Cyber Threat Attribution with Multi-View heuristic analysis

Dilip Sahoo

School of Computer Science, University of Guelph, Ontario, Canada, e-mail: dsahoo@uoguelph.ca

Abstract: Over the years, a lot of malware variants have emerged and many of them are known to have originated from different Advanced Persistent Threats (APTs). The APT groups are the nation-state actors or well-resourced groups that target to compromise and exploit public or private organizations as well as individuals. If the source of the malware can be identified at an early stage, then it will significantly help the cybersecurity specialists to know with what they are dealing with and in making decisions about the best approach to remediate. APT groups can be attributed to their attack campaigns by observing their methods and Tactics, Techniques, and Procedures (TTP). A heuristic analysis of malware by taking multiple characteristics of the malware files corresponding to Opcode sequence, Bytecode sequences, and headers can provide a better comprehension of the TTP used in the campaign. The Machine Learning based multi-view analysis can help attribute the malware to its source with higher accuracy. In this experiment, we have adopted a 12 Multiview approach like a recent work that implements a Fuzzy Consensus Clustering Model for threat attribution. The experiment is conducted with 3594 malware samples corresponding to12 different APT groups. We used five different Machine Learning classifiers i.e. SVM, Decision Tree, KNN, Deep Learning (MLP), and Fair Clustering to evaluate all the 12 views and got an overall accuracy of 99%.

Keywords: APTs, Threat Attribution, Machine Learning, Deep Learning, Malware, Multi-View

1 Introduction

In the context of cybersecurity, threat attribution is a fundamental step to find out who is behind an attack. Ascribing a group or agency to threat helps the security professionals to take appropriate countermeasures to protect the individuals and organizations. APTs are the most challenging which are on the rise for the security professionals to defend against [1]. The APT groups use specific TTP to target, penetrate, and exploit organizations. Because of the sophisticated nature of the attack strategies adopted by the APT actors, it is not easy to attribute them against an attack [2]. A report from McAfee claims that most APT attacks are interrelated in their nature of the attack and they have similar target organizations [3]. The general characteristics of APTs are that they are sophisticated, targeted, evasive, the attack adopts to security measures, and has multiple attack vectors.

The APT attacks are evasive that use several methods to stay undetected like using commonly accepted protocols for sending threat contents. They use custom encryption techniques to overcome firewall detection. It is common to notice various detection evading techniques and code obfuscation techniques used in many malware variants which creates confusion and may sidetrack the analysts while focusing on a specific characteristic(from a single-view) of the malware.

In most APT campaigns, it is noticed they launch very sophisticated attacks and target a particular type of organization. For example, the Stuxnet campaign was targeting centrifuges that use programmable logic controllers (PLCs) manufactured by Siemens. Stuxnet was using an extremely sophisticated worm that exploits zero-day vulnerabilities of windows systems [4]. These type of attacks can easily drop and install payloads in the target system easily as the attacks are designed for exclusively for the target systems. It is often impossible to detect such attacks by traditional Antivirus or Intrusion Detection Systems.

In the last decade, behavioral analysis of the malware files in a sandbox environment has become more popular. In this method, the researchers observe malware behaviors like network traffic, system calls, registry updates, etc at runtime by executing it in an isolated environment. This method is very effective against the unknown malware payloads which belong to APTs. However, this method is a time-consuming process and not useful in realtime detection scenarios. Also, sometimes the sophisticated malware programs can distinguish the sandbox environment from a real environment and behave differently.

On the other hand heuristic analysis, uses ML algorithms to train the systems with malware behavior. Such systems can be trained on the existing known malware file features like Opcode, Bytecode, Header details. The features can be fed to different Machine Learning and Deep Learning classifiers to perform the classification task. Heuristic analysis is effective against both unknown and metamorphic malware detection. A major pitfall of machine learning classifiers is that the output can get biased based on the training data. Sometimes training data may contain biases that can result in improper outcomes and impact the overall performance of the ML classifiers [4]. To address the issue of biased prediction by a classifier, it is important to feed the ML classifier with balanced data [5].

In our experiment, we used more than 3000 malware files from 12 APT families to conduct our research. Below are the measure research contributions made as part of this work.

I. We created 11 different views based on the extracted Opcode, Bytecode, and Header features to look at the files under observation from different aspects. This helps to make the system resilient against obfuscation and evasion techniques.

II. Five different machine learning algorithms namely SVM, Decision Tree, KNN, MLP, and Fair Clustering were evaluated with all the views.

III. A SMOTE data set was developed with balanced distributions of data samples to reduce bias in favor of any particular class.

IV. A Multi-View prediction approach was adopted by combining the individual predictions from the single-views based on majority class and accuracy(%).

Section 2 of this paper contains details on related work done recently for APT threat detection. Section 3 contains a brief description of the dataset used in the experiment. Section 4 details our experiment methodology which is followed by our Experimentation and Results in Section 5.

Section 6 highlights a comparison of our findings with related works. Section 7 presents Our concluding statements and avenue for future work.

2 Related Work

Knowledge of the threat source increases the confidence of the security professionals during incident triaging and later with the incident response phase. It also helps them to decide the next course of action in a time-efficient manner due to the additional supplements of information regarding the TTP used by the attacker groups. Due to the substantial benefits of threat source attribution, various approaches have been taken by researchers to effectively automate the process of cyber threat attribution.

The paper [6] combines several individually contributed papers and provides a basic understanding of APTs along with explanations, examples, and case studies on the APT phenomenon, their characteristics, APT attack stages, and how they should be handled. The paper discusses APT definitions considering the viewpoints and case study reports provided by various leading security organizations and government agencies. The papers suggest that any organization should consider APT threat seriously due to the targeted nature and suggest ways to efficiently protect against APT campaign attacks.

