Analyzing Learning-based Encrypted Malware Traffic Classification with AutoML

Didier Frank Isingizwe
Department of Computer Science and Technology
Tsinghua University
Beijing, China
ididierfrank@yahoo.fr

Meng Wang
Department of Innovation Center
NSFOCUS Inc.
Xi'an, China
wangmeng7@nsfocus.com

Wenmao Liu
Department of Innovation Center
NSFOCUS Inc.
Beijing, China
liuwenmao@nsfocus.com

Dongsheng Wang
Department of Computer Science and Technology
Tsinghua University
Beijing, China
wds@tsinghua.edu.cn

Tiejun Wu
NSFOCUS FUYING LAB
NSFOCUS Inc.
Beijing, China
wutiejun@nsfocus.com

Jun Li
Department of Automation
Tsinghua University
Beijing, China
junl@tsinghua.edu.cn

***Abstract*—Malware traffic classification is an essential pillar of network intrusion detection systems. The explosive growth of traffic encryption makes it infeasible to classify malware traffic with port-based or signature-based approaches. Nowadays, researchers and industrial developers turn to learning-based approaches for encrypted malware traffic classification, mining the statistical patterns of traffic behaviors. However, different machine learning models with different hyper-parameters can be used, and one can hardly explain why a learning approach works or not. To alleviate this problem, this paper conducts encrypted malware traffic classification with the automated machine learning (AutoML) approach which contains 7 representative models in the pipeline and realizes automated hyper-parameter tuning and model assembling. Experimenting on real-world encrypted malware traffic, this paper analyzes the performance of the ensemble model of AutoML and how each model performs in detail to understand its contribution. Moreover, the analysis of AutoML feature selection shows discriminant features on encrypted malware traffic especially TLS metadata related. The concrete experiments and analysis give insight to the following studies on encrypted malware traffic classification.**

***Index Terms*—Malware, Encrypted Network Traffic, Automated Machine Learning, TLS, Traffic Classification**

I. Introduction

Traffic classification is an automated process that has been introduced to categorize computer network traffic according to various parameters (for example, based on port number or protocol type) into several traffic classes. The outstanding abnormal patterns are mostly referred to us as anomalies or outliers in this context.

Network encryption refers to the method of encrypting messages before and decrypting them after they are transmitted over a network. While it is crucial and necessary to protect the confidentiality of network communication and the privacy of network users, however, encryption prevents traditional network security products from inspecting the payload of network traffic being transmitted.

Taking into account that in a network setting, Transport Layer Security (TLS) has been the most dominant protocol to offer better encryption for network traffic, genuine traffic has seen fast adoption of the TLS standard over the previous decade, taking up as much as 60% of network traffic [1].

Different techniques have been used in anomaly detection, and data-driven approaches have been proven to be most effective in all aspects when it comes to encrypted network traffic, as it has advanced and extended capabilities to better understand the correlation between data or deeper meaning from large scale network traffic data.

In numerous studies, mixture or multi-level models have been recommended to expand the effectiveness and increase the accuracy in classifying network traffic specifically in the detection of anomaly [2-7]. The combined utilization of various dataset mining strategies is known as an ensemble approach, and the way toward learning the relationship between these ensemble methods is called meta-learning.

The approach of this research work is to introduce the significance of each learning paradigm and perceive how every one of them can be used to benefit overall performance, than pick the best one over the others. This methodology will lead us to find an augmented solution to reduce false alarms and computation time. In this paper, we will design, develop, and evaluate an efficient classifier for encrypted network traffic that accurately categorize malware traffic into their given families with the use of an automated machine learning approach.

The paper is organized as follows: Section II outlines related previous works concerning the basis of our methodology, and

Section III describes our proposed approach in a more detailed way as follow: how datasets are captured, how features are selected and extracted, and how evaluation experiments are conducted and analyzed, and section IV shows the actual experimental results with graphs that illustrate the evaluation metrics in a bigger picture, and section V outlines the conclusion.

II. Background

Recognizing and identifying threats in encrypted network traffic presents critical difficulties. However, two answers have been advanced in the recent years to take care of this issue. The traditional techniques basically decrypt all traffic that flows through a security apparatus, thus referred as Man-in-the-Middle (MITM) approaches [11]. When the traffic has been decoded, traditional signature-based strategies, like Snort [12], can be applied. While this methodology can be fruitful at discovering threats, there are a few significant inadequacies. In the first place, this technique does not regard the intended security protection on the network. Second, this technique is computationally costly and hard to convey and keep up. Third, this strategy depends on malware to not change their conduct when a MITM intervenes.

The advanced techniques for detecting threats in encrypted network traffic use flow-based metadata. These techniques analyze significant features of a network flow, such as the number of packets and bytes within a flow. This information is ordinarily sent out and put away as IPFIX [13] or NetFlow [14]. There have been a few papers that stretch the boundaries of conventional flow monitoring frameworks. For example, utilizing NetFlow and outside reputation scores to classify botnet traffic [15]. These work can likewise be applied to encrypted network traffic, yet does not exploit the TLS-explicit data signatures.

There has been a lot of previous work in the field of anomaly detection with the use of Machine Learning (ML) and the most challenging aspect of it is dealing with encrypted traffic, mainly extracting and selecting important features that have a degree of differentiation as well as finding effective ML models which accurately detect malicious flows and be able to adapt to various scenarios.

With regards to the supervised learning approach, straightforward multi-classifier model has been proposed for being used specifically for anomaly detection in network traffic dependent on big data [8]. With evaluation performed on the NSL-KDD dataset by utilizing the Weka tool for data pre-processing, the offered model has shown nice outcomes regarding anomaly detection accuracy and speed in general. Comprised of the Decision Tree (or J48), LogitBoost, IBk, AdaBoost, RandomTree classifiers, the model has not been hyper-parameter tuned to reach the maximum accuracy. In contrast, our work uses an automated machine learning approach that is rich in hyper-parameter tuning, and it significantly minimizes the computation time.

