A RANSOMWARE TRIAGE APPROACH USING A TASK MEMORY BASED ON META-TRANSFER LEARNING FRAMEWORK

Jinting Zhu, Julian Jang-Jaccard, Ian Welch, Harith Al-Sahaf
Cybersecurity Lab, School of Engineering and Computer Science
Massey University, Victoria University of Wellington,
Auckland, New Zealand, Wellington, New Zealand

Seyit Camtepe
‡CSIRO Data61, Australia

ABSTRACT

To enhance the efficiency of incident response triage operations, it is not cost-effective to defend all systems equally in a complex cyber environment. Instead, prioritizing the defense of critical functionality and the most vulnerable systems is desirable. Threat intelligence is crucial for guiding SOC analysts’ focus toward specific system activity and provides the primary contextual foundation for interpreting security alerts. This paper explores novel approaches for improving incident response triage operations, including ransomware attacks and zero-day malware. This solution for rapid prioritization of different ransomware has been raised to formulate fast response plans to minimize socioeconomic damage from the massive growth of ransomware attacks in recent years; it can also be extended to other incident responses. To address this concern, we propose a ransomware triage approach that can rapidly classify and prioritize different ransomware classes. We utilize a pre-trained ResNet18 network based on Siamese Neural Network (SNN) to reduce the biases in weight and parameters. In addition, our approach uses the entropy features directly obtained from the binary ransomware files to improve feature representation, resilient to obfuscation noise, and computationally less expensive, which evaluation also shows that this classification part of our proposed approach achieves the accuracy exceeding and outperforms other similar classification performance. This new triage strategy based on Task memory with meta-learning evaluates the level of similarity matching across ransomware classes to identify any risky and unknown ransomware (e.g., zero-day attacks) so that a defense of those that support critical functionality can be conducted.

Keywords Incident response, Malware Classification, Ransomware Triage, Meta-transfer Learning, Few-shot Learning

1 Introduction

In security operations, threat intelligence plays a crucial role in directing SOC analysts towards specific system activity and provides the background context for interpreting security alerts. To optimize the use of human resources, lower-threat incidents should be assigned to lower-level analysts, while senior analysts should focus on more serious threats. Additionally, the automated resolution should be the preferred approach for most malicious attacks, with manual intervention reserved only for rare cases such as unknown malware. According to the report by the European Union Agency for Cybersecurity in 2022, mentioned that ransomware and attacks against availability rank the highest during the reporting period. Ransomware accounted for the highest threat. Moreover, ransomware refers to a type of malicious program designed to block access to a computer system until a sum of money is paid, hence the name ransomware. The emergence and rapid increase of ransomware, compared to other types of malware, has been due to bitcoin and encryption technology. The invention of bitcoin has provided an anonymous payment channel for criminals demanding ransomware. The wide use of strong encryption techniques in many applications also allowed the creation of malware where decryption is not possible without knowing the cryptographic key. A rapid increase in ransomware has been reported causing the loss of millions of dollars for businesses and individuals. Especially, resourceful threat

\[1\text{https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022}\]
actors have utilized 0-day exploits to achieve their operational and strategic goals. The more organizations increase the maturity of their defenses and cybersecurity programs, the more they increase the cost for adversaries, driving them to develop and/or buy 0-day exploits since defence in-depth strategies reduce the availability of exploitable vulnerabilities. However, statistic-based methods such as clustering [1], entropy analysis [2, 3], similarity analysis [4], information flow analysis [5], and examining manifest file [6] have been proposed to recognize malware quickly. However, the heavy reliance on manual analysis along with tools support became either too expensive or not possible due to the growing volume of ransomware attacks to millions within a very short period of time.

The proliferation of machine learning techniques has allowed the reduction of manual intervention and has offered more automation-based machine analysis to rapidly recognize different types of ransomware (and other malware) to reduce a significantly increasing loss of money and productivity [7]. Semi-automated approaches using random forest [11, 12] were proposed to detect malware rapidly. For example, a KNN-based [13] and decision tree-based [14] classifiers to rapidly triage a large number of malware samples have been proposed in recent years.

However, many hurdles remain in providing effective machine learning-based triage solutions. We use the term “classification” when we can clearly classify ransomware samples into specific known profiles of existing ransomware classes. Extending from the classification, we use the term “triage” to refer to the overall process of conducting an assessment not only classifying ransomware samples into known ransomware classes but also identifying new types of ransomware samples, for example, risky and unknown ransomware.

One big obstacle to the development of machine learning-based triage solutions is the availability of ransomware samples to train machine learning models to learn about the features involved in ransomware codes. Unfortunately, current machine learning techniques demand a very large dataset (e.g., hundreds and thousands) to train a machine learning technique, such as to recognize important/relevant features to detect different ransomware signatures or to find correlations across different features. The other obstacle is with feature representation when the quality of features feeding to machine learning models heavily decides the quality of outcomes (e.g., detection accuracy). In many cases, malware authors can still easily avoid detection by applying obfuscation techniques to change feature representations easily even after a machine learning model is trained with a set feature representation.

To address these issues, we propose a ransomware triage approach using a Siamese Neural Network (SNN). The main contributions of our proposed approach follow.

- Our work has developed a new meta-transfer learning framework that for incident response. This framework takes full advantage of external information and enables us to optimize the learning process, improving the overall efficiency of the triage system. Our work represents the first task-driven meta-transfer learning effort in cybersecurity and highlights the potential for this approach to enhance cybersecurity operations.
- We uses a first-order with the Resnet18 fine-tuning to write the support items to memory, simplifying and speeding up episode training without backpropagation.
- Using a weighted ratio based on similarity scores, our approach can triage the risky and unknown ransomware classes (i.e., possibly zero-day attack) that exhibit the features common for ransomware but do not exactly match known ransomware profiles. This risky and unknown ransomware can be prioritized for further analysis to formulate the right response strategies.
- We conduct extensive experiments on popular benchmark datasets for few-shot learning to demonstrate that our method can effectively leverage unlabeled data in few-shot learning and achieve competitive results.

The rest of the paper is organized as follows. Section 2 presents related works. Section 3 provides the details of the proposed approach. Section 4 describes the details of the experiments and discusses the results, discussion, and limitations of the proposed approach. Finally, Section 5 draws a conclusion and describes the planned future work.

2 Related work

2.1 Triage Approach

Triage aims to assess the severity of the malware based on impact on the victim. The literature offers a variety of methods to speed up the severity of classification. The aim is to automatically investigate malware samples and pass them to the proper channels (e.g., cybersecurity professionals) for further analysis. A semi-automatic malware analysis architecture proposed in [11] replaces multi-class classification with a group of one-class classifiers to decrease the runtime. A random forest classifier is used in [12] with static malware features. Static features can be extracted fast without running the malware code and help perform quick triage. However, a weakness random forest algorithm as a
classifier, this approach tends to overfit when there is malware noise (e.g., obfuscated code) intentionally introduced by malware authors [15].