Several types of research have been conducted to detect and prevent APT attacks. Implementation of traffic data analysis is one of the most popular approaches suggested by several researchers [7]–[9]. Trafic data analysis is conducted by analysis of network protocols, carried operations, and data that flows through the network. Researchers in [8] suggested a combination of traffic data analysis with an open-source intrusion detection system that analyzes the protocols used, requests sent, and uses filtering using black-list.

Pattern recognition is another popular approach to detect and prevent and APT attacks. In this approach, malicious programs are considered to be similar and they are distinguished from the benign applications by tracing their operational similarities and differences. Authors in [10] suggest a single layer pattern recognition approach. It is also common that several methods are combined to create a system that can protect against the APT threat than the individual methods. Moon et al. [11] and Vert et al. [12] used a combination of pattern recognition and multilayer security for detection and protection against APT threats at different security layers.

Heuristic analysis using ML classifiers is becoming more popular than traditional detection methods in the recent decade. Authors in [13] present an interesting approach to detect malware Application. The proposed system uses the Application Programming Interfaces called by the malware program and technical PE features to classify malware files. It uses the chi-square (KHI^2) measure and the Phi (φ) coefficient for considering features by relevance. The system could accomplish binary classification with 98% accuracy in a time-efficient manner.

Authors in [14] implemented a Multi-View Fuzzy Consensus Clustering Model for Malware Threat Attribution. The suggested approach uses 12 views to attribute the malware from five APT class. It implements a fuzzy pattern tree, multi-modal fuzzy classifier, and consensus clustering technique to analyze the malware behavior. The suggested system could perform threat attribution with 95% accuracy.

3 Dataset

We used 3594 malware file samples [15] belonging to 12 different APT groups namely APT1, APT10, APT19, APT21, APT28, APT29, APT30, DarkHotel, EnergeticBear, EquationGroup, GorgonGroup, and Winnti for the experiment. These APTs were alleged to be sponsored by five different nation-states. Each of the malware files was processed using additional python scripts to extract details of the Opcode, Bytecode, and Header information and to create multiple views based on that information. Details of the data processing and view creation are discussed in later sections 4.1 and 4.2.

Sl no	APT Group name	No of Malware files	Nation-State
1	APT1	405	China
2	APT10	244	China
3	APT19	32	China
4	APT21	106	China
5	APT28	214	Russia
6	APT29	281	Russia
7	APT30	164	China
8	DarkHotel	273	NorthKorea
9	EnergeticBear	132	Russia
10	EquationGroup	395	USA
11	GorgonGroup	961	Pakistan
12	Winnti	387	China

Table 1 illustrates the number of malware samples collected against each APT group.

Sl no	APT Group name	No of Malware files	Nation-State
1	APT1	405	China
2	APT10	244	China
3	APT19	32	China
4	APT21	106	China
5	APT28	214	Russia
6	APT29	281	Russia
7	APT30	164	China
8	DarkHotel	273	NorthKorea
9	EnergeticBear	132	Russia
10	EquationGroup	395	USA
11	GorgonGroup	961	Pakistan
12	Winnti	387	China

Table 1 APT-Malware data description

4 Methodology

In this section, we describe the detailed steps taken during the experiment to implement a multi-view-based malware attribution model. The Malware files for 12 different APT groups were collected and processed to create the multi-view data samples. Later these multi-view data samples were used to attribute each malware file to an APT group. During the experiment, we trained five Machine learning classifiers and evaluated them in terms of accuracy. Finally, the best detection models for each view were identified and implemented for malware attribution. The proposed system consists of three important modules namely the pre-processing module, the view extraction module, and a threat attribution module.

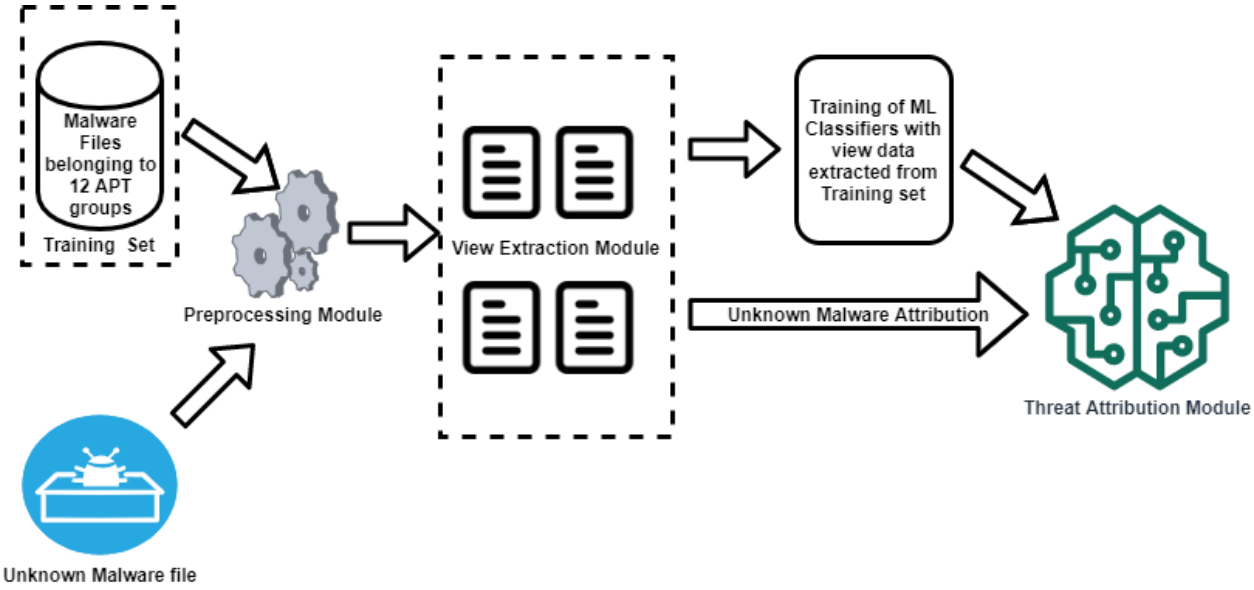

Figure 1 The Multi-view malware attribution system

4.1 Preprocessing and View Extraction

The raw malicious files were processed using custom python scripts to extract information on details of the Opcode, Bytecode, and Header from each of them. This extracted information is treated as the foundation of our multi-view approach to perform further heuristic analysis using ML classifiers. The processing and creation of each view are described in the below sections 4.1.1, 4.1.2,4.1.3, 4.1.4.

