirmed. Additionally, there is always a risk of bias in DL models and datasets, thus, making it important to ensure the algorithms and data do not perpetuate bias and/or discriminate against certain groups, individuals or networks. An example of this would be data that only contains one form of malware or algorithms trained on data specific to only one network type.

Any negative consequences of the project should be taken responsibility for. This includes but is not limited to false positives or negatives, missed malware detection and incorrect malware detection. Measures should be put in place to mitigate these risks.

5 Related Work

Zeng et al. [20] present a framework for a deep learning-based network encrypted traffic classification and intrusion detection framework. In the paper, three deep-learning algorithms are tested and employed. These are namely, a Convolutional Neural Network (CNN), a Long Short-Term Memory (LSTM) and a Stacked Auto-Encoder (SAE). The authors also address storage and computational expense factors by providing a lightweight framework. The paper accurately classifies encrypted traffic, with the CNN obtaining an accuracy of 99.85%. While LSTM obtains an accuracy of 99.41% for intrusion detection. The paper does not specifically target ransomware and hence may come short when detecting the vast amounts of ransomware families.

Modi et al. [10] performed ransomware detection using machine learning on encrypted network traffic. The authors mention the lack of datasets containing encrypted ransomware network traffic and hence solve this by collecting a dataset that was made public [16]. Three machine learning classifiers were used, support vector machine (SVM), random forest and logistic regression. In the results, it is shown that the random forest performs best, obtaining an accuracy of 99.9% and a false-positive rate of 0%. These results are exceptional, however, these machine-learning approaches are computationally intensive and require large storage, making them less ideal for resource constrained nodes.

Papadogiannaki et al. [14], recognizing the difficulty NIDS have when handling encrypted traffic, propose an approach which uses signatures based on payload sizes. This approach attempts to generate a comprehensive language of malicious traffic signatures, where a signature is defined as a sequence of consecutive payload sizes; network flows are inspected to see if they contain these signatures [14]. For example, Papadogiannaki et al. [14] recognized that should a network flow contained a sequence of four packets of respective sizes 22, 976, 48, 16, an intrusion attempt of a Hydra password cracking tool has taken place. An important aspect of this approach is the size of the signature - or, how many payload sizes a signature refers to. If a signature is too short, it is likely to result in false positives; false positives are particularly undesirable for IDS since they undermine the confidence of the system, which can lead to ignoring real intrusion events.

A common approach to traffic classification and network intrusion is payload inspection. This involves inspecting the payload of each packet. Previously, methods involve using regular expressions as signatures for the different protocols. These methods struggle when the regular expressions need to be updated for each new protocol released. Moreover, these methods become less efficient, or unusable when dealing with encrypted traffic. Sherry et al. [17] proposed a system to perform deep packet inspection with encrypted data. Although the paper only deals with the HTTPS protocol and does not use deep learning, the importance of working with encrypted traffic is highlighted.

6 Anticipated Outcomes

6.1 DL Models

By the end of this project, three models will be trained and tested. These models will be a combined SAE and CNN (as one model), an LSTM and an MLP. The MLP will be used as an ML benchmark test against the DL models. The models will be able to detect ransomware and worms from both encrypted and unencrypted traffic. Their performance (accuracy) and computational efficiency will be tested to conclude how effective these models are.

6.2 Expected Impact

This section will discuss the expected impact the project will have on research by applying DL models to ransomware and worm detection with encrypted network traffic.

6.2.1 Expected Results. Research on malware detection is abundant, but little focuses on detecting ransomware and worms. Our goal is to achieve high accuracy and fast detection times for these types of malware by using optimal DL models with suitable hyperparameters. Similar research [20][10] achieved over 90% accuracy when detecting/classifying malware, and we aim for similar results. Our DL models should be robust to new strains and outperform traditional machine learning approaches in terms of speed, accuracy, and resource consumption.

6.2.2 Effects. This project can impact various stakeholders, including the cybersecurity community, businesses, and individuals affected by ransomware or worms. The project aims to minimize these risks and create opportunities for further work in ransomware and worm detection with encrypted network traffic. It also contributes to the ongoing development of DL methods for malware detection, with broader implications for DL in other cybersecurity applications and fields.

6.3 Key Success Factors

In order to ensure the success of the project, a set of criteria must be met. Primarily, an assessment of the suitability of deep learning (DL) as a means of detecting ransomware and worms with encrypted network traffic is required. The effectiveness of DL models must be evaluated based on various metrics, including but not limited to accuracy, computer resource consumption, detection time, false positives and negatives. The targeted benchmark for accuracy is a threshold of greater than 90% with minimal utilization of computational resources. To achieve the aforementioned objectives, the DL models will undergo rigorous testing. By conducting exhaustive experiments, the research questions should be answered with certainty.