Aljawarneh and others [9] proposed utilizing a hybrid classifier approach comprising of J48, MetaPagging, Random Tree, REPTree, AdaBoostMl, DecisionStump, and NaïveBayes classifiers, with high accuracy and minimization of training time. However, algorithms were hand-picked and parameters were not fully optimized in this work, dissimilar to our approach with an automated feature selection and ensemble which is fully optimized and suitable in realtime scenarios due to its adaptability.

Imamverdiyev and Sukhostat [10] proposed a method that depends on informative data features for anomaly detection in network traffic. A new advanced method is described by the size and number of features which has been the fundamental issue in the analysis of encrypted network traffic. By mainly focusing on the most discriminant features, the feature selection (also known as reduction) procedure significantly improved the efficiency and accuracy.

Notwithstanding the idea of pure flow-based features to distinguish encrypted network traffic of malware, there have been numerous papers that increase and extend detailed data features about a flow [16-24]. For example, some extracted size and inter-arrival time of packets to get more insight about a flow, and others examined data features dependent on the packet size to analyze site fingerprinting attacks within encrypted network traffic. However, in our work, we mainly focused on identifying malware families by analyzing TLS metadata features.

There has been previous related work that employs active probing [25] and inactive monitoring to acquire perceivability into how TLS is utilized over the network [26]. In contrast, our outcomes explicitly prove that malware's utilization of the TLS protocol and the presence of TLS metadata features can be of great help in classifiers.

Various ML algorithms have been employed to identify malicious events, and the performance of those approaches is determined by the features and algorithms. MalFinder, an ensemble learning-based framework for malicious traffic detection, broadened the dimensions of statistical features and sequence features to portray network traffic [36]. Feature importance analysis and differentiation experiments delineate the adequacy of their new features. Among their chosen classifiers reasonable for malicious traffic detection, boosting-based classifiers XGBoost and LightGBM could lessen bias, and bagging-based classifier Random Forest could reduce variance. Stacking, which is the integration strategy for the classification results utilized in their system, could improve the speculation capacity of the technique. In comparison, our AutoML approach leverages a voting-stacking scheme with fully tuned parameters to further enhance the ensemble model performance.

Malware grouping and family attribution have had a great deal of openness in scholarly writing [27-30]. Interestingly, our work gives a better data collection approach to get more detailed insight from network flows and explore how malware flows utilize intraflow data and TLS metadata, and shows how the features that we selected exposes a high degree of differentiation for achieving an accurate and effective malware identification and family attribution.

III. Methodology

The adaptability of the existing approaches have been facing challenges. Automated methods are mainly used in dynamic scenarios where adaptability is a crucial characteristic to take into account.

Many existing works use the pre-selected feature approach, however, pre-selected features are really limited and may not include all/enough heuristics for traffic classification, so automated feature selection is necessary. Hand-picked algorithms and their combination has been being difficult especially in making choices and optimizing parameters, hence an automated ensemble solution is the way to go.

In this section, we are mainly analyzing and extracting features that are having a degree of differentiation for classifying encrypted malware traffic into their given families as well as proposing an automated machine learning approach which outputs a new hybrid ensemble model composed of best classifiers with fine-tuned parameters to achieve maximum accuracy while reducing false alerts and minimizing computation time.

A. Data Collection

In our study, we used a real traffic dataset that contains 3 TLS encrypted malware families traffic captured from our sandbox in 2020.

To collect raw network traffic data, we adopted the open-source package named Cisco Joy [31]. It is usually used to capture and analyze the network flow data (SourceIP, DestinationIP, SourcePort, DestinationPort, Proto, Number of Bytes, Number of Packets, and so on), intraflow data (Sequence of Packet Lengths and Times, Byte Distribution and Entropy, and so on), TLS metadata (Ciphersuites, Extensions, Server Name Indication (SNI), Certificate Strings, etc.), DNS data and HTTP data, for network research and security monitoring. Cisco Joy can effectively extract data features from live network traffic or packet capture (pcap) files, using a flow-oriented model like that of IPFIX or Netflow, and then represent all these data features in JSON format. It is also equipped with various analysis tools that can be applied to these data for different purposes. Cisco Joy has been used by many researchers in exploring data at scale, especially in network security, and threat-relevant data.

B. Feature Selection

The choice of relevant and discriminant features is arguably one of the most important and crucial steps in the creation of a high quality model suitable for anomaly detection tasks. Preferably, these selected features must have the following properties to fully maximize detection while minimizing management and storage costs [32]:

- Compact: Considering that the number of bits needed to store one observation (composed of multiple relevant features) should be significantly smaller than the original flow size to ensure that a large number of meaningful observations can be fully stored and re-used later for training purposes.
- Informative: the selected features must contain relevant insights to the initial problem and should be as independent as possible from one another to achieve better results.
- Economical: collecting and capturing network flows and extracting features should not require excessive computing power and time.

Considering the observations and a study of TLS parameters [32] by Cisco, the features that have been selected by Cisco have been presented in Fig. 1. They can be grouped into 3 following categories:

- Flow Metadata: data features that are not related to TLS but are observable in any NetFlow over the network.
- Distributions: data features that cannot be directly & fully extracted from network packets but they can be considered to be the results of some kind of frequency analysis when it comes to packets flow.
  - Sequence of Packet Lengths (SPL). Packet lengths are usually placed into bins for better processing. For example, the packet of size in the range from 0 to 150 will be then placed into the first bin and so on.
  - Sequence of Packet Time (SPT). In this case, we can consider it the same as for SPL but with packets' inter-arrival times instead of packet length.
  - The Byte distribution. In this case, the length-256 of an array can be used in keeping the count of each byte size encountered as the payload of packets.
- TLS Metadata: These are data features that are fully extracted from TLS handshake packets (ClientHello, ServerHello, Certificate, Client Key Exchange, Change Cipher Specification) to tackle TLS malware activities in encrypted flows.