4.1.1 Opcode

Opcodes are the assembly instructions present in the malware executable files. To extract the opcode information from the malware executable files we used the Linux 'Objdump' command to disassemble the binary files. We then created a dictionary of all available unique opcodes which became the base for further processing of Opcode based views; we refer it as Opcode_Dic. Five different types of view samples were derived from the Opcode data extracted from the Malware binary files namely Binary, Count, Frequency, Term Frequency-Inverse Document Frequency (TFIDF), and Eigen Vector. For the creation of Binary, Count, Frequency, and TFIDF views we used the Text tokenization utility class using Keras [16]. For the creation of the Eigen Vector view, we followed the method proposed by Hashem et al. [17] to create the Eigen Vector view. ML classifiers were trained with all the Opcode based views and their performance in

terms of accuracy was noted against each view. This gave us a holistic comprehension of how different ML classifiers perform for different views. It is worth noting that all the Opcode based views were derived from the same Opcode information extracted from the malware files and by using different processing techniques.

4.1.1.1 Binary

This view is named as binary due to the nature of the data values present inside this view which is either '0' or '1'.The Opcode Binary view sample was created by checking whether a particular Opcode value of the Opcode_Dic is present in a malware binary file or not. We assign a value of '1' if it is present and '0' if it is not. The Opcode_Dic is considered as the base document to be referred for creating this view. Hence, the number of columns or features for this view remains constant for all the malware files. The number of columns is the same as the unique number of Opcodes present in the Opcode_Dic.

4.1.1.2 Count

In this view, the data values are represented as the count of each Opcode of the Opcode_Dic file, that is present in malware files. Hence, in contrast to the binary view, this view represents the actual count value of the Opcodes instead of '0' and '1'. The number of columns is the same as the unique number of Opcodes present in the Opcode_Dic.

4.1.1.3 Frequency

This view represents the frequency value of each Opcode from Opcode_Dic file as a ratio of the Opcodes present in the malware files. The number of columns is the same as the unique number of Opcodes present in the Opcode_Dic.

4.1.1.4 TFIDF

The TFIDF view represents the Term Frequency-Inverse Document Frequency score of each Opcode present in the malware files. The number of columns is the same as the unique number of Opcodes present in the Opcode_Dic.

4.1.1.5 Eigen Vector

The eigenvector view uses function call graph as the signature of the program proposed by authors in [18]. It uses the graph representation of the program and apply the mathematical equation to detect malware.

4.1.2 Bytecode

The Bytecode sequences of the malware files were extracted using custom python scripts. The bytecode values lie between 0 to 255. Hence, the bytecode dictionary file was created with all the values ranging from 0 to 255 and referred to as Bytecode_Dic. Five different sample views Binary, Count, Frequency, TFIDF, and Eigen Vector were created from the extracted bytecode values like the Opcode views discussed in section 4.1.1. Finally, the ML classifiers were evaluated for each of the Bytecode views.

4.1.3 Header

The header view represents the header information gathered from the malware Portable Executable (PE) files. The header fields were extracted from the PE file header and PE optional header sections using python libraries like: *'pefile'* [19] *and 'lief'* [20]. We have extracted field

information like ‘Machine’, ‘SizeOfOptionalHeader’, and ‘Characteristics’ from the PE file header section of the malware PE files. Similarly, from PE Optional header section, we extracted the fields: ‘MajorLinkerVersion’, ‘MinorLinkerVersion’, ‘SizeOfCode’ etc. Figure 2 shows the PE file header and PE Optional header information of a sample malware PE file. Due to the huge variance between the raw data collected from the header section, the raw data was later normalized using a logarithmic function. The normalized data were used to create the final Header view.