BitShred [4] is a system developed for large-scale malware clustering and similarity analysis. BitShred uses feature hashing to reduce feature space dimensionality and uncovers family relations by investigating correlated features with Jaccard similarity. SigMal [13] extracts features from the malware executable headers. SigMal transforms these features into a digital image. Signatures are extracted from images and investigated with similarity measures using the KNN algorithm. However, the KNN algorithm is susceptible to class imbalance was pointed out by [16]. TriDroid [14] extracts coarse-grained features (e.g., permissions) to allocate an investigated android app into a three-class queue and uses fine-grained static features with three-class classifiers to confirm the queue assignments with high accuracy.

TRIFLOW [5] uses information flow in Android apps to characterize behaviors of the risky apps for triage purposes. DROIDSEARCH [6] is a search engine with triage capability that triages based upon information such as ratings, downloads, or information from the manifest file. Mobile Application Security Triage (MAST) [17] uses statistical methods, called Multiple Correspondence Analysis (MCA), to find correlations on qualitative data obtained from the market. Rather than inspecting malware itself, tweet messages are investigated in [18] using Deep Neural Networks (DNN) for malware severity classification.

In the previous malware triage work, the proposed models have not achieved either satisfactory accuracy or were extensively tested on different attack classes. This leaves the triage system vulnerable only to classifying known malware samples with a high false-positive rate but also unable to identify unknown samples. The significance of the triage system is to support the automated operation of the system with less human intervention or cyber security analysts to recognize the urgency of the malicious attack to formulate more informed response strategies. Therefore, it is essential for a triage system to be equipped not only accurately classify them, but in some special cases, it identifies risky unknown malware for further analysis.

2.2 Transfer-Learning based meta Learning

The training strategy based on Transfer-learning differs from the episodic training strategy in meta-learning. Instead, the conventional techniques are able to be applied in a pre-train model with a large amount of data from the base classes. And then, the pre-trained model is adapted to recognize the base classes and novel classes. [19, 20, 21, 22, 23]. Yuan et al. [19] proposed an offline adaptive learning algorithm that is able to be learned through the meta-training part and the fine-tuning part. The meta-training part aims to optimize the task procedure and fine-tuning part adjust the pre-trained parameters with the limited data in the new application task. Sun et al. [20] learned a base learner that could be adjusted to the new task with a few labeled samples. In order to improve the efficacy of performance, they further introduced the hard task meta-batch scheme as a learning curriculum. Soh et al. [23] exploit both external and internal instances to present a Meta-Transfer Learning for the zero-shot task.

3 Methodology

The overview of our proposed model is shown in Fig. 1. Our model consists of three phases, (1) feature preprocessing phase, (2) meta learning phase, and (3) triage phase, respectively. The main goal of the feature preprocessing phase is to obtain entropy features from the bytecode of ransomware and preprocess them to feed to the model of resnet 18 for fine-tuning. In the meta-learning phase, our model constructs the feature embedding space of the input by utilizing two identical ResNet18 networks. We load a pre-trained version of the ResNet18 network trained on more than a million images from the ImageNet database.2 This pre-trained ResNet18 is used in a meta-learning fashion where the weights and the model parameters are further adjusted for the entropy features. The triage phase provides a triage service that classifies different ransomware classes as well as identifies risky and unknown ransomware based on the weight ratio.

3.1 Problem Formulation

We proceed to introduce the problem setup and notations of meta-learning. In the context of meta-learning, which can be divided into few-shot learning or zero-shot learning. In this paper, we construct the few-shot tasks based on the episode training, which is defined as a N-way K-shot classification problem. In few-shot classification, a small support set S included of S = and query set Q = are associated with the Meta-training and Meta-testing phase. During the optimizing stage with loss function, a parameter θ is learned from episode training data. The number of episodic optimization equals to the number of updating times. During the testing stage, an optimized meta-learning dθ measure the correlation between unseen samples in the support S and query set Q.

2ImageNet. http://www.image-net.org
In a nutshell, MTL is a novel learning method that helps deep neural nets converge faster while reducing the probability to overfit when using few labeled training data only.

### 3.2 Feature Preprocessing Phase

Entropy is often used as the measure of changes in information content. More changes to the original information content produce higher entropy values while fewer changes to the original information are associated with lower entropy values. As discussed in [24, 25], using entropy values as feature representation has a number of advantages compared to using grayscale image features. The feature values associated with grayscale features can be easily fooled if malware authors apply certain obfuscation techniques. For instance, control flow obfuscation techniques [26, 27, 28] can easily alter the control flow path of a malware program. For example, it inserts a junk code or shuffles the order of function calls which does not affect the semantics of the original malware program. This simple technique can effectively cause the appearance of grayscale images different from each other. This can easily lead to misclassification results where a classification algorithm puts them into different malware families when in fact they belong to the same malware family.

In addition, the computation cost associated with grayscale image features is usually much higher due to the higher cost involved in processing texture features of grayscale images [24, 25]. To avoid these weaknesses associated with grayscale features, we instead use entropy features that are more resilient to changes in the malware binary code and reduce computation costs.

To construct entropy features, we first read a ransomware binary file as a stream of bytes. The bytes are made into multiple segments each of which is comprised of 200 bytes. We further count the frequency of unique byte values, in the range of pixel values between 0 and 255, followed by computing the entropy using Shannon’s formula as seen in the following Eq. (1).

\[
H(p) = - \sum_i p_i \log p_i
\]

where \( p_i \) is the probability of an occurrence of a byte value. The entropy obtains the minimum value of 0 when all the byte values in a binary file are the same while the maximum entropy value of 8 is obtained when all the byte values are different. The entropy values are then concatenated as a stream of values and are used to generate an image-based entropy graph. The entropy graph, an image of size 224*224, is fed as entropy features as input to our model.

The process of generating the entropy features from a ransomware binary file is depicted in Algorithm 3.
Algorithm 1: Generating Entropy Features

\textbf{Input} : \( f \): malware binary file; \( l \): segment length; \( n \): the number of files

\textbf{Output} : entropy pattern image \( m \)

\begin{algorithmic}[1]
\While {not reach to \( n \)}
\State 1. read \( l \) bytes from \( f \), and defined as a segment \( s_i \);
\State 2. \For {\( j = 0 \) to 255}
\State 2.1 compute the probability \( p_j \) of \( j \) appearing in \( s \) using the Shannon entropy function
\EndFor
\State 3. a entropy graph image \( m \) generated from whole segments in a binary file
\EndWhile
\end{algorithmic}

3.3 Resnet 18 with Fine Tuning for Feature Extraction

shows a typical parameter-level Fine-Tuning (FT) operation, which is in the meta optimization phase o At the first stage, we fine-tune Resnet 18 with the support set to adjust the feature embedding, and then it is prepared to be applied in backbone network to generate feature embedding for meta-learning phase (i.e., the entropy feature representing a ransomware byte code from the same family). The structure of ResNet18 network follows the paper except the fully connect layers, in which the first fully connected layer has 512 neurons and the last fully connected layer serves as the output layer and has 256 neurons. The last fully connected layer from each Resnet18 is combined together to create a single fully connected layer with 512 neurons to compute a loss function across the features processed by two ResNet18 networks. The inputs are feed by the entropy graph that is represented as a image form of 2-dimensional vectors of a fixed size \( 224 \times 224 \), the fine-tuned ResNet18 is further optimized to obtain updated weights and parameters. The four blocks of each ResNet18 are followed by two fully connected layers with a fattening layer in-between.