Feature	Size	Dynamic?	In reduced set?	Type
Ephemeral src port	1	No	Yes	Boolean
TLS dest port	1	No	Yes	Boolean
Nb of inbound bytes	1	No	No	Integer
Nb of outbound bytes	1	No	No	Integer
Nb of inbound packets	1	No	No	Integer
Nb of outbound packets	1	No	No	Integer
Flow duration	1	No	No	Integer
SPL	100	No	No	Stochastic matrix
SPT	100	No	No	Stochastic matrix
Byte dist mean	1	No	No	Float
Byte dist std	1	No	No	Float
Byte entropy	1	No	No	Float
Ciphersuites	146	Yes	Yes	Binary vector
Extensions	16	Yes	Yes	Binary vector
Nb of extensions	1	No	Yes	Integer
Supported Groups	36	Yes	Yes	Binary vector
Point Formats	4	No	Yes	Binary vector
Client's key length	1	No	No	Integer
Certificate's validity	1	No	Yes	Integer
Certificate's nb of SAN	1	No	Yes	Integer
Self-signed certificate	1	No	Yes	Boolean
Total	417		208	

Fig. 1. Cisco's Features for TLS Malware Detection

C. Classification Method

Automated Machine Learning (AutoML) is the method that enables researchers and developers to build efficient machine learning models with high scalability, efficiency, and productivity while sustaining and maintaining the model quality in all

aspects. AutoML itself is also an ML method that focuses on automating the time-consuming iterative tasks of ML model development. The built-in ensemble learning capabilities of AutoML improve the predictive performance of the modeling result by combining multiple different algorithms into one as opposed to using individual algorithms. It also uses both voting and stacking ensemble methods so that the final ensemble model comes out with fully tuned hyper-parameters to reach maximum accuracy and a shorter training time. During the training stage, AutoML builds several pipelines in parallel for different algorithms and parameters and then iterates through all given ML algorithms paired with feature selections, where each iteration output a model with a training score. The higher the score, the better that specific model will be considered to fit the target dataset.

Many traditional ML model development methods are usually resource-intensive, requiring significant domain knowledge as well as the time to produce and compare many models. The use of AutoML surely accelerates the time it takes to get production-ready ML models with great ease and efficiency in all aspects.

In our study, an open-source automated machine learning python package named "mljar-supervised" is adopted. Under the MIT license, it has been proven to be efficient while performing feature engineering as well as hyper-parameter tuning, with source code and documentation publicly available [33, 34].

Our approach is based on an AutoML pipeline that contains 7 representative algorithms that have been commonly used in related works and have proven to be the most effective in supervised learning methods as follows: Decision Tree, Logistic Regression, Random Forest, XGBoost, CatBoost, LightGBM & Neural Networks. This AutoML method utilizes a stacking approach where the stacked models are built from past (unstacked) models. The stacked models reuse tuned hyper-parameters of effectively discovered great models.

- During the stacking stage up to 10 best models from every algorithm are utilized.
- The out-of-folds predictions are utilized to build broadened training dataset and the stacking just mainly works for validation strategy="kfold" (k-fold cross-validation).
- In our case, the AutoML chooses the best models from unstacked XGBoost, LightGBM, CatBoost models, and reuses their tuned hyperparameters to prepare and train stacked models.

AutoML results in a new hybrid ensemble model which accurately classifies encrypted malware traffic in their given family. The ensemble procedure in AutoML performs a greedy search of used models and attempts to add (with re-iteration) a model to the ensemble to improve its performance [35]. The ensemble performance is computed dependent on out-of-folds predictions of utilized best models. In Fig. 2, we introduced an AutoML approach that uses a voting-stacking scheme to pick the best classifiers to be used in the ensemble model. The main benefit of this approach is that the AutoML pipeline does the work for us in testing all scenarios when tuning parameters as well as finding the best-ensembled model which will be effective in classifying the encrypted malware traffic accurately and minimizing computation time.

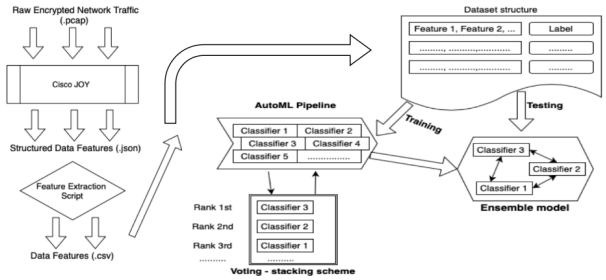

Fig. 2. An AutoML Approach With Voting-Stacking Scheme

IV. Evaluation

This section presents the results obtained from the AutoML pipeline which used 7 representative algorithms. Our real traffic dataset contains network traffic of 3 malware families, where 0 indicates Benign, 1, 2, and 3 indicates Zbot, Bublik, and Small, respectively.

- Zeus, ZeuS, or Zbot is a Trojan horse malware that usually runs on various versions of Microsoft Windows.
- Bublik (also known as Pkybot) malware steals information about the affected systems and transmits it to a remote server.
- Small is another type of Trojan horse malware that secretly downloads malicious files from a remote server, then installs and executes the files on the victim system without end-users awareness.

A. AutoML Overall Performance Metric Results

As shown from Fig. 3, after running the AutoML pipeline with 7 classifiers we got the overall performance of all classifiers in terms of log loss value and training time. The AutoML pipeline has automatically built an ensemble model for this experiment by combining the best classifiers to build a more optimized and advanced model named "Ensemble" its log loss value significantly proven to be best in consideration with it's training time after many testing trials. From Fig. 3, we can see each classifier's performance relative to other ones and the AutoML pipeline decided to use the best ones as shown with less log loss value.

B. Ensemble Model Performance

In table I, we can see the structure of the ensembled model that we got from the AutoML pipeline which is composed of all classifiers used as well as their weight which is equivalent to how many times it has been re-used to reach the maximum accuracy as well as reducing training time.

In Fig. 4, we used 3 metric types, i.e. precision, recall, and support, to measure the performance of our ensemble model.