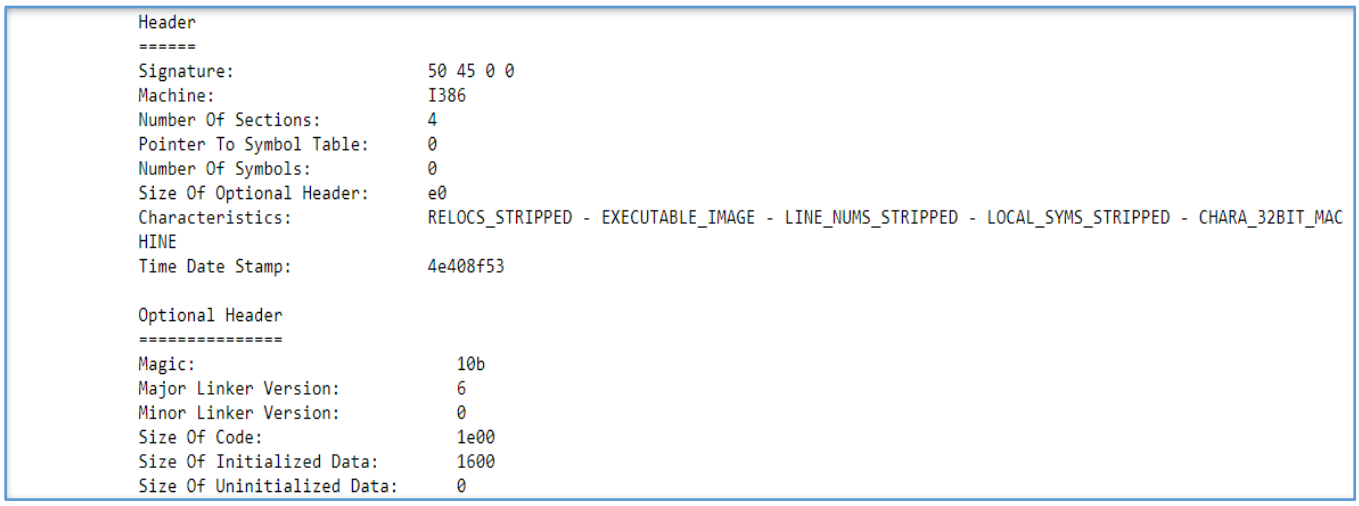

Figure 2: file header and Optional Header section fields of a malware PE file

4.2 Data balancing using Synthetic Minority Over-sampling Technique (SMOTE)

After the creation of the views, we observed that the data samples inside the views were not balanced. This is because certain APT groups had more malware samples than others. For example, there were only 32 malware samples belong to APT19 (shown in Table 1) which is considerably less compared to other APT groups. An imbalanced dataset can cause biased results and poor predictive performance, especially for the minority class. Hence, the imbalance dataset poses a challenge to the overall ML algorithm performance [21] [5]. To overcome the issues of the imbalanced data in the views, the SMOTE technique was used to balance the dataset by upsampling the minority class data. The SMOTE enhanced views were then used to train the ML classifiers.

4.3 Machine Learning classifier phase

In this phase, four well-known ML classifiers namely SVM, Decision Tree, KNN, and MLP were implemented from the open-source scikit-learn library (https://scikit-learn.org). Also, a Fair Clustering algorithm suggested by Backurs et al. [22] was adopted for the experiment. Each of the above-mentioned classifiers was evaluated with all the view samples’ data explained in section 4.1. The experiments were conducted in a 13 GB RAM Windows 10 virtual machine with 2.21 GHz 64-bit intel i7 processor and other 4 GB RAM. We used a 4GB Ubuntu 20.04 virtual machine for the extraction of Opcode information from the malware files. We used python 3.6.5 and MATLAB engine with jupyter notebook.

4.3.1 Support Vector Machine (SVM)

SVM is a simple algorithm that produces significant accuracy with less computational power [23]. The SVM algorithm finds hyperplane to classify N-dimensional data where 'N' is the number of features in the dataset. Due to the multiclass nature of our sample views' data, we chose the 'decision_function_shape' parameter value as 'one-vs-one (ovo)' which is a common approach followed during multi-class classification.

4.3.2 Decision Tree

A decision tree classifier can be used for both classification and regression tasks [24]. The feature importance and relations can be visualized clearly in a decision tree. It uses a greedy algorithm to lower costs.

4.3.3 K-Nearest Neighbour (KNN)

The KNN is an unsupervised ML algorithm that predicts the label of a new point from the testing sample by checking the label of 'K' predefined training samples which are closet in distance. The value of 'K' was kept as K=5 during our experiment which is the default value and gave us an optimum result.

4.3.4 Multi-layer Perceptron (MLP)

MLP is a type of neural network which is a deep-learning-based classifier. MLP is powerful because of multiple layers but can be a computationally expensive classifier. During our experiment, we used three hidden layers with 100 nodes each and the maximum number of iteration was set as 200 epochs.

4.3.5 Fair Clustering

A fair clustering approach was suggested by Backurs et al. [22] that provides fairness as well as scalability to the clustering algorithm and runs in near-linear time. Because of the additional benefits, we decided to implement this approach instead of the traditional k-median clustering algorithm. We used the elbow method to find an optimum cluster value 'k' which was set to k=20 during our experiment.

5 Experiments and Results

This section describes the details of the experiment conducted and highlights the results. Section 5.1 describes the evaluation measures adopted for the assessment. We experimented in two phases, in the first phase, described in section 5.2, the original view data obtained from the raw malware samples were used to train five Machine Learning classifiers(details of ML classifiers are mentioned in section 4.3) and the results were analyzed. After analyzing the results obtained from the first phase, we conducted a second phase of the experiment by feeding SMOTE enhanced balanced datasets to the four ML classifiers that performed best during the first phase. The details of the second phase experiment and its results are highlighted in section 5.3. Finally, section 5.4 demonstrates the results obtained using a multi-view prediction approach.

5.1 Evaluation measures

The threat attribution model in the experiment is a multi-class classification model where each threat actor is considered as a class. For instance, considering the confusion matrix we have

obtained during our experiment from the OPCODE_TFIDF view illustrated in Figure 3, there are 12 different classes denoting 12 APT groupsFigure 3. The multiclass classification model matrices can be understood as a set of several binary class classification models (where there are only two classes as 'Positive' or 'Negative'). For example, in our classifier, if we consider a malware file belong class 'APT1' then a True Positive occurs, when the malware is correctly predicted to be of class 'APT1'. Any other prediction will be considered to be a 'false negative'. In the multi-class classification, the positive and negative will depend on the true label of a sample and can change based on the object label. It means that for a given prediction, there will be multiple classes as 'true negative'. For instance, while considering class APT10, if a malware file that originally belongs to APT21 is predicted to be of any class (i.e APT1, APT19, APT21,…, Winnti) other than APT10, then it will be considered as true negative for the class APT10.