7 Project Plan

7.1 Risks

During the course of the project, there are certain risks that we may face. Appendix A contains our risk assessment document where we highlight these risks. In the document, the risks are mentioned, their impact and probability of happening are shown, and lastly, ways of mitigating and managing these risks are recommended.

7.2 Timeline

The project timeline commences on the 22nd of February and ends on the 24th of October with a School of IT showcase. The complete timeline for the project is shown in a Gantt chart in Appendix A. The Gantt chart gives a breakdown of the tasks (both individual and group work) to be completed throughout the timeline of the project. The Gantt chart ensures that we start on our tasks early, allowing us ample time to get feedback and make necessary changes.

7.3 Resources Required

7.3.1 Software. For the software, we will be using Python 3 to implement our DL models. The Python libraries we will use are *Numpy, TensorFlow, pandas, PyTorch* and *Matplotlib. Scikit* will be used for hyperparameter training. The Python code will be implemented and executed on *Google Colab* to allow us access to more powerful GPUs, and hence faster training. Another essential piece of software are the datasets that we will use to train our models on. The datasets we will be using are the Open Access Dataset [2] discussed in Section 3.1, and a combination of datasets from the Stratosphere Laboratory [6].

7.3.2 Hardware. Most of the work required for this project will be completed on personal laptops. However, for the more resource-intensive tasks such as training and testing, Google Colab will be used for access to their more powerful GPUs. This will allow for faster training, testing and detecting times. Additionally, some training on the DL models may be performed on our supervisor's more powerful computer.

7.3.3 People. The people required to complete this project are the two authors of this proposal, Chris Lamprecht and Lucas Carr.

7.4 Milestones

Due Date	**Deliverable**
20 March	Literature Review Due
21 April	Full Project Proposal
25 April	Project Proposal Presentation
2 May	Revised Project Proposal
12 May	Ethics Application
17 July	Project Progress Demonstration
28 August	Complete Draft of final paper
11 September	Project Paper Final Submission
15 September	Project Code Final Submission
26 September	Final Project Demonstration
9 October	Poster Due
16 October	Website Due
24 October	School IT Showcase

7.5 Deliverables

Below is a list of the deliverables that will be accumulated throughout the course of the project.

- Two Literature Reviews discussing the background of the project and previous deep learning approaches to the detection of malware in encrypted network traffic.
- A Project Proposal
- A software feasibility demonstration
- A complete final paper that discusses and evaluated the deep learning approaches and their success towards detecting ransomware and worms within encrypted network traffic.
- A project poster highlighting the key aspects of our research objectives
- A web page that contains the details and deliverables of our project