In Fig. 5, the learning curve shows the evaluation metric values in each iteration of the training. The learning curves are plotted for training and validation datasets. The log loss has

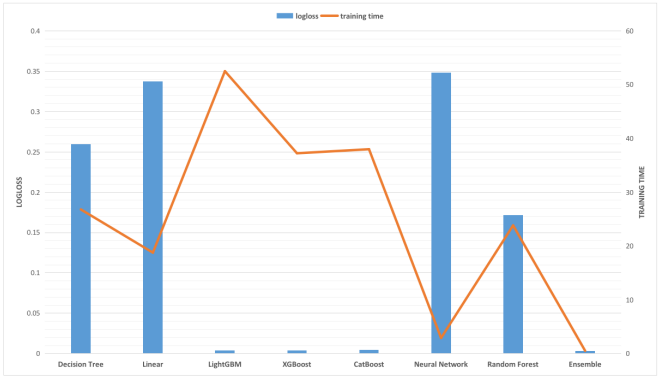

Fig. 3. Overall Performance of Individual and Ensemble Models

TABLE I
ENSEMBLE MODEL STRUCTURE

Modal	Weight
LightGBM	4
XGBoost	2
CatBoost	1

significantly dropped down and reached a very low level only after two iterations. This indicates that our ensemble model performs well.

Next, we illustrate how the best classifiers performed in a detailed way to understand the main reason it has been voted to be used considering feature importance metrics in our ensemble model.

1) LightGBM Classifier Performance: Again, precision, recall, and support are the metrics types leveraged to measure and visualize the performance of the LightGBM Classifier. As seen from Fig. 6, all 4 classes achieved at least the precision of over 99% which contributed a lot to the ensemble model, however, the recall which is also known as the true positive rate has relatively dropped on Bublik and Small malware in comparison to their precision which is also the reason why the AutoML pipeline added more classifiers to compensate on this weakness. Fig. 7 shows the learning curve of the

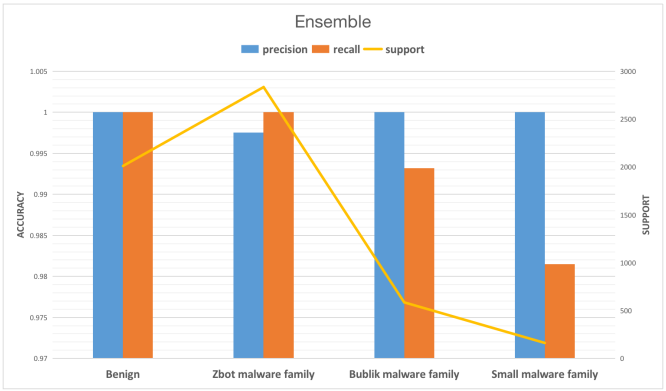

Fig. 4. Ensemble Model Performance Metric

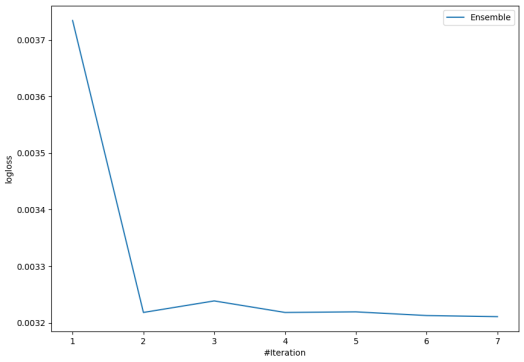

Fig. 5. Ensemble Model Learning Curve

LightGBM Classifier, where the vertical line shows the optimal iteration number. It can be seen that the logloss value dropped significantly with iterations, and the train test logloss value distributions align close to each other. It proves that the model has successfully classified datasets at the testing stage, in respective classes as it has been trained.

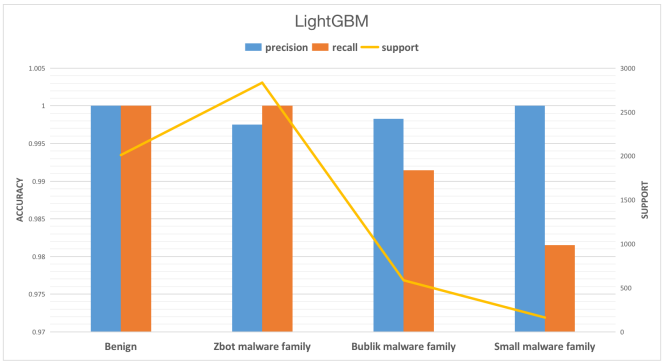

Fig. 6. LightGBM Classifier Performance Metric

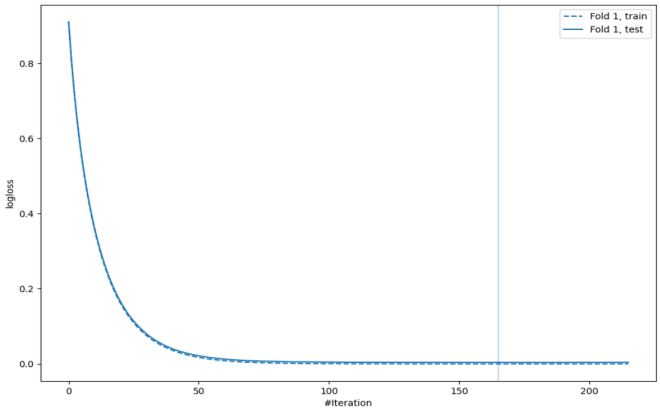

Fig. 7. LightGBM Classifier Learning Curve

In Fig. 8, all data features are significant and have contributed a lot to this model performance, however, the top 5

discriminant data features for LightGBM Permutation are as follow:

- The number of incoming bytes (denoted as 7)
- The number of outgoing bytes (denoted as 8)
- The number of outgoing packets (denoted as 6)
- The number of password packages provided by client (denoted as 646)
- The number of incoming packets (denoted as 5).