The evaluation measures for the experiment are derived from the confusion matrix. A common Confusion matrix represents the summary of all the predicted results of a classifier in terms of the number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). The diagonal elements in the confusion metrics represent the correctly classified samples for each APT. 'Accuracy' is the number of samples correctly identified as true positive or true negative out of all items. 'Precision' is the number of correctly identified positive samples out of all the positive predictions. 'Recall' also known as 'True positive rate' or 'Sensitivity rate' is the number of correctly predicted positive samples out of all the actual positives. F1-Score is the harmonic average of precision and recall and determines the effectiveness of the identification.

Predicted Label

True Label	APT1	APT10	APT19	APT21	APT28	APT29	APT30	DarkHotel	EnergeticBear	EquationGroup	GorgonGroup	Winnti
APT1	372	2	3	1	1	4	3	9	0	0	2	3
APT10	1	381	1	1	3	2	0	7	0	0	4	0
APT19	0	0	400	0	0	0	0	0	0	0	0	0
APT21	0	0	0	399	1	0	0	0	0	0	0	0
APT28	0	0	0	0	392	2	1	2	0	0	2	1
APT29	0	1	1	0	3	384	0	1	0	0	4	6
APT30	1	0	0	0	0	0	396	1	0	0	1	1
DarkHotel	1	7	0	1	1	5	5	371	0	0	4	5
EnergeticBear	0	0	0	0	1	0	0	0	399	0	0	0
EquationGroup	0	0	0	0	0	0	1	0	0	399	0	0
GorgonGroup	1	1	2	0	1	4	2	4	1	0	374	10
Winnti	4	1	2	0	2	3	0	3	0	1	8	376

Figure 3: OPCODE_TFIDF Confusion Matrix using MLP

TP: An APT actor with a true label as positive predicted correctly to be positive
TN: An APT actor with a true label as negative predicted correctly to be negative
FP: An APT actor with a true label as negative predicted incorrectly to be positive
FN: An APT actor with a true label as positive predicted incorrectly to be negative

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

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

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

F1-score = $2 * \left(\frac{Precision * Recall}{Precision + Recall}\right)$

5.1.1 Single-View Prediction Vs Multi-View Prediction

The classifiers(described in 4.3) evaluated with the individual views extracted during the preprocessing phase(described in section 4.1) referred to as 'Single-View prediction'. Single-View predictions include assessments from Opcode and Bytecode views (binary, count, frequency, and TFIDF) and header view. Optimization of the prediction results was done by leveraging multiple Single-View predictions. The high-level approach is to consider the prediction from the majority of the individual Single-View as the final Multi-View outcome. If there the majority APT actor cannot be decided between the Single-Views (When every Single-View predicted a different APT Class) then give weightage to individual Single-View predictions based on accuracy. The final Multi-View predictions will be determined by combining the individual predictions from the Single-View predictions. Figure 4 & Figure 5 illustrates scenarios of Multi-View prediction.

Multi-View Prediction:

Scenario-1

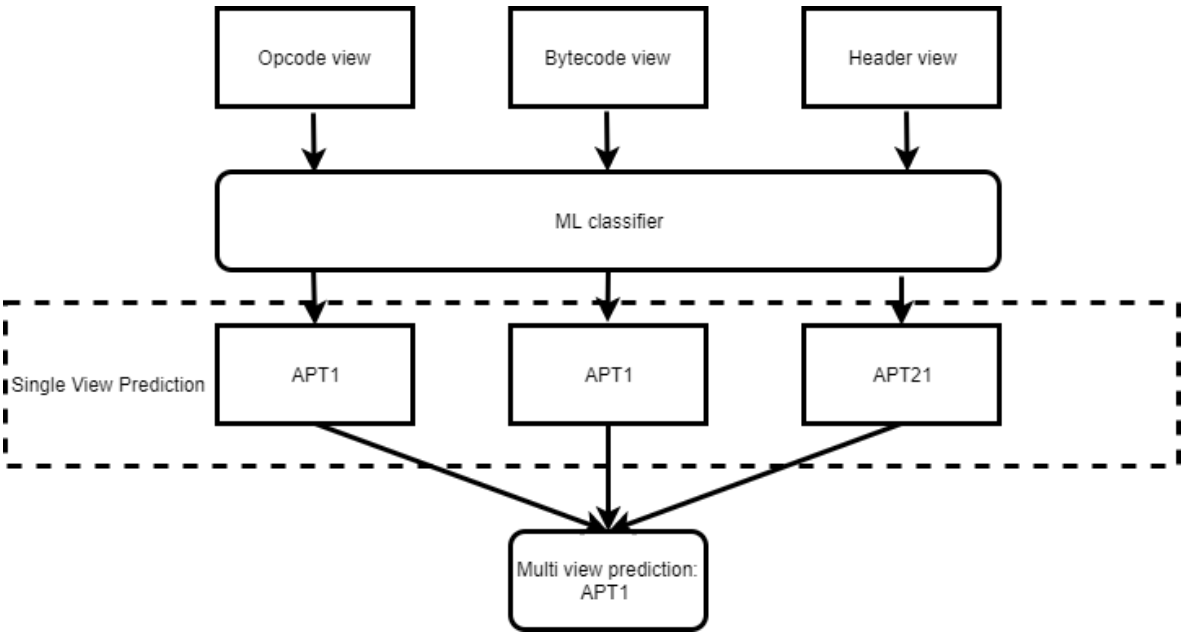

Figure 4: Multi-View Prediction using majority class predicted by individual Single-View