3.4 Meta-Transfer learning

The first-order for optimization follows the MAML algorithm [29] to shallow the gap from the distribution of unseen samples. When the fine-tuned model is obtained, we begin the training process for meta transfer-learning outlined with a set of weight optimized for the last Residual Network block. This make the last residual network block that can easily adopts weights to new examples, thus ensuring effective handling of novel inputs. Despite being trained solely on the malware source domain, the malimage dataset has been observed to facilitate the acquisition of transferable features during training, as noted in [20]. Moreover, transferring statistics is used in this paper to calibrate the distribution of a limited number of training samples, which can be challenging as the learned model is prone to overfitting due to the biased and limited number of samples. The specific training process is outlined in Algorithm 1.

To further optimizes the meta operations scaling and shifting through meta-batching training. Given a task \( T \), the loss of \( T \) (tr) is used to optimize the current base-learner (classifier) by gradient descent:

\begin{align}
\theta - \beta \tag{2}
\end{align}

3.5 Task Memory for Adapted Semantic Feature

We makes a external memory module that aims to story the previous experience from the fine-tuning stage with true label \( y_m \). Each of the columns represent as an average feature of one class from other data sets under the similar distribution. With memory columns given by, which could be queried by finding the top \( k \) neighbors between a feature and memory columns. Meanwhile, the feature in the support and query set could absorb the task specific information and therefore able to adopt the current task, as

\begin{align}
v_m = \sum_{j=1}^{n_y} a_j c_i^j \tag{3}
\end{align}

where \( c_i \) is the adaptive prototype for class \( i \) and \( a_j \) indicates the class weights on each cluster. so that a high similar score given by the relation are obtained of higher weights to the instance in the support set.

Our aim is to create a semantic embedding invariance \( c_i \) refer to the prototypical network [30] and the \( v_i \) can be constructed to a memory module called \( \mathbf{V}_c \). We define the entries in \( V \) as \( \{v_1, v_2, v_3, ..., v_k\} \), in which each entries \( v_i \)
Algorithm 2: Learning in Meta-Training Phase

**Input**: $X$: malware entropy;

**Output**: updated $\theta_i$

1. **Require**: Model parameter $\theta_N$ trained for base learner;
2. Support sample in task $T_i$;
3. Query batch in task $T_i$;
4. **while** not reach in episodes of $T$ **do**
5. Randomly sample task $T_i d(x_s, x_q) =$ Feature extractor based on SNN Resnet18
6. Distance feature
7. Optimize $\theta$ by Adam and cross-entropy loss $L_s(\cdot, d, f)$

is an average feature performance computed via all instances in one class of the support set,

$$v_i = \frac{1}{n} \sum_{j=1}^{n_i} x^j_i$$

where $c_i$ is prototype for calculating relation $r_i$ to the novel set and $x_i$ is an instance of total number $n_i$ in the feature embedding of base set $s$. Gao et al. [31] argued that a given instance may has a higher variance from the mean feature calculated by all of rest feature or some of them have been wrongly labeled. Moreover, the small number of sample tend to result in a large variance so that a straightly average performance may not estimate a truly distribution over the all instances. To enhance the ability of feature representation, we make use of the extent memory to compute the relation between the support and query samples, which correlation score $a_{j,i}$ could be computed based on softmax function.

$$Sim(g(x^j_s), g(v_i)) = \frac{g(x^j_s) \cdot g(v_i)}{g(x^j_s) g(v_i)}$$

where $g(\cdot)$ is a feature embedding generated by the linear layer, $\tanh$ denoted by $\sigma$ is selected as an activation function to produce results among $[-1,1]$ and $\langle \cdot, \cdot \rangle$ means that the inner production of the two feature vectors of instances from the support set and query set respectively. And then, we aggregate the sequentially arriving semantic feature $x^j_i$ regardless of whose gradient and then average all the instance embedding in the support set for each relation,

$$h_{s,q} = \tau v_m + (1 - \tau) f(x_e)$$

where $\tau$ is the hyper-parameter determined by cross validation.

To construct training episodes, a random selection of classes is made from the training set, followed by selecting a subset of examples within each class to form the support set.

$$p^*_j = \frac{exp(-d(h_{s_j}, h_{q_j}))}{exp(-d(\sum_{c \in V_c} exp(h_{s_i}, h_{q_i})))}$$

where $d(\cdot, \cdot)$ is the Euclidean distance function for a relation vectors according to the strategy of Snell et al. [30] and it also generate a distribution over the specific class with the cross entropy loss function. The SGD algorithm is utilized in the learning process to minimize the negative log-probability of the actual class $k$.

### 3.6 Triage Phase

Our triage approach is similar to that of one used in the hospital system. Like patients whose symptoms clearly match the known profiles of illnesses, any (test) ransomware samples whose features clearly match the known ransomware classes are classified immediately to formulate a rapid response strategy. In some patient diagnosis whose symptoms does not clearly match one known profile but several profiles of illnesses, they could be sent to a designated specialist.
Algorithm 3: Prediction in the Meta-Testing Phase

Input: $f$: malware binary file; $l$: segment length; $n$: the number of files
Output: entropy pattern image $m$

1. while not reach to $n$ do
2. 1. read $l$ bytes from $f$, and defined as a segment $s_i$;
3. 2. for $j = 0$ to $255$ do
4. 2.1 compute the probability $p_j$ of $j$ appearing in $s$ using the Shannon entropy function
5. 3. a entropy graph image $m$ generated from whole segments in a binary file

for further examinations. Our triage also recognizes this type of ransomware sample whose feature clearly demonstrates that it is ransomware but does not match a specific ransomware family. These cases are classified as risky and unknown ransomware so that further analysis is carried out rapidly to establish if this is a new variant (e.g., zero-day attack). Towards this approach, our model first uses similarity matching to search if a ransomware sample under the examination (i.e., test sample) exhibits features similar to any known ransomware classes. The level of similarity matching to different ransomware classes is computed as a weight ratio. By evaluating the weight ratio, we can triage a ransomware sample to if this can be classified as specific ransomware or should be classified as risky unknown ransomware.

3.6.1 Similarity Matching

In a general similarity searching approach, a query record is compared against a stored database of records. The main goal here is to retrieve a set of database records that are similar to the query record. For example, if there is a picture of a dog as a query record, a similarity search should give a list of pictures with dogs in them.