As seen from Fig. 8, it is clear that malware traffic can be detected by verifying the number of password packages that have been provided by the client but also in consideration of other discriminant data features. In Fig. 9, all data features

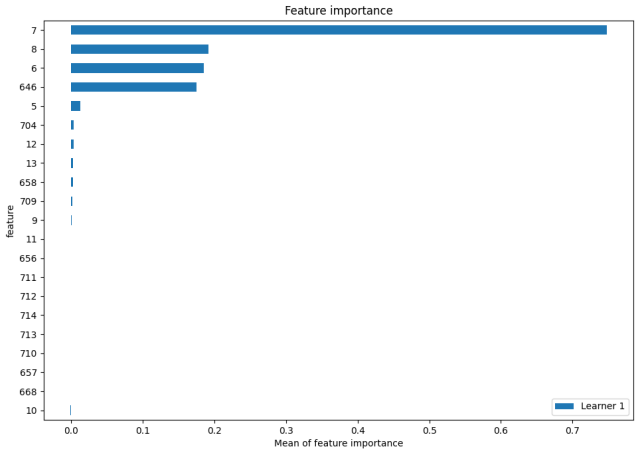

Fig. 8. LightGBM Permutation-based Feature Importance

are significant and have contributed a lot to this model performance, however, the top 5 discriminant data features for LightGBM SHAP are as follow:

- The number of incoming bytes (denoted as 7)
- The number of password packages provided by client (denoted as 646)
- TLS version provided by TLS client (denoted as 709)
- The number of outgoing packets (denoted as 6)
- The number of outgoing bytes (denoted as 8).

As seen from Fig. 9, we discovered one more data feature denoted as TLS version provided by TLS client which contributed a lot to this model performance because, in our experiment, we have found out that malware traffic uses TLS versions very wisely to escape from being detected. It is in this regard that we need to use different metric types to analyze the model performance in a detailed way. In Fig. 10, the top 5 discriminant data features for LightGBM learner fold 0 shap are as follow:

- The number of incoming bytes (denoted as 7)
- The number of password packages provided by client (denoted as 646)
- TLS version provided by TLS client (denoted as 709)
- The number of outgoing packets (denoted as 6)
- The number of outgoing bytes (denoted as 8).

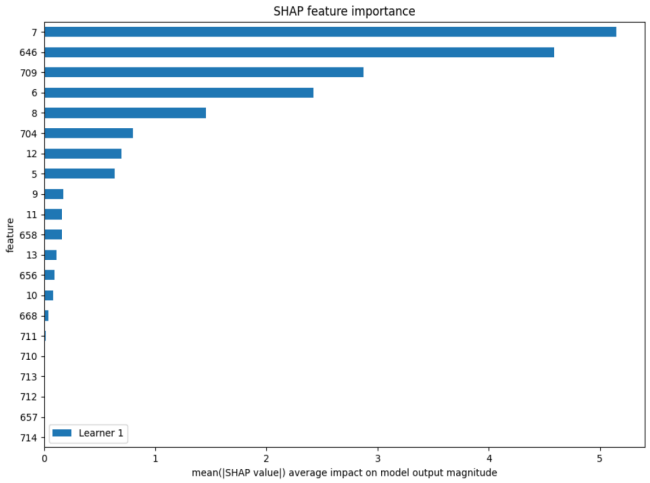

Fig. 9. LightGBM SHAP Feature Importance

In Fig. 10, we are illustrating malware families and benign traffic distribution in feature importance to have a better picture of which features have been able to accurately classify more data points of given class label. From Fig. 10, we can see that the number of elliptic curves supported in the extension (denoted as 704) has been fully used to classify benign traffic, together with the number of password packages provided by the client (denoted as 646) and TLS version provided by TLS client (denoted as 709). All data features presented in Fig. 10 are significant to our model performance, and this hints that if one data feature is ignored then it would definitely lower the model performance.

2) XGBoost Classifier Performance: As seen from Fig. 11, XGBoost is doing better in making Bublik standing out (best precision for Bublik) and it has improved the recall for Bublik in comparison with LightGBM. In Fig. 12, we can see that the XGBoost classifier used many iterations compared with LightGBM to significantly drop the learning curve.

In Fig. 13, all data features are significant and have contributed a lot to this model performance, however, the top 5 discriminant data features for XGBoost Permutation are as follow:

- The number of incoming bytes (denoted as 7)
- The number of outgoing bytes (denoted as 8)
- The number of outgoing packets (denoted as 6)
- The total entropy (denoted as 12)
- The number of password packages provided by client (denoted as 646).

As seen from Fig. 13, the total entropy and the number of password packages provided by the client are not as significant as other data features, however, the total entropy was not that important (at least not in the top 5) for LightGBM, this means that different algorithms are using features differently to achieve maximum performance. In Fig. 14, all data features are significant and have contributed a lot to this model performance, however, the top 5 discriminant data features for XGBoost SHAP are as follow:

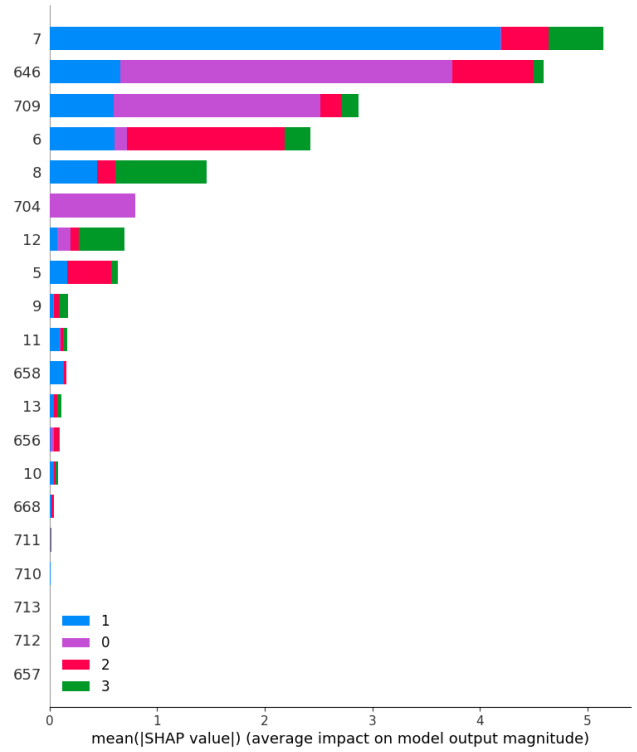

Fig. 10. LightGBM Learner Fold 0 SHAP Feature Importance

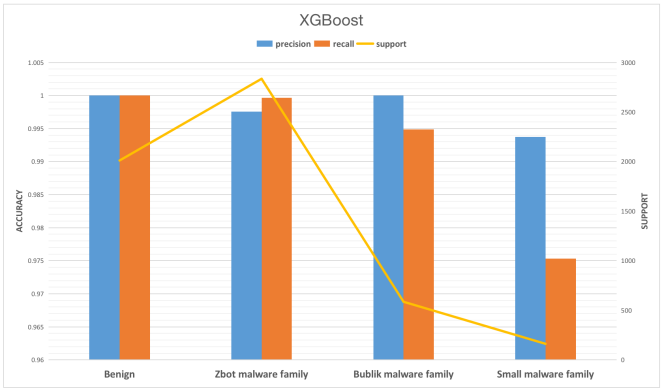

Fig. 11. XGBoost Classifier Performance Metric

- The number of password packages provided by client (denoted as 646)
- The number of outgoing packets (denoted as 6)
- The number of incoming bytes (denoted as 7)
- The number of outgoing bytes (denoted as 8)
- TLS version provided by TLS client (denoted as 709).