Scenario-2

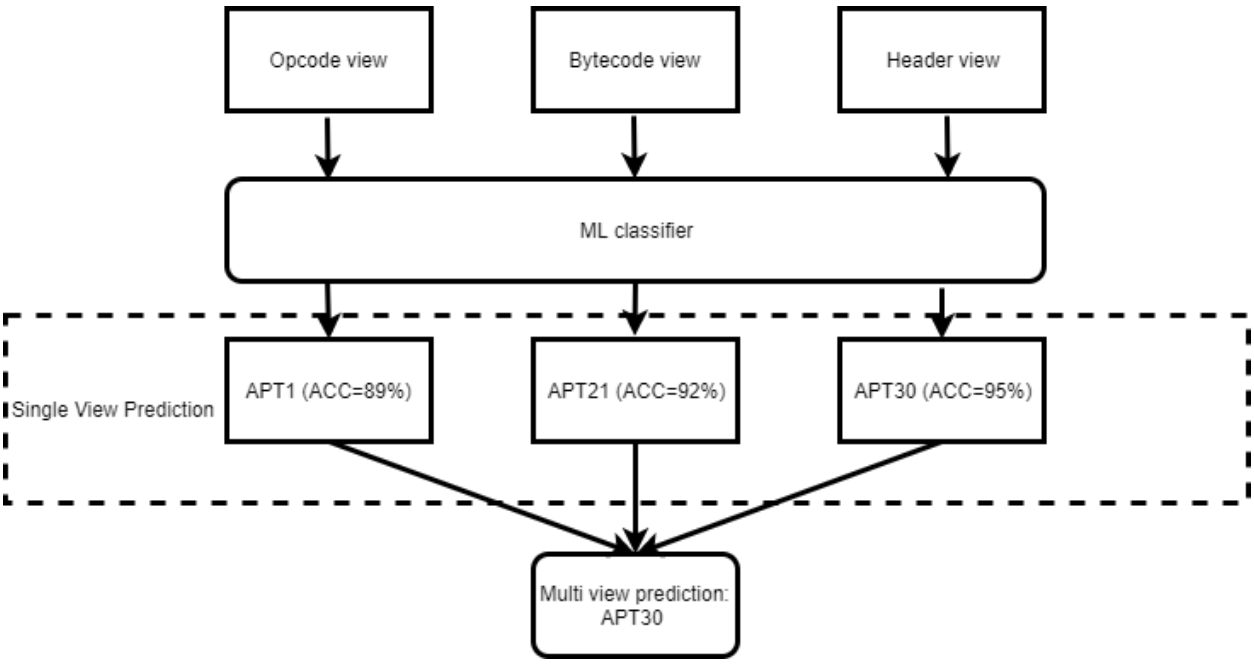

Figure 5: Multi-View Prediction using highest accuracy class predicted by individual Single-View

5.2 Experiment Phase-1 and Results

In this phase of the experiment, the individual Single-Views that belong to Opcode, Bytecode(binary, count, frequency, tfidf, Eigen Vector), and Header are evaluated with the classifiers individually. The data samples from original views were used to evaluate the classifiers at this phase using a 10 fold cross-validation technique. It was observed that OPCODE_TFIDF view and BYTECODE_FREQUENCY views gave the best results among other Opcode and Bytecode based views respectively. We also noticed that SVM, DT, KNN, and MLP classifiers outperformed the FAIR_CLUSTERING classifier in terms of accuracy. The overall accuracy results obtained using 10 fold cross-validation from the experiment phase-1 is summarized in Table 2.

View name	Accuracy(%)				
	SVM	DT	KNN	MLP	FAIR_CLUSTERING
OPCODE_BINARY	77	85	82	89	49
OPCODE_COUNT	53	89	81	86	64
OPCODE_FREQUENCY	75	88	82	89	58
OPCODE_TFIDF	73	89	85	91	57
OPCODE_EIGEN_VECTOR	51	67	52	68	28
BYTECODE_BINARY	30	31	22	31	24
BYTECODE_COUNT	58	82	81	81	47
BYTECODE_FREQUENCY	83	80	81	88	54
BYTECODE_TFIDF	72	82	82	88	49
BYTECODE_EIGEN_VECTOR	44	52	30	39	35
HEADER	82	89	86	91	62

Table 2: Summary of accuracy(%) from experiment phase-1

5.3 Experiment Phase-2 and Results

After analyzing the overall outcome of the experiment phase-1, the views that provided the best accuracy under each category (i.e Opcode, Bytecode, and Header) are selected for the experiment phase-2. To further optimize the performance of the classifiers, the data balancing of the selected views were done using SMOTE technique described in section 4.2. Finally, each view is evaluated with the top 4 classifiers that performed best in experiment phase-1. In this phase, a 5-10% improvement in accuracy was observed for each view for the four classifiers

under consideration. Table 3 illustrates the summary of accuracy improvement in the experiment phase-2.

View Name	SVM Accuracy(%)		DT Accuracy(%)		KNN Accuracy(%)		MLP Accuracy(%)	
	Original	SMOTE Enhanced	Original	SMOTE Enhanced	Original	SMOTE Enhanced	Original	SMOTE Enhanced
OPCODE_TFIDF	73	83	89	92	85	92	91	97
BYTECODE_FREQUENCY	83	93	80	92	81	95	88	98
HEADER	82	87	89	93	86	93	91	96

Table 3: Summary of improvement of accuracy in experiment phase-2

Clearly, the MLP classifier prediction results were best among other classifiers that are evaluated during the experiments. MLP is a powerful deep learning algorithm and hence it is important to analyze performance matrices like overall runtime and evaluation measures of individual APT class.

Table 4 illustrates a detailed APT class wise prediction result for the views along with runtime and overall accuracy.