In the context of the triage approach, the database corresponds to a collection of vectors of ransomware, which is a collection of feature embedding trained for a pair of training samples from a ransomware class. Similarity searching in our context refers to searching for similar feature embeddings. We employ the FAISS algorithm, Facebook’s library for faster similarity searching even for very large datasets, to calculate the similarity based on cosine distance. The details of the cosine distance we use are:

In our model, we use the FAISS algorithm to return the top $K$ most similar feature embeddings of ransomware samples whose cosine distance between the feature embedding of a test sample and the feature embeddings of the training sample are minima. Here, a similarity score indicates a different relationship between two samples (i.e., a test sample and a class representative from the trained ransomware classes). The minimum value of 0 indicates two samples are completely dissimilar. The maximum value of 1 indicates two samples are completely similar. The value close to 0 indicates the characteristics of orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

Fig. illustrates how FAISS-based similarity matching works. Let’s assume that the input is a pair of positive samples from the test set in which a feature embedding is constructed. The feature embedding is fed to the FAISS algorithm as a feature vector. The FAISS searches through the collection of feature embeddings from the training set which computes similarity scores based on cosine distance. The top $K$ feature embeddings with the highest similarity score are selected. Different weights are assigned based on the similarity score.

3.6.2 Weight Ratio and Classification

Based on the top $K$ search results and their weights, a weight ratio is computed and the final triage/classification is done. Mathematically, weight $w_i$ to the $i$th nearest neighbor for $i = 1, ..., K$ and classifies the test sample $x$ as the class that is assigned the most weight $w_i$, as follows,

$$W_{nor} = \text{argmax} \sum_{i=1}^{K} w_i I\{y_i = g\}$$ (9)

where $w_i$ denotes that the weights and $I$ is the total weights in the same class.

We normalize and regularize the weight ratio in a style that satisfies a linear interpolation with the maximum entropy (LiME) objective by linearly combining each weight. This allows the combined weights to be better balanced and the test sample, $x$, is best approximated based on the training samples through the Equation 10. The LiME objective is

---

3https://github.com/facebookresearch/faiss/blob/main/INSTALL.md
described as follows:

\[
\min_d = \| \sum_{i=1}^{k} d_i x_i - f(t) \|_2^2 - \lambda \| \sum_{i=1}^{k} d_i \|_2^2 \\
\text{subject to } \sum_{i=1}^{k} d_i = 1, d_i \geq 0, i = 1, ..., k
\]  

where \( x_i \rightarrow \mathbf{R} \) is \( i \)th training sample, the \( d_i \) is the basis atoms for the feature vector, \( x_i \), and \( f(t) \rightarrow \mathbf{R} \) is the feature vector of test sample, \( t \), \( \lambda \) is a regularization parameter.

Fig. 3 illustrates how the weight ratios are computed to decide a triage/classification outcome. Let’s take a test sample \( t_1 \) whose top 5 search results were the 2 feature embeddings from the ransomware class3 and the 3 feature embeddings from the ransomware class1. The weights from the 2 feature embedding from the ransomware class3 were 5 (had the highest similarity score based on FAISS similarity matching) and 2 (had the 2nd lowest similarity score), respectively. This results in the total accumulated weight for the ransomware class3 = 7. Similarly, the weights from the 3 feature embedding from the ransomware class1 were 4, 3, and 1 which resulted in the total accumulated weight for the ransomware class1 = 8. The weight ratio is calculated by dividing the total accumulated weights for a class by the total weights of search results which results in a weight ratio of approximately 0.53 for class1 and 0.46 for class3. Let’s presume that the weight ratio rate we want to decide to classify a given sample is set at the threshold = 0.6. Neither the weight ratio of class1 nor class3 satisfies this threshold. In this case, our model will put the test sample in the “risky and unknown” class and the sample does not exhibit any clear features that can classify into a known ransomware class despite it exhibiting some common features from known ransomware classes. In contrast, take a test sample \( t_2 \) where the weight ratio for class1 is equal to (or greater than) the threshold. In this case, the \( t_2 \) is classified as belonging to the ransomware class1.

### 4 Experimental Results

This section describes the details of our experiments including the system environment, dataset, and performance metrics we used. We also discuss the experimental results with discussion.

#### 4.1 Experimental Environment

This study was carried out using a 3.6 GHz 8-core Intel Core i7 processor with 32 GB memory on Windows 10 operating system. The proposed approach is developed using Python programming language with several statistical and visualization packages such as Scikit-learn, Numpy, Pandas, and Pytorch. Table 1 summarizes the system configuration for our environment.
Table 1: System configuration

| Unit               | Description                  |
|--------------------|------------------------------|
| Processor          | 3.6 GHz 8-core Inter Core i7 |
| RAM                | 32 GB                        |
| GPU                | GeForce GTX 1080 Ti          |
| Operating System   | Windows 10                   |
| Packages           | Pytorch, Scikit-Learn, Numpy, Pandas |

4.2 Dataset

We created a dataset containing ransomware binaries from VirusShare. The dataset comprises a total number of 1,048 samples from 11 families/classes of ransomware, each of which consists of a varying number of samples as listed in Table 2.

Table 2: Details of the ransomware dataset

| Training and Testing | Instances | Ratio (%) |
|----------------------|-----------|-----------|
| Bitman               | 99        | 9.45      |
| Cerber               | 91        | 8.68      |
| Dalexis              | 9         | 0.86      |
| Gandcrab             | 100       | 9.54      |
| Locky                | 96        | 9.16      |
| Petya                | 6         | 0.57      |
| Teslacrypt           | 91        | 8.68      |
| Upatre               | 18        | 1.72      |
| Virlock              | 162       | 15.46     |
| Risk Pool Test       |           |           |
| Wannacry             | 178       | 16.98     |
| Zerber               | 198       | 18.89     |

The dataset is highly imbalanced as some classes, e.g., Petya and Dalexis, are largely outnumbered by the other classes, e.g., Virlock. We split the dataset into training and test datasets. The training contains 80% of the data from 9 classes while the test contains 20% of the data from all 11 classes. Two ransomware classes, Wannacry and Zerber, were never trained by our model but only used during the test phase to evaluate the triage capability of our model.

4VirusShare. https://virusshare.com/
Table 3: Details of the Malimage dataset

| Class name | Family     | Instances |
|------------|------------|-----------|
| Worm       | Allaple.L  | 1591      |
| Worm       | Allaple.A  | 2949      |
| Worm       | Yuner.A    | 800       |
| PWS        | Lolyda.AA 1| 213       |
| PWS        | Lolyda.AA 2| 184       |
| PWS        | Lolyda.AA 3| 123       |
| Trojan     | C2Lop.P    | 146       |
| Trojan     | C2Lop.gen!g| 200       |
| Dialer     | Instantaccess | 431  |
| TDownloader| Swizzot.gen!I| 132 |
| TDownloader| Swizzor.gen!E| 128     |
| Worm       | VB.A       | 408       |
| Rogue      | Fakerean   | 381       |
| Trojan     | Alueron.gen!J| 198   |
| Trojan     | Malex.gen!J| 136       |
| PWS        | Lolyda.AT  | 159       |
| Dialer     | Adialer.C  | 125       |
| TDownloader| Wintrim.BX | 97        |
| Dilaer     | Dialplatform.B| 177 |
| TDownloader| Dontovo.A  | 162       |
| TDownloader| Obfuscator.AD| 142     |
| Backdoor   | Agent.FYI  | 116       |
| Worm:AutoIT| Autorun.K    | 106      |
| Backdoor   | Rbot!gen   | 158       |
| Trojan     | Skintrim.N | 80        |

4.2.1 Performance Metrics

To evaluate the performance of our model, we used the balanced classification accuracy of F1 score, precision, and recall as performance metrics. Table 4 illustrates the confusion matrix.