As seen from Fig. 14, we discovered one more data feature denoted as TLS version provided by TLS client which contributed a lot to this model performance because, in our experiment, we have found out that malware traffic uses TLS versions very wisely to escape from being detected. It is in this regard that we need to use different metric types to analyze

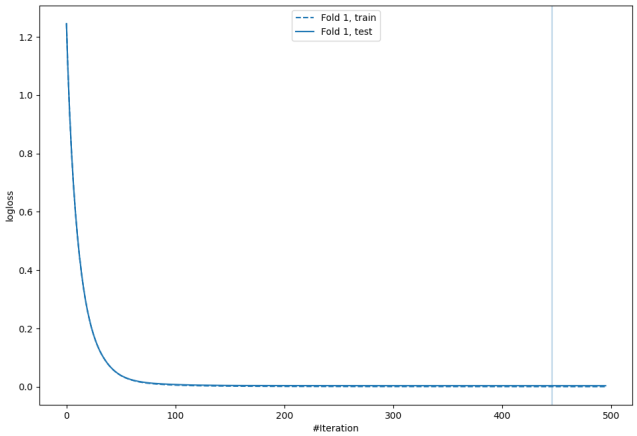

Fig. 12. XGBoost Classifier Learning Curve

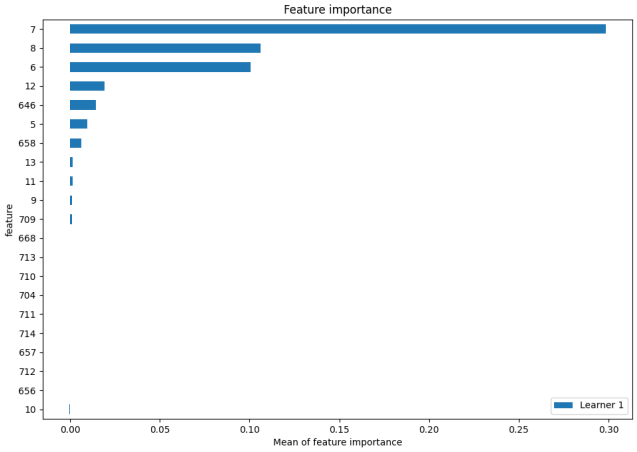

Fig. 13. XGBoost Permutation-based Feature Importance

the model performance in a detailed way. In Fig. 15, the top 5 discriminant data features for XGBoost learner fold 0 shap are as follow:

- The number of password packages provided by client (denoted as 646)
- The number of outgoing packets (denoted as 6)
- The number of incoming bytes (denoted as 7)
- The number of outgoing bytes (denoted as 8)
- TLS version provided by TLS client (denoted as 709).

In Fig. 15, we can see that benign traffic class denoted as 0-purple has dominantly used the number of password packages provided by client and TLS version provided by TLS client features compared to other features.

3) CatBoost Classifier Performance: As seen from Fig. 16, CatBoost improved on Zbot recall which is also known as true positive rate compared with LightGBM and XGBoost. In Fig. 17, we can see that CatBoost used more iterations compared to LightGBM and XGBoost.

In Fig. 18, all data features are significant and have contributed a lot to this model performance, however, the top