View Name	Classifier	APT Name	precision	recall	f1-score	Run Time(Sec)	Overall Accuracy(%)
OPCODE_TFIDF	MLP	APT1	0.98	0.93	0.95	255.24	97
		APT10	0.97	0.95	0.96		
		APT19	0.98	1	0.99		
		APT21	0.99	1	1		
		APT28	0.97	0.98	0.97		
		APT29	0.95	0.96	0.96		
		APT30	0.97	0.99	0.98		
		DarkHotel	0.93	0.93	0.93		
		EnergeticBear	1	1	1		
		EquationGroup	1	1	1		
		GorgonGroup	0.94	0.94	0.94		
		Winnti	0.94	0.94	0.94		
BYTECODE_FREQUENCY	MLP	APT1	0.98	0.98	0.98	312.94	98
		APT10	0.96	0.97	0.97		
		APT19	0.99	1	0.99		
		APT21	0.99	1	0.99		
		APT28	0.95	0.99	0.97		

		APT29	0.97	0.97	0.97		
		APT30	0.99	0.99	0.99		
		DarkHotel	0.97	0.98	0.97		
		EnergeticBear	0.99	1	1		
		EquationGroup	1	1	1		
		GorgonGroup	0.97	0.88	0.92		
		Winnti	0.96	0.96	0.96		
VG_HEADER	MLP	APT1	0.93	0.92	0.92	160.1	96
		APT10	0.92	0.94	0.93		
		APT19	0.97	1	0.98		
		APT21	0.97	0.99	0.98		
		APT28	0.96	0.96	0.96		
		APT29	0.95	0.95	0.95		
		APT30	0.96	0.97	0.96		
		DarkHotel	0.92	0.9	0.91		
		EnergeticBear	0.99	0.98	0.99		
		EquationGroup	1	1	1		
		GorgonGroup	0.96	0.95	0.96		
		Winnti	0.95	0.93	0.94		

Table 4 Detail results from MLP classifier for the views of Opcode, Bytecode and Header category that performed best during the experiment

5.4 Multi-View Prediction

After having the best prediction results from individual single views during experiment phase-2, the multi-view prediction approach(as described in section 5.1.1) was adopted to further optimize the prediction performance. Table 5 shows the prediction performance obtained with the multi-view approach using a subset of the original data. The multi-view prediction provided an accuracy of 99% which is higher than the individual single-views observed in experiment phase-2.

APT1	400	0	0	0	0	0	0	0	0	0	0	0
APT10	0	227	0	0	0	0	0	1	0	0	0	0
APT19	0	0	31	0	0	0	0	0	0	0	0	0
APT21	0	0	0	86	0	0	0	0	0	0	0	0
APT28	0	0	0	0	162	0	0	0	0	0	0	0
APT29	0	0	0	0	0	254	0	0	0	0	0	0
APT30	0	0	0	0	0	0	159	0	0	0	0	0
DarkHotel	0	0	0	0	0	0	0	268	0	0	0	0
EnergeticBear	0	0	0	0	0	0	0	0	130	0	0	0
EquationGroup	0	0	0	0	0	0	0	0	0	395	0	0

	APT1	APT10	APT19	APT21	APT28	APT29	APT30	DarkHotel	EnergeticBear	EquationGroup	GorgonGroup	Winnti
GorgonGroup	0	0	0	0	0	0	0	0	0	0	278	0
Winnti	0	0	0	0	0	0	0	0	0	0	0	384

Figure 6: Multi-View Confusion matrix

View Name	Classifier	APT Name	precision	recall	f1-score	Overall Accuracy(%)
Multi_View	MLP	APT1	1	1	1	99
		APT10	0.995	1	0.997	
		APT19	1	1	1	
		APT21	1	1	1	
		APT28	1	1	1	
		APT29	1	1	1	
		APT30	1	1	1	
		DarkHotel	1	0.996	0.998	
		EnergeticBear	1	1	1	
		EquationGroup	1	1	1	
		GorgonGroup	1	1	1	
		Winnti	1	1	1	

Table 5: Overall performance Matrix of Multi-view

6 Results Comparison

This section, an efficiency evaluation of our system is done by comparing the results with some previously cited similar research work.

Table 6 shows the detailed comparison of results between our experiment and other similar work.

Method	Classifier	Classification Type	Accuracy
Our Method	MLP	Multi-Class(12 APT actor)	99%
Hamed et al.	Fuzzy Classifier	Multi-Class(4 APT actors)	95%

Mohamed et al.	BJ-48	Binary (Malware or Benign)	98%

Table 6: Result comparison from similar previously cited research work

Form the results shown in

Table 6, It can be seen that our experiment provides higher accuracy than the other two presented work. Our proposed system uses an MLP classifier which is a deep learning-based powerful algorithm that is more complex than the classifiers used in other systems and needs higher runtime.

The system proposed by Mohamed et al. provides 98% accuracy and takes only 0.090 seconds for the categorization process. However, his system only does binary classification. In contrast, our proposed system provides 99% accuracy for a multi-class classification for 12 different APT class. The system by Hamed et al. provides an overall accuracy of 95% for a multiclass classification for 4 APT classes where the number of the APT classes is 3 times lesser than our system. Hence, it can be seen that our proposed system performs better in terms of accuracy and is more efficient to perform the multiclass classification task.

7 Conclusion and Future Work

In this work, we have analyzed the malware files and successfully attributed them to their source family of the APT actor with 99% accuracy. We have used more than 3000 malware samples that belong to 12 APT groups and created 11 different views from the extracted features of Opcodes, Bytecodes, and Headers. Hence, our approach deals with a comprehensive analysis during the attribution process which makes the system resilient towards the complex obfuscation and evasion techniques, commonly used during APT campaigns. The multi-view approach used for the threat attribution provides the final prediction by considering the underlined results from individual single-views. It could optimize the final prediction accuracy to 99% which is higher than the respective single-views.