Table 4: Confusion matrix.

| Actual | Predicted | True Positive (TP) | False Positive (FP) |
|--------|-----------|--------------------|---------------------|
| Class_{pos} | Class_{neg} | False Negative (FN) | True Negative (TN) |

True Positive (TP) indicates a positive class correctly classified as a positive class. True Negative (TN) indicates a negative class correctly classified as a negative class. False Negative (FN) indicates a positive class is incorrectly classified as a negative class. False Positive (FP) indicates a negative class incorrectly classified as a positive class.

Based on the aforementioned terms, the evaluation metrics are calculated as follows:

\[
TPR(\text{Recall}) = \frac{TP}{TP + FN} \tag{11}
\]

\[
Precision = \frac{TP}{TP + FP} \tag{12}
\]

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}
\]

\[
F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{14}
\]

\[
AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP} \tag{15}
\]
4.3 Hyperparameter Performance

Figure 4 shows the performance of different hyperparameter tuning. As seen in the performance of learning rate (a), the learning rate = 0.0001 shows the best performance achieving the highest accuracy of over 80%. There is a significant difference in accuracy when other learning rates were used. The learning rate = 0.00001 achieved less than 20% accuracy while the learning rate = 0.001 improved the accuracy up to 60% but less than the learning rate = 0.0001. It shows that the learning rate was a significant hyperparameter influencing the overall accuracy compared to the center loss function. In terms of the performance of batch size, the best accuracy was achieved when the batch size = 64 was used. Similar accuracy rates all above 80% were achieved by the batch size = 32 and 128. We also tested the influence of hyperparameter $\alpha$ that is used for computing center loss. Though $\alpha = 0.1$ provided a slightly better accuracy, all three $\alpha$ rates contributed to achieving a high accuracy of around 90%. As these results illustrated the learning rate had the highest sensitivity to the model performance when compared to the batch size and hyperparameter $\alpha$.

In summary, Table 5 illustrates the best hyperparameters we used for our proposed model.

![Figure 4: Hyperparameter Performance](image)

Table 5: Training parameters

| Hyperparameters | Values | Descriptions |
|-----------------|--------|--------------|
| Learning rate   | 0.0001 | Learning speed (within range 0.0 and 1.0) |
| Batch size      | 64     | No. of samples in one fwd/bwd pass |
| Epoch           | 50     | No. of one fwd/bwd pass of all samples |
| Center loss     | 0.1    | Training loss rate |

4.4 Convergence Performance

Figure 5 illustrate the convergence loss and accuracy for both training and validation datasets. In terms of convergence loss shown in (a), the training of our model is done after 10 epochs and stabilizes until it reaches 50 epochs in both training and validation datasets confirming the training of our model is done at this stage. In terms of accuracy shown in (b), a high training and validation accuracy between 80% and 95% is achieved after 10 epochs where similar accuracy rates maintain until 50 epochs.

4.5 Classification Performance

We first check our model’s classification performance - that is, how well our model can classify ransomware samples that match the known profiles of existing ransomware. The results are shown in Table 6. For this benchmark, we used 9 ransomware classes depicted in Table 2 except Wannacry and Zerber. We applied a manual data argumentation technique to increase the sample size for three imbalance classes, Dalexis, Petya, and Uptare, to have at least 30 samples each. We used 80% for training and 20% for testing. The existing methods were re-implemented using the entropy features based on our dataset to ensure the fairness of the evaluation. We ran the experiments on 10-fold validation - running ten times of different 80% and 20% split.

As it is shown, our approach exceeded existing approaches proposed for the classification-only based triage applications. The classification part of our triage application exceeds the performance of Decision Trees by 18.4%, SVM by 10.4%, KNN by 15.6%, and Random Forest by 10%. In all the metrics, our approach based on the classification outperformed other similar models while achieving accuracy of more than 88%.
Figure 5: Convergence Performance

Table 6: Comparison to Similar Classification Models

|                  | Our model | Decision Trees [14] | SVM [14] | KNN [13] | Random Forest [11] |
|------------------|-----------|---------------------|----------|----------|--------------------|
| Accuracy         | 88.6%     | 70.2%               | 78.2%    | 73.0%    | 78.6%              |
| F1-score         | 87.2%     | 69.4%               | 77.8%    | 71.9%    | 78.1%              |
| Precision        | 85.4%     | 69.9%               | 77.2%    | 73.8%    | 78.0%              |
| Recall           | 88.6%     | 70.2%               | 78.2%    | 73.0%    | 78.6%              |

Fig 6 illustrates the results based on AUC_ROC curve that shows the area under the curve for different ransomware classes in receiver operating characteristics (ROC).

The micro-average of all the classes is 0.98 and the macro-average of all the classes is 0.96. These high micro and macro averages indicate that our model has a good performance across the different classes. The highest value was observed for the “gradcrab” class with the perfect 100% (i.e., no false positive or false negative). Except for the “petya” class which recorded the lowest value at 0.85, all other classes reached above 91% values.

4.6 Triage Performance

Figure 7 shows the projection of feature embedding space using the t-SNE dimensionality reduction technique. As seen in (a), the features from 9 different ransomware families contained in the training dataset are all intertwined together at Epoch = 1 (i.e., before training) which makes it difficult for our triage system to classify them into different classes. However, by Epoch = 50 seen in (b), there are distinct clusters formed around different ransomware where the unknown ransomware could not be placed into the known classes via the classifier, we could utilize the similarity compared with other features of samples from other families and put the unknown ransomware into the risky pool, as the (c) shows that the features of unknown ransomware are distinguished from other families. In other words, this is able to classify
the unknown and known samples with high false-positive rates into the risky pool. Ransomware families with larger samples have bigger clusters while a small cluster is seen around the ransomware families with very small sample sizes (e.g., Dyslexia and Petya). This confirms that our proposed model is effective at training to distinguish different ransomware classes.

We further examined the sensitivity of the threshold used for classification and accuracy, which is shown in Table 7. This is based on the similarity score obtained through running the KNN algorithm used in the FAISS similarity search with k size = 20. Here the threshold is the value associated with the normalized and regulizied weight ratio.

| Threshold | Classified | Risk Pool | Accuracy |
|-----------|------------|-----------|----------|
| $W_{nor} = 0.50$ | 51.2% | 48.8% | 91.3% |
| $W_{nor} = 0.45$ | 55.8% | 44.2% | 89.1% |
| $W_{nor} = 0.40$ | 61.2% | 38.8% | 78.4% |
| $W_{nor} = 0.30$ | 85.1% | 14.9% | 65.5% |
We take the weight ratio with higher confidence into account. Concretely speaking, when we used the highest threshold \(= 0.50\), slightly more than half of the test samples \(= 51.2\%\), were fully classified into known ransomware classes while slightly less than half were classified as risky unknown ransomware. With such a high threshold rate, the accuracy rate was also very high because at this stage the similar score contributing to weight ratio calculation has to be very high (i.e., two samples being compared need to have a very high correlation). As expected, as the threshold value decreased, more test samples were classified into known ransomware classes as it require a lower level of similarity score computed between a test sample and one classified during training. The accuracy rate dropped orthogonally compared to the decrease of the threshold value. At the lowest threshold value \(= 0.30\), 85.1 \% of the test samples were classified into known malware classes while 14.9\% of the test samples went into the risky and unknown ransomware class. The lowest accuracy rate of 65.5\% was achieved at the lowest threshold rate.