7.6 Work Allocation

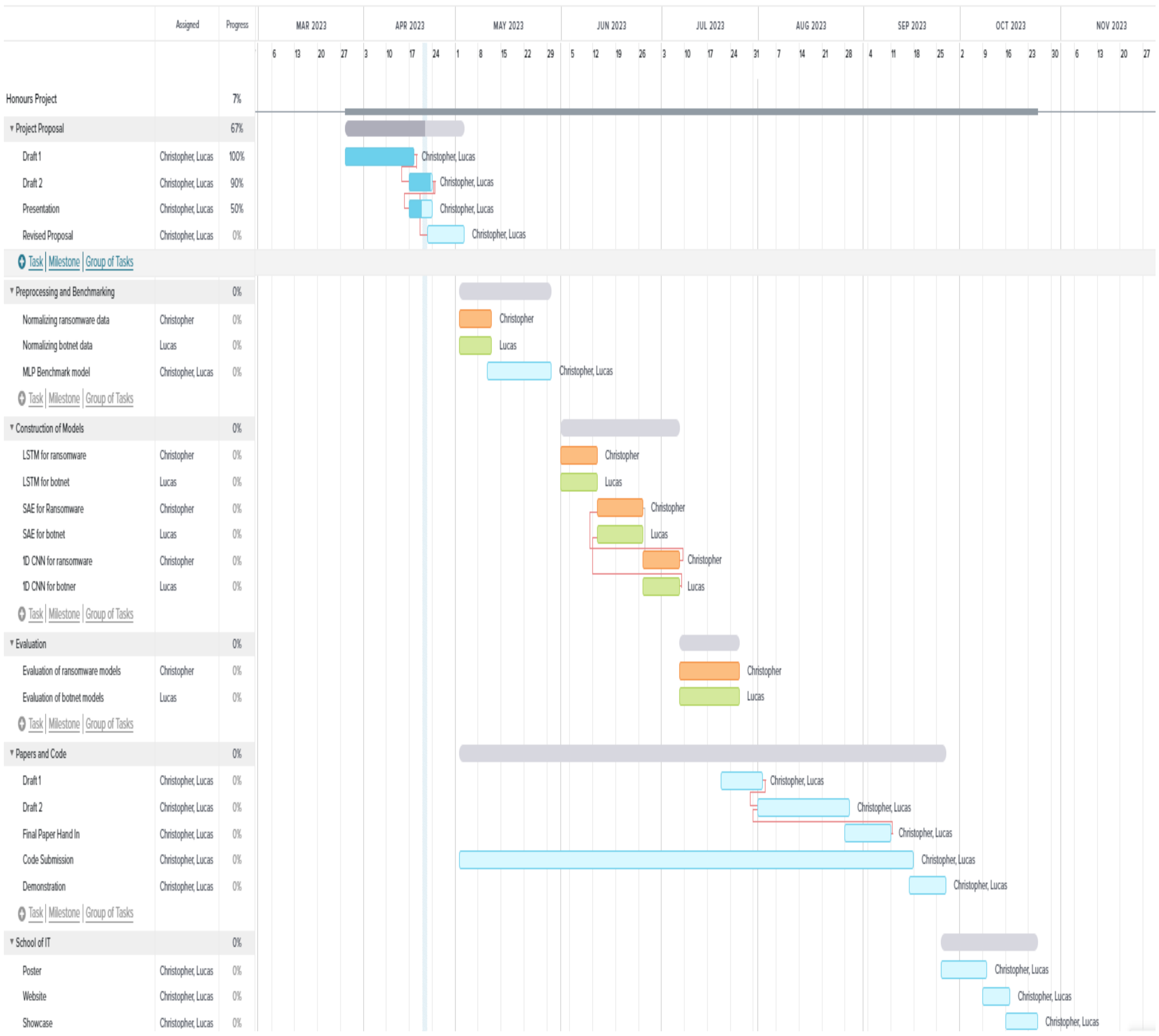
The project involves shared and individual work stages. Both group members will implement the MLP benchmark model, while Lucas and Chris will individually build, tune, test and evaluate DL models for detecting worms/botnets and ransomware, respectively. As each partner works with different datasets, they will handle their data pipeline, including data organization and labelling.

8 Appendix

8.1 Risk Matrix

Risk Description	**Impact (1-4)**	**Probability (1-5)**	**Mitigation**	**Management**
The project not being completed in the required time	4	2	Work daily on the project Be sure to meet all deadlines	Constant communication with partner and supervisor, to ensure work is being done
One partner not contributing their weight	3	2	Keep up to date with each other's work, and help the other partner get up to speed if they fall behind	*i.* Keep an updated to-do list *ii.* Follow the Gantt chart
Poor communication with partner	3	1	Maintain a good relationship with your partner, and adjust to each other's schedules	Have daily/semi-daily meetings to discuss what work has and is being done
Poor communication with the supervisor	4	2	*i.* Maintain a respectful relationship with the supervisor *ii.* Ensure that we listen to the supervisor's advice	Keep consistency with the weekly meetings and arrange additional meetings if needed.
Partner falls sick and is unable to do their work	2	2	*i.* Do not leave work until the last minute, so that if your partner does fall ill, they will have time to recover *ii.* Keep up to date with your partner's work so that you can fill in gaps	Keep up constant communication between partners to see how their well-being is.
We are unable to answer our research questions	4	1	*i.* Ensure that research questions are specific enough so that their goals are clear and reachable *ii.* Start working on the project early to increase the time needed to answer the questions	Track Progress
Our models do not perform as we expect them to	3	4	Ground the problem statement, and hypothesis in reality - this might be based off findings of related works	The case where our findings do not match our expectations is not necessarily a bad result; since these findings could provide useful insight into the difficulty of the problem