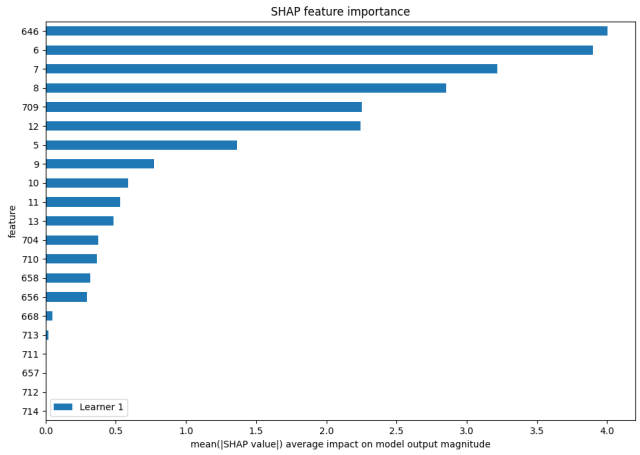

Fig. 14. XGBoost SHAP Feature Importance

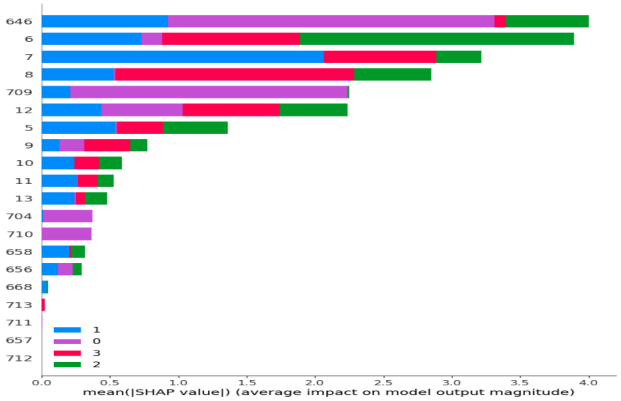

Fig. 15. XGBoost Learner Fold 0 SHAP Feature Importance

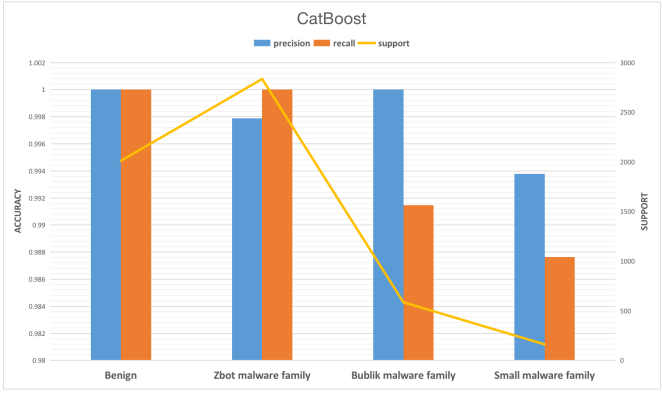

Fig. 16. CatBoost Classifier Performance Metric

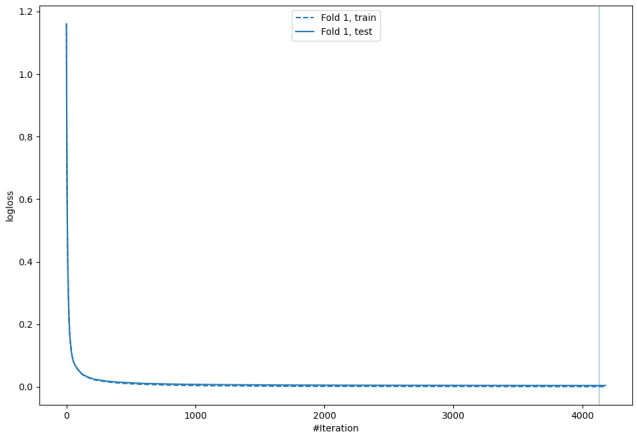

Fig. 17. CatBoost Learning Curve

5 discriminant data features for Catboost Permutation are as follow:

- The number of incoming bytes (denoted as 7)
- The number of outgoing bytes (denoted as 8)
- The number of outgoing packets (denoted as 6)
- The password suite selected by the server (denoted as 658)
- The total entropy(denoted as 12).

As seen from Fig. 18, we discovered another new discriminant data feature denoted as the password suite selected by the server which was not the case for LightGBM and XGBoost. This again demonstrated that different algorithms can use several data features differently to achieve a better ensemble model performance.

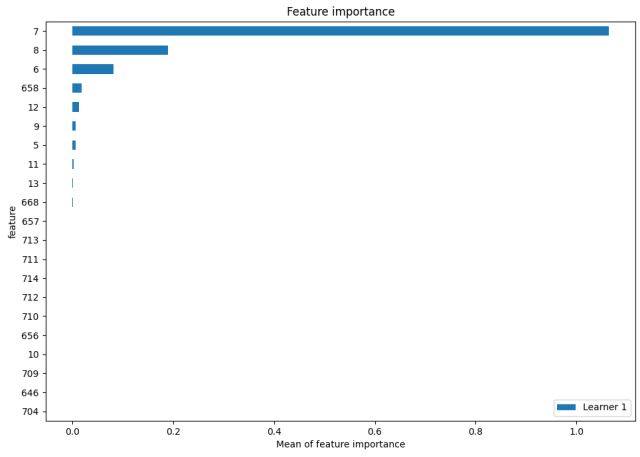

Fig. 18. CatBoost Permutation-based Feature Importance

V. Conclusion

The network security field presents a novel and unique set of challenges, outstandingly, the huge volume of the network traffic data, high demand for low false alarms, and

fast processing speed. In our work, an automated machine learning tool AutoML is employed to explore and analyze the encrypted malware network traffic classification methods based on ML. An effective way was proposed for the data collection of raw encrypted network traffic with the help of the Cisco Joy tool as well as the selection of the most discriminant features, such as TLS metadata features directly extracted from TLS handshake packets. These features offer a significant degree of differentiation while detecting encrypted malware traffic over the network. Considering the tricky dynamics of malware traffic, the adaptability of classifier systems must surely be enhanced, therefore, an automated machine learning approach is greatly suitable in contributing and assisting existing models to adapt to new dynamic scenarios especially in real-time environments. Hybrid methods have been proven to contribute a lot to the performance improvement of ML classifiers, however, the main problem that many ML-based classifiers face is hyper-parameter tuning because it requires expensive computation resources in case of large data volume. It is in this regard that the AutoML approach is proposed to maximize accuracy, reduce false alarms while minimizing the training and inferring time. Although it is found the automated machine learning approach to be the most robust for this specific problem domain, it is also recognized that the choice of features had a significant impact on the overall performance. An enhanced feature set has been built by using an automated feature selection approach which takes all available features that can be obtained from the raw encrypted network traffic especially the TLS metadata features and then statistically selected the most discriminant features based on their contributions to the performance of all selected classifiers to maximize the accuracy and efficiency of the proposed approach. By not only relying on data features that were convenient to gather and engaging with domain experts to iterate on how the data would be best represented, but the algorithms used in the proposed AutoML pipeline also had significant performance improvements. Combining the automated feature selection and the ensemble classifier model is the key innovative contribution in our work. In our experiments, only 7 classifiers (Decision Tree, Logistic Regression, Random Forest, XGBoost, CatBoost, LightGBM, and Neural Networks) are used in the AutoML pipeline but there is still more room for exploration and improvement like adding more classifiers as well as doing more testing on live-network as it is the main reason of using automated approaches to maximize the performance and minimize the false alarms.

REFERENCES

[1] Most Internet Traffic will be Encrypted by Year End. Here's Why. http://fortune.com/2015/04/30/netflix-internet-traffic-encrypted/, accessed: 2016-03-23.

[2] H. Kim, J. Kim, I. Kim, and T. M. Chung, "Behavior-based anomaly detection on big data," in Proceedings of the 13th Australian Information Security Management Conference, 30 November â€“ 2 December, Edith Cowan University, Western Australia, 2015, pp. 73-80.

[3] R. M. Alguliyev, R. M. Aliguliyev, Y. N. Imamverdiyev, and L. V. Sukhostat, "An anomaly detection based on optimization," International Journal of Intelligent Systems and Applications, vol. 12, pp. 87-96, 2017.