4.7 Discussion and Limitations

In this study, we propose our model that can be used as a triage application based on the classification of ransomware. Instead of using image features, we utilize entropy features that are more robust against noises (e.g., changes made by obfuscation technique) and less computationally complex. We use pre-trained two ResNet18 networks in a meta-learning fashion to obtain accurate weights and other parameters when our training sample size is limited. Instead of a feature embedding created based on a single input, our model constructs a feature embedding based on two positive inputs from the twin ResNet18. This results in our feature embedding containing relevant features to detecting a certain ransomware class. Note that we only use positive inputs (i.e., two samples from the same ransomware family) to train our model to find more common features exhibited within the same ransomware family. This approach is different from other proposals where negative inputs (i.e., two samples from different ransomware families) are used. Our experimental results confirm that the use of positive inputs contributes to improving detection accuracy compared to the models using the mix of negative inputs. The triage part of our model utilizes weight ratio to compute a better classification result by taking into account the similarity scores produced by the feature vectors of test samples compared to the feature vectors of the whole training samples. This allows our triage system to classify known ransomware classes with high accuracy while being able to classify risky unknown ransomware. These risky unknown ransomware samples can be given a high priority for further analysis.

At the moment, our model simply classifies any samples that cannot be mapped into any known ransomware classes as the risky and unknown ransomware class as these samples exhibit the features of ransomware. However, it may require further deep analysis to investigate the true nature and severity of these samples (e.g., these may well be zero-day attack or maybe disguised through more complex obfuscation techniques based on the known ransomware classes).

We observed that there could be a potential bias when the feature vectors from unknown samples are made into feature embeddings trained with known samples. To reduce this bias, we may require further analysis of the nature of unknown samples before feeding them to our model.

Our model classifies any unseen test samples into the risky and unknown pool. However, it is possible that some of these samples may exhibit some features that appear in ransomware but it can be benign. Our model can be trained with some benign samples to understand the features associated with these types of samples to more clearly classify whether it is completely benign or risky ransomware.

By using entropy features, our model is more resilient to producing misclassification when obfuscated malware is included in training samples. Though the resilience against the control flow obfuscation technique has been evidenced in the influence of other types of obfuscation techniques requires further investigation.

5 Conclusion

We proposed a ransomware triage approach that can rapidly classify different ransomware classes even in the presence of unknown classes. Our Siamese Neural Network (SNN) based approach utilizes a pre-trained ResNet18 in a meta-learning fashion to generate more accurate feature embedding and overcomes the biases in weight and parameter calculation typically associated with a model trained with training samples. Instead of image features typically used as inputs to machine learning-based malware detection and triage applications, our approach use entropy features directly obtained from the ransomware binary files. Our evaluations confirm that the use of entropy features provide a better feature representation and contribute towards improving triage accuracy. The experimental results tested on various ransomware samples show a very high classification accuracy exceeding 88\%. In addition, we offer a new triage strategy that can recognize risky and unknown ransomware which exhibits the feature commonly seen in other known ransomware but exact matching profiles cannot be found. These types of ransomware can be easily prioritized.
for further analysis to formulate an appropriate response strategy faster before any significant damages emerge (e.g., the loss of ransom payments or reduced productivity).

We plan to extend our work to assign more sophisticated weights for the matching feature vectors for Top $- k$ nearest neighbors as discussed in [38, 39, 40] and also include benign samples to compare their features with existing ransomware samples.

Acknowledgment

This research is supported by the Cyber Security Research Programme—Artificial Intelligence for Automating Response to Threats from the Ministry of Business, Innovation, and Employment (MBIE) of New Zealand as a part of the Catalyst Strategy Funds under the grant number MAUX1912.

References

[1] Jinting Zhu, Julian Jang-Jaccard, Tong Liu, and Jukai Zhou. Joint spectral clustering based on optimal graph and feature selection. *Neural Processing Letters*, 53(1):257–273, 2021.

[2] Timothy McIntosh, Julian Jang-Jaccard, Paul Watters, and Teo Susnjak. The inadequacy of entropy-based ransomware detection. In *International Conference on Neural Information Processing*, pages 181–189. Springer, 2019.

[3] Timothy R McIntosh, Julian Jang-Jaccard, and Paul A Watters. Large scale behavioral analysis of ransomware attacks. In *International Conference on Neural Information Processing*, pages 217–229. Springer, 2018.

[4] Jiyong Jang, David Brumley, and Shobha Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 309–320, 2011.

[5] Omid Mirzaei, Guillermo Suarez-Tangil, Juan Tapiador, and Jose M de Fuentes. Triflow: Triaging android applications using speculative information flows. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 640–651, 2017.

[6] Siegfried Rasthofer, Steven Arzt, Max Kolhagen, Brian Pfretzschner, Stephan Huber, Eric Bodden, and Philipp Richter. Droidsearch: A tool for scaling android app triage to real-world app stores. In *2015 Science and Information Conference (SAI)*, pages 247–256. IEEE, 2015.

[7] Jinting Zhu, Julian Jang-Jaccard, and Paul A Watters. Multi-loss siamese neural network with batch normalization layer for malware detection. *IEEE Access*, 8:171542–171550, 2020.

[8] Yuanyuan Wei, Julian Jang-Jaccard, Fariza Sabrina, Amardeep Singh, Wen Xu, and Seyit Camtepe. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access*, 9:146810–146821, 2021.

[9] Wen Xu, Julian Jang-Jaccard, Amardeep Singh, Yuanyuan Wei, and Fariza Sabrina. Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. *IEEE Access*, 9:140136–140146, 2021.

[10] Tong Liu, Fariza Sabrina, Julian Jang-Jaccard, Wen Xu, and Yuanyuan Wei. Artificial intelligence-enabled ddos detection for blockchain-based smart transport systems. *Sensors*, 22(1):32, 2021.

[11] Giuseppe Laurenza, Riccardo Lazzeretti, and Luca Mazzotti. Malware triage for early identification of advanced persistent threat activities. *Digital Threats: Research and Practice*, 1(3):1–17, 2020.

[12] Giuseppe Laurenza, Leonardo Aniello, Riccardo Lazzeretti, and Roberto Baldoni. Malware triage based on static features and public apt reports. In Shlomi Dolev and Sachin Lodha, editors, *Cyber Security Cryptography and Machine Learning*, pages 288–305, Cham, 2017. Springer International Publishing.