8.2 Gantt Chart



	Assigned	Progress
Honours Project		7%
▾ Project Proposal		67%
Draft 1	Christopher, Lucas	100%
Draft 2	Christopher, Lucas	90%
Presentation	Christopher, Lucas	50%
Revised Proposal	Christopher, Lucas	0%
Task | Milestone | Group of Tasks		
▾ Preprocessing and Benchmarking		0%
Normalizing ransomware data	Christopher	0%
Normalizing botnet data	Lucas	0%
MLP Benchmark model	Christopher, Lucas	0%
Task | Milestone | Group of Tasks		
▾ Construction of Models		0%
LSTM for ransomware	Christopher	0%
LSTM for botnet	Lucas	0%
SAE for Ransomware	Christopher	0%
SAE for botnet	Lucas	0%
1D CNN for ransomware	Christopher	0%
1D CNN for botner	Lucas	0%
Task | Milestone | Group of Tasks		
▾ Evaluation		0%
Evaluation of ransomware models	Christopher	0%
Evaluation of botnet models	Lucas	0%
Task | Milestone | Group of Tasks		
▾ Papers and Code		0%
Draft 1	Christopher, Lucas	0%
Draft 2	Christopher, Lucas	0%
Final Paper Hand In	Christopher, Lucas	0%
Code Submission	Christopher, Lucas	0%
Demonstration	Christopher, Lucas	0%
Task | Milestone | Group of Tasks		
▾ School of IT		0%
Poster	Christopher, Lucas	0%
Website	Christopher, Lucas	0%
Showcase	Christopher, Lucas	0%

References

[1] Abu Rajab, M., Zarfoss, J., Monrose, F., and Terzis, A. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (New York, NY, USA, 2006), IMC '06, Association for Computing Machinery, p. 41–52.

[2] Berrueta, E., Morato, D., Magaña, E., and Izal, M. Open repository for the evaluation of ransomware detection tools. *IEEE Access 8* (2020), 65658–65669.

[3] Berrueta, E., Morato, D., Magaña, E., and Izal, M. Crypto-ransomware detection using machine learning models in file-sharing network scenarios with encrypted traffic. *Expert Systems with Applications 209* (2022), 118299.

[4] Cheng, Q., Wu, C., Zhou, H., Kong, D., Zhang, D., Xing, J., and Ruan, W. Machine learning based malicious payload identification in software-defined networking. *Journal of Network and Computer Applications 192* (2021), 103186.

[5] Firdausi, I., Erwin, A., Nugroho, A. S., et al. Analysis of machine learning techniques used in behavior-based malware detection. In *2010 second international conference on advances in computing, control, and telecommunication technologies* (2010), IEEE, pp. 201–203.

[6] Garcia, S., Grill, M., Stiborek, J., and Zunino, A. An empirical comparison of botnet detection methods. *computers & security 45* (2014), 100–123.

[7] Gholamy, A., Kreinovich, V., and Kosheleva, O. Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation.

[8] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[9] Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., and Saberian, M. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing 24*, 3 (2020), 1999–2012.

[10] Modi, J., Traore, I., Ghaleb, A., Ganame, K., and Ahmed, S. Detecting ransomware in encrypted web traffic. In *Foundations and Practice of Security: 12th International Symposium, FPS 2019, Toulouse, France, November 5–7, 2019, Revised Selected Papers 12* (2020), Springer, pp. 345–353.

[11] Mott, G., Turner, S., Nurse, J. R., MacColl, J., Sullivan, J., Cartwright, A., and Cartwright, E. Between a rock and a hard (ening) place: Cyber insurance in the ransomware era. *Computers & Security 128* (2023), 103162.

[12] O'Shea, K., and Nash, R. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

[13] Pachhala, N., Jothilakshmi, S., and Battula, B. P. A comprehensive survey on identification of malware types and malware classification using machine learning techniques. In *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)* (2021), pp. 1207–1214.

[14] Papadogiannaki, E., Tsirantonakis, G., and Ioannidis, S. Network intrusion detection in encrypted traffic. In *2022 IEEE Conference on Dependable and Secure Computing (DSC)* (2022), pp. 1–8.

[15] Rezaei, S., and Liu, X. Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine 57*, 5 (2019), 76–81.

[16] Sherif Saad, Issa Traore, A. A. G. B. S. D. Z. W. L. J. F. P. H. Detecting p2p botnets through network behavior analysis and machine learning. 100013.

[17] Sherry, J., Lan, C., Popa, R. A., and Ratnasamy, S. Blindbox: Deep packet inspection over encrypted traffic. In *Proceedings of the 2015 ACM conference on special interest group on data communication* (2015), pp. 213–226.

[18] Wang, Z., Fok, K. W., and Thing, V. L. Machine learning for encrypted malicious traffic detection: Approaches, datasets and comparative study. *Computers & Security 113* (2022), 102542.

[19] Weisz, S., and Chavula, J. Deep learning traffic classification in resource-constrained community networks. 22.

[20] Zeng, Y., Gu, H., Wei, W., and Guo, Y. $deep-full-range$: a deep learning based network encrypted traffic classification and intrusion detection framework. *IEEE Access 7* (2019), 45182–45190.

[21] Zheng, W., Zhong, J., Zhang, Q., and Zhao, G. Mtt: an efficient model for encrypted network traffic classification using multi-task transformer. *Applied Intelligence 52*, 9 (2022), 10741–10756.