[4] H. H. Pajouh, G. Dastghaibyfard, and S. Hashemi, "Two-tier network anomaly detection model: A machine learning approach," Journal of Intelligent Information Systems, vol. 48, pp. 61-74, 2017.

[5] R. M. Alguliyev, R. M. Aliguliyev, and F. J. Abdullayeva, "Hybridisation of classifiers for anomaly detection in big data," International Journal of Big Data Intelligence, vol. 6, pp. 11-19, 2019.

[6] S. Aljawarneh, M. Aldwairi, and M. B. Yasin, "Anomaly-based intrusion detection system through feature selection analysis and building a hybrid efficient model," Science Journal of Computational, vol. 25, pp. 152-160, 2018.

[7] A. Branitskiy and I. Kotenko, "Hybridization of computational intelligence methods for attack detection in computer networks," Journal of Computational Science, vol. 23, pp. 145-156, 2017.

[8] Ramiz M. Aliguliyev Makrufa Sh. Hajirahimova. "Classification Ensemble Based Anomaly Detection in Network Traffic," Review of Computer Engineering Research, Conscientia Beam, vol. 6(1), pp 12-23, 2019.

[9] S. Aljawarneh, M. Aldwairi, and M. B. Yasin, "Anomaly-based intrusion detection system through feature selection analysis and building a hybrid efficient model," Science Journal of Computational, vol. 25, pp. 152-160, 2018.

[10] Y. Imamverdiyev and L. Sukhostat, "Network traffic anomalies detection based on informative features," Radio Electronics, Computer Science, Control, pp. 113-120, 2017.

[11] F. Callegati, W. Cerroni, and M.Ramilli, "Man-in-the-middle attack to the HTTPS protocol," IEEE Security Privacy, vol.7(1), pp 78–81, 2009.

[12] M. Roesch, "Snort - lightweight intrusion detection for networks," Proceedings of the 13th USENIX Conference on System Administration," pp. 229–238. LISA, USENIX Association (1999).

[13] B. Claise, B. Trammell and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) protocol for the exchange of flow information," RFC 7011, 2013.

[14] B. Claise,"Cisco Systems NetFlow Services Export Version 9," RFC 3954, 2013.

[15] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: Detecting Botnet Command and Control Servers through Large-Scale NetFlow Analysis. In: Proceedings of the 28th Annual Computer Security Applications Conference. pp. 129–138. ACM (2012).

[16] Dietrich, C.J., Rossow, C., Pohlmann, N.: Cocospot: Clustering and Recognizing Botnet Command and Control Channels using Traffic Analysis. Computer Networks 57(2), 475–486 (2013).

[17] Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection. In: USENIX Security Symposium. vol. 5, pp. 139–154 (2008).

[18] Moore, A.W., Zuev, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. In: ACM SIGMETRICS Performance Evaluation Review. vol. 33, pp. 50–60. ACM (2005).

[19] Panchenko, A., Lanze, F., Zinnen, A., Henze, M., Pennekamp, J., Wehrle, K., Engel, T.: Website Fingerprinting at Internet Scale. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2016).

[20] Wang, K., Cretu, G., Stolfo, S.J.: Anomalous Payload-Based Worm Detection and Signature Generation. In: Recent Advances in Intrusion Detection. pp. 227–246. Springer (2006).

[21] Wang, L., Dyer, K.P., Akella, A., Ristenpart, T., Shrimpton, T.: Seeing through Network-Protocol Obfuscation. In: Proceedings of the Conference on Computer and Communications Security (CCS). pp. 57–69. ACM (2015).

[22] Williams, N., Zander, S., Armitage, G.: A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification. Computer Communication Review 30 (2006).

[23] Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C., Kirda, E.: Automatically Generating Models for Botnet Detection. In: Computer Security–ESORICS 2009, pp. 232–249. Springer (2009).

[24] Zander, S., Nguyen, T., Armitage, G.: Automated Traffic Classification and Application Identification using Machine Learning. In: The 30th IEEE Conference on Local Computer Networks. pp. 250–257. IEEE (2005).

[25] Durumeric, Z., Wustrow, E., Halderman, J.A.: Zmap: Fast Internet-Wide Scanning and Its Security Applications. In: USENIX Security Symposium. pp. 605–620 (2013).

[26] Holz, R., Amann, J., Mehani, O., Wachs, M., Kaafar, M.A.: TLS in the Wild: an Internet-Wide Analysis of TLS-Based Protocols for Electronic

Communication. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2016).
[27] Anderson, B., Storlie, C., Lane, T.: Multiple Kernel Learning Clustering with an Application to Malware. In: 12th International Conference on Data Mining (ICDM). pp. 804–809. IEEE (2012).
[28] Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C., Kirda, E.: Scalable, Behavior-Based Malware Clustering. In: Proceedings of the Network and Distributed System Security Symposium (NDSS). vol. 9, pp. 8–11. Citeseer (2009).
[29] Perdisci, R., Lee, W., Feamster, N.: Behavioral Clustering of HTTP-Based Malware and Signature Generation using Malicious Network Traces. In: NSDI. pp. 391–404 (2010).
[30] Rieck, K., Holz, T., Willems, C., Du ssel, P., Laskov, P.: Learning and Classification of Malware Behavior. In: Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 108–125. Springer (2008).
[31] https://github.com/cisco/joy
[32] David McGrew and Blake Anderson. “Enhanced telemetry for encrypted threat analytics”. In: Proceedings of the 2016 IEEE 24th International Conference on Network Protocols (ICNP). 2016, pp. 1–6. DOI: 10.1109/ICNP.2016.7785325.
[33] https://github.com/mljar/mljar-supervised
[34] https://supervised.mljar.com
[35] Caruana, Rich Niculescu-Mizil, Alexandru Crew, Geo Ksikes, Alex. (2004). Ensemble Selection from Libraries of Models.
[36] C. Rong, G. Gou, M. Cui, G. Xiong, Z. Li and L. Guo, ”MalFinder: An Ensemble Learning-based Framework For Malicious Traffic Detection,” 2020 IEEE Symposium on Computers and Communications (ISCC), 2020, pp. 7-7.