[13] Dhilung Kirat, Lakshmanan Nataraj, Giovanni Vigna, and BS Manjunath. Sigmal: A static signal processing based malware triage. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 89–98, 2013.

[14] Abdelouahab Amiria, Abdelouahid Derhab, ElMouatez Billah Karbab, Omar Nouali, and Farrukh Aslam Khan. Tridroid: A triage and classification framework for fast detection of mobile threats in android markets. *Journal of Ambient Intelligence and Humanized Computing*, 12(2):1731–1755, 2021.

[15] Yasser Alsouda, Sabri Pllana, and Arianit Kurti. Iot-based urban noise identification using machine learning: performance of svm, knn, bagging, and random forest. In *Proceedings of the international conference on omni-layer intelligent systems*, pages 62–67, 2019.
[16] Yulia Ery Kurniawati, Adhistya Erna Permanasari, and Silmi Fauziati. Adaptive synthetic-nominal (adasyn-n) and adaptive synthetic-knn (adasyn-knn) for multiclass imbalance learning on laboratory test data. In 2018 4th International Conference on Science and Technology (ICST), pages 1–6. IEEE, 2018.

[17] Saurabh Chakradeo, Bradley Reaves, Patrick Traynor, and William Enck. Mast: Triage for market-scale mobile malware analysis. In Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, pages 13–24, 2013.

[18] R Vinayakumar, Mamoun Alazab, Alireza Jolfaei, KP Soman, and Prabaharan Poornachandran. Ransomware triage using deep learning: twitter as a case study. In 2019 Cybersecurity and Cyberforensics Conference (CCC), pages 67–73. IEEE, 2019.

[19] Yi Yuan, Gan Zheng, Kai-Kit Wong, Björn Ottersten, and Zhi-Quan Luo. Transfer learning and meta learning-based fast downlink beamforming adaptation. IEEE Transactions on Wireless Communications, 20(3):1742–1755, 2020.

[20] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 403–412, 2019.

[21] John Bronskill, Daniela Massiceti, Massimiliano Patacchiola, Katja Hofmann, Sebastian Nowozin, and Richard Turner. Memory efficient meta-learning with large images. Advances in Neural Information Processing Systems, 34:24327–24339, 2021.

[22] Qianru Sun, Yaoyao Liu, Zhaozheng Chen, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning through hard tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[23] Jae Woong Soh, Sunwoo Cho, and Nam Ik Cho. Meta-transfer learning for zero-shot super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3516–3525, 2020.

[24] Jinting Zhu, Julian Jang-Jaccard, Amardeep Singh, Ian Welch, Al-Sahaf Harith, and Seyit Camtepe. A few-shot meta-learning based siamese neural network using entropy features for ransomware classification. Computers & Security, page 102691, 2022.

[25] Jinting Zhu, Julian Jang-Jaccard, Amardeep Singh, Paul A Watters, and Seyit Camtepe. Task-aware meta learning-based siamese neural network for classifying obfuscated malware. arXiv preprint arXiv:2110.13409, 2021.

[26] Lantao Wang, Yun Li, Haitao Zhang, Qigu Han, and Lirong Chen. An efficient control-flow based obfuscator for micropython bytecode. In 2021 7th International Symposium on System and Software Reliability (ISSSR), pages 54–63. IEEE, 2021.

[27] Kimia Zamiri Azar, Hadi Mardani Kamali, Shervin Roshanisefat, Houman Homayoun, Christos P Sotiriou, and Avesta Sasan. Data flow obfuscation: A new paradigm for obfuscating circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29(4):643–656, 2021.

[28] Stanley Chow, Yuan Gu, Harold Johnson, and Vladimir A Zakharov. An approach to the obfuscation of control-flow of sequential computer programs. In International Conference on Information Security, pages 144–155. Springer, 2001.

[29] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In International conference on machine learning, pages 1126–1135. PMLR, 2017.

[30] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. Advances in neural information processing systems, 30, 2017.

[31] Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 6407–6414, 2019.

[32] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", pages 986–996. Springer, 2003.

[33] Yang Ma and Xujun Zhao. Pod: A parallel outlier detection algorithm using weighted knn. IEEE Access, 9:81765–81777, 2021.

[34] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi, and Rama Chellappa. Kernel dictionary learning. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2021–2024. IEEE, 2012.

[35] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15750–15758, 2021.
Running Title for Header

[36] Zhixue Wang, Chaoyong Peng, Yu Zhang, Nan Wang, and Lin Luo. Fully convolutional siamese networks based change detection for optical aerial images with focal contrastive loss. *Neurocomputing*, 457:155–167, 2021.

[37] Huikai Shao, Dexing Zhong, Xuefeng Du, Shaoyi Du, and Raymond NJ Veldhuis. Few-shot learning for palmprint recognition via meta-siamese network. *IEEE Transactions on Instrumentation and Measurement*, 70:1–12, 2021.

[38] Matthias Dorfer, Jan Schlüter, Andreu Vall, Filip Korzeniowski, and Gerhard Widmer. End-to-end cross-modality retrieval with cca projections and pairwise ranking loss. *International Journal of Multimedia Information Retrieval*, 7(2):117–128, 2018.

[39] Yongjun Li, Xiujuan Chai, and Xilin Chen. End-to-end learning for action quality assessment. In *Pacific Rim Conference on Multimedia*, pages 125–134. Springer, 2018.

[40] Andrew Yates, Siddhant Arora, Xinyu Zhang, Wei Yang, Kevin Martin Jose, and Jimmy Lin. Capreolus: A toolkit for end-to-end neural ad hoc retrieval. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 861–864, 2020.

References

[1] Jinting Zhu, Julian Jang-Jaccard, Tong Liu, and Jukai Zhou. Joint spectral clustering based on optimal graph and feature selection. *Neural Processing Letters*, 53(1):257–273, 2021.

[2] Timothy McIntosh, Julian Jang-Jaccard, Paul Watters, and Teo Susnjak. The inadequacy of entropy-based ransomware detection. In *International Conference on Neural Information Processing*, pages 181–189. Springer, 2019.

[3] Timothy R McIntosh, Julian Jang-Jaccard, and Paul A Watters. Large scale behavioral analysis of ransomware attacks. In *International Conference on Neural Information Processing*, pages 217–229. Springer, 2018.

[4] Jiyong Jang, David Brumley, and Shobha Venkataraman. Bitshred: feature hashing malware for scalable triage and semantic analysis. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 309–320, 2011.

[5] Omid Mirzaei, Guillermo Suarez-Tangil, Juan Tapiador, and Jose M de Fuentes. Triflow: Triaging android applications using speculative information flows. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 640–651, 2017.

[6] Siegfried Rasthofer, Steven Arzt, Max Kolhagen, Brian Pfretzschner, Stephan Huber, Eric Bodden, and Philipp Richter. Droidsearch: A tool for scaling android app triage to real-world app stores. In *2015 Science and Information Conference (SAI)*, pages 247–256. IEEE, 2015.