Because of the complex nature of APT attacks, it is not always easy to attribute a threat vector to its source. However, heuristic analysis using Machine Learning algorithms can be used to automate the threat attribution process with higher accuracy. The threat attribution results can contribute significantly to improve the decision-making process and reduce time during an investigation. During our experiment, we have used malware data belong to 12 APT groups and evaluated five different ML classifiers against it. We have observed that the performance of the ML classifiers varied with respect to different input views. SMOTE technique was used to balance the dataset. However, the SMOTE technique provided us with synthetic data samples that are different from the real data. Higher quality real-world data can help towards creating a more reliable system. Also, more ML algorithms can be evaluated against the data sets to optimize the system.

8 Acknowledgment Session

The authors would like to thank Dr. Ali Dehghantanha and Hamed Haddadpajouh for their valuable review during the research work.

References

[1] N. Pitropakis, E. Panaousis, A. Giannakoulias, G. Kalpakis, R. D. Rodriguez, and P. Sarigiannidis, "An Enhanced Cyber Attack Attribution Framework," in *Trust, Privacy and Security in Digital Business*, Cham, 2018, pp. 213–228, doi: 10.1007/978-3-319-98385-1_15.

[2] "Advanced Persistent Threat Groups," *FireEye*. https://www.fireeye.com/current-threats/apt-groups.html (accessed Jul. 05, 2020).

[3] D. Alperovitch, "Revealed: Operation Shady RAT," p. 14.

[4] H. HaddadPajouh, R. Khayami, A. Dehghantanha, K.-K. R. Choo, and R. M. Parizi, "AI4SAFE-IoT: an AI-powered secure architecture for edge layer of Internet of things," *Neural Comput & Applic*, Feb. 2020, doi: 10.1007/s00521-020-04772-3.

[5] J. Brownlee, "A Gentle Introduction to Imbalanced Classification," *Machine Learning Mastery*, Dec. 22, 2019. https://machinelearningmastery.com/what-is-imbalanced-classification/ (accessed Aug. 14, 2020).

[6] M. Ask, "Advanced Persistent Threat (APT) Beyond the hype Project report in IMT 4582 Network security at Gjøvik University College during spring 2013," 2013.

[7] K. Chang and D. Y.-D. Lin, "Advanced Persistent Threat:," p. 12.

[8] I. Ghafir *et al.*, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Generation Computer Systems*, vol. 89, Jul. 2018, doi: 10.1016/j.future.2018.06.055.

[9] Y. Su, M. Li, C. Tang, and R. Shen, "A Framework of APT Detection Based on Dynamic Analysis," Dec. 2015, pp. 1047–1053, doi: 10.2991/nceece-15.2016.187.

[10] B. E. Binde, R. McRee, and T. J. O'Connor, "Assessing Outbound Traffic to Uncover Advanced Persistent Threat," p. 35.

[11] D. Moon, H. Im, J. D. Lee, and J. Park, "MLDS: Multi-Layer Defense System for Preventing Advanced Persistent Threats," *Symmetry*, vol. 6, pp. 997–1010, Dec. 2014, doi: 10.3390/sym6040997.

[12] G. Vert, B. Gonen, and J. Brown, "A Theoretical Model for Detection of Advanced Persistent Threat in Networks and Systems Using a Finite Angular State Velocity Machine (FAST-VM)," *International Journal of Computer Science and Application*, vol. 3, p. 63, Jan. 2014, doi: 10.14355/ijcsa.2014.0302.01.

[13] M. Belaoued and S. Mazouzi, "A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features," *Journal of Information Processing Systems*, vol. 12, pp. 644–660, Dec. 2016, doi: 10.3745/JIPS.03.0058.

[14] H. Haddadpajouh, A. Azmoodeh, A. Dehghantanha, and R. M. Parizi, "MVFCC: A Multi-View Fuzzy Consensus Clustering Model for Malware Threat Attribution," *IEEE Access*, vol. 8, pp. 139188–139198, 2020, doi: 10.1109/ACCESS.2020.3012907.

[15] cyber-research, *cyber-research/APTMalware*. 2020.

[16] "tf.keras.preprocessing.text.Tokenizer | TensorFlow Core v2.3.0," *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer (accessed Aug. 14, 2020).
[17] H. Hashemi, A. Azmoodeh, A. Hamzeh, and S. Hashemi, "Graph embedding as a new approach for unknown malware detection," *J Comput Virol Hack Tech*, vol. 13, no. 3, pp. 153–166, Aug. 2017, doi: 10.1007/s11416-016-0278-y.
[18] J. Bai, Q. Shi, and S. Mu, "A Malware and Variant Detection Method Using Function Call Graph Isomorphism," *Security and Communication Networks*, vol. 2019, pp. 1–12, Sep. 2019, doi: 10.1155/2019/1043794.
[19] E. Carrera, *pefile: Python PE parsing module*. .
[20] "PE — LIEF 0.10.0-845f675 documentation." https://lief.quarkslab.com/doc/stable/api/python/pe.html (accessed Aug. 14, 2020).
[21] B. Rocca, "Handling imbalanced datasets in machine learning," *Medium*, Mar. 30, 2019. https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28 (accessed Aug. 14, 2020).
[22] A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner, "Scalable Fair Clustering," *arXiv:1902.03519 [cs]*, Jun. 2019, Accessed: Aug. 14, 2020. [Online]. Available: http://arxiv.org/abs/1902.03519.
[23] R. Gandhi, "Support Vector Machine — Introduction to Machine Learning Algorithms," *Medium*, Jul. 05, 2018. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (accessed Aug. 14, 2020).
[24] P. Gupta, "Decision Trees in Machine Learning," *Medium*, Nov. 12, 2017. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (accessed Aug. 14, 2020).