[7] Jinting Zhu, Julian Jang-Jaccard, and Paul A Watters. Multi-loss siamese neural network with batch normalization layer for malware detection. *IEEE Access*, 8:171542–171550, 2020.

[8] Yuanyuan Wei, Julian Jang-Jaccard, Fariza Sabrina, Amardeep Singh, Wen Xu, and Seyit Camtepe. Ae-mlp: A hybrid deep learning approach for ddos detection and classification. *IEEE Access*, 9:146810–146821, 2021.

[9] Wen Xu, Julian Jang-Jaccard, Amardeep Singh, Yuanyuan Wei, and Fariza Sabrina. Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. *IEEE Access*, 9:140136–140146, 2021.

[10] Tong Liu, Fariza Sabrina, Julian Jang-Jaccard, Wen Xu, and Yuanyuan Wei. Artificial intelligence-enabled ddos detection for blockchain-based smart transport systems. *Sensors*, 22(1):32, 2021.

[11] Giuseppe Laurenza, Riccardo Lazzерetti, and Luca Mazzotti. Malware triage for early identification of advanced persistent threat activities. *Digital Threats: Research and Practice*, 1(3):1–17, 2020.

[12] Giuseppe Laurenza, Leonardo Aniello, Riccardo Lazzereetti, and Roberto Baldoni. Malware triage based on static features and public apt reports. In Shlomi Dolev and Sachin Lodha, editors, *Cyber Security Cryptography and Machine Learning*, pages 288–305, Cham, 2017. Springer International Publishing.

[13] Dhilung Kirat, Lakshmanan Nataraj, Giovanni Vigna, and BS Manjunath. Sigmal: A static signal processing based malware triage. In *Proceedings of the 29th Annual Computer Security Applications Conference*, pages 89–98, 2013.

[14] Abdelouahab Amira, Abdelouahid Derhab, ElMouatez Billah Karbab, Omar Nouali, and Farrukh Aslam Khan. Tridroid: A triage and classification framework for fast detection of mobile threats in android markets. *Journal of Ambient Intelligence and Humanized Computing*, 12(2):1731–1755, 2021.

[15] Yasser Alsouda, Sabri Pllana, and Arianit Kurti. Iot-based urban noise identification using machine learning: performance of svm, knn, bagging, and random forest. In *Proceedings of the international conference on omni-layer intelligent systems*, pages 62–67, 2019.
[16] Yulia Ery Kurniawati, Adhistya Erna Permanasari, and Silmi Fauziati. Adaptive synthetic-nominal (adasyn-n) and adaptive synthetic-knn (adasyn-knn) for multiclass imbalance learning on laboratory test data. In 2018 4th International Conference on Science and Technology (ICST), pages 1–6. IEEE, 2018.

[17] Saurabh Chakradeo, Bradley Reaves, Patrick Traynor, and William Enck. Mast: Triage for market-scale mobile malware analysis. In Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks, pages 13–24, 2013.

[18] R Vinayakumar, Mamoun Alazab, Alireza Jolfaei, KP Soman, and Prabaharan Poornachandran. Ransomware triage using deep learning: twitter as a case study. In 2019 Cybersecurity and Cyberforensics Conference (CCC), pages 67–73. IEEE, 2019.

[19] Yi Yuan, Gan Zheng, Kai-Kit Wong, Björn Ottersten, and Zhi-Quan Luo. Transfer learning and meta learning-based fast downlink beamforming adaptation. IEEE Transactions on Wireless Communications, 20(3):1742–1755, 2020.

[20] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 403–412, 2019.

[21] John Bronskill, Daniela Massiceti, Massimiliano Patacchiola, Katja Hofmann, Sebastian Nowozin, and Richard Turner. Memory efficient meta-learning with large images. Advances in Neural Information Processing Systems, 34:24327–24339, 2021.

[22] Qianru Sun, Yaoyao Liu, Zhaozheng Chen, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning through hard tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[23] Jae Woong Soh, Sunwoo Cho, and Nam Ik Cho. Meta-transfer learning for zero-shot super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3516–3525, 2020.

[24] Jinting Zhu, Julian Jang-Jaccard, Amardeep Singh, Ian Welch, Al-Sahaf Harith, and Seyit Camtepe. A few-shot meta-learning based siamese neural network using entropy features for ransomware classification. Computers & Security, page 102691, 2022.

[25] Jinting Zhu, Julian Jang-Jaccard, Amardeep Singh, Paul A Watters, and Seyit Camtepe. Task-aware meta learning-based siamese neural network for classifying obfuscated malware. arXiv preprint arXiv:2110.13409, 2021.

[26] Lantao Wang, Yun Li, Haitao Zhang, Qigu Han, and Lirong Chen. An efficient control-flow based obfuscator for micropython bytecode. In 2021 7th International Symposium on System and Software Reliability (ISSSR), pages 54–63. IEEE, 2021.

[27] Kimia Zamiri Azar, Hadi Mardani Kamali, Shervin Roshanisefat, Houman Homayoun, Christos P Sotiriou, and Avesta Sasan. Data flow obfuscation: A new paradigm for obfuscating circuits. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 29(4):643–656, 2021.

[28] Stanley Chow, Yuan Gu, Harold Johnson, and Vladimir A Zakharov. An approach to the obfuscation of control-flow of sequential computer programs. In International Conference on Information Security, pages 144–155. Springer, 2001.

[29] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In International conference on machine learning, pages 1126–1135. PMLR, 2017.

[30] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. Advances in neural information processing systems, 30, 2017.

[31] Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 6407–6414, 2019.

[32] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pages 986–996. Springer, 2003.

[33] Yang Ma and Xujun Zhao. Pod: A parallel outlier detection algorithm using weighted knn. IEEE Access, 9:81765–81777, 2021.

[34] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi, and Rama Chellappa. Kernel dictionary learning. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2021–2024. IEEE, 2012.

[35] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15750–15758, 2021.
[36] Zhixue Wang, Chaoyong Peng, Yu Zhang, Nan Wang, and Lin Luo. Fully convolutional siamese networks based change detection for optical aerial images with focal contrastive loss. *Neurocomputing*, 457:155–167, 2021.

[37] Huikai Shao, Dexing Zhong, Xuefeng Du, Shaoyi Du, and Raymond NJ Veldhuis. Few-shot learning for palmprint recognition via meta-siamese network. *IEEE Transactions on Instrumentation and Measurement*, 70:1–12, 2021.

[38] Matthias Dorfer, Jan Schlüter, Andreu Vall, Filip Korzeniowski, and Gerhard Widmer. End-to-end cross-modality retrieval with cca projections and pairwise ranking loss. *International Journal of Multimedia Information Retrieval*, 7(2):117–128, 2018.

[39] Yongjun Li, Xiujuan Chai, and Xilin Chen. End-to-end learning for action quality assessment. In *Pacific Rim Conference on Multimedia*, pages 125–134. Springer, 2018.

[40] Andrew Yates, Siddhant Arora, Xinyu Zhang, Wei Yang, Kevin Martin Jose, and Jimmy Lin. Capreolus: A toolkit for end-to-end neural ad hoc retrieval. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 861–864, 2020.