Supervised Learning for State-Sponsored Malware Attribution

Coen Boot*, Erik Poll*, Alex Serban*
*Radboud University,
Nijmegen,
The Netherlands
{c.b}@abc.com

***Abstract*—Cyber-threats have evolved in numerous ways over the past decade, causing malware authorship attribution to be a increasingly complex problem. Recent work has described several techniques to make authorship attribution possible to some extent, often by using Machine Learning (ML) techniques. Since ML techniques are designed for finding patterns in complex data, it is plausible that they outperform other, more conventional methods. Finding a solution which reliably performs malware attribution could benefit law enforcement and antivirus vendors in the understanding and monitoring of malware campaigns and families.**

The first contribution that we make in this paper, consists of a dataset of over 3,500 state-sponsored malware samples, labeled by Advanced Persistent Threat (APT) group and country, which can be used for future benchmarks. The dataset is constructed using a method which has not been described before. Next to that, we replicate the most notable experiments regarding malware authorship attribution using our dataset, and present the results.

I. Introduction

Cyber-threats have extended from individual or small organized groups to professional intelligence officers leading state-sponsored cyber-missions. With access to more resources, the attack sophistication has also increased and raises new challenges for creating defenses or attributing attacks to certain authors. In particular, there are higher incentives to hide attack traces, since they can have direct political and economical consequences. Conversely, cyber-attack attribution becomes more important to governmental or intergovernmental organizations.

The increased sophistication of attacks, their relatively high frequency and the intrinsic incentive to hide their traces makes the authorship attribution problem ill-posed. In the simplest terms, relating authors to attacks requires searching for particular traces of their provenience, such as IP addresses that can be traced back to locations or instructions in specific languages (e.g. Russian or Chinese). Nonetheless, the information that can be extracted from traces is often complex and highly dimensional, causing an increased need to automate (at least parts of) the analysis in order to discover common characteristics between cyber-attacks. If any samples are available which are known (with high confidence) to belong to certain authors, the problem can be posed as finding similarities between such samples and new examples.

Finding complex similarities, or patterns, between samples is the crux of ML algorithms. In case the samples are already annotated with labels (e.g. if a person already labeled the attacks as belonging to one actor), the problem can be formulated as a supervised learning problem. Whenever the labels are not available, the problem is suitable for unsupervised learning - a class of algorithms that discover similarities between samples without label attribution (e.g. grouping samples together).

Indeed, using ML algorithms to perform software authorship attribution has been studied over the last decade [1]. However, many of the proposed methods cannot be used to classify malware because the malware source code is not available and because most malware is equipped with trace or intend hiding techniques. From the few approaches which are able to cope with (complex) malware, the benchmark datasets or the code associated with the experiments are not publicly available, impeding experiment replication and reuse by practitioners.

In this paper, we verify to what extent supervised learning methods are suitable for malware authorship attribution. We choose to explicitly focus on approaches which use sandbox reports as input, since sandboxes are an easy way to gather much information about a malware sample [2]. We test the results from two of the most relevant publications which tackle this problem from different angles: using ML algorithms with high capacity that are less interpretable and using ML algorithms with lower capacity, but with interpretable results. Being able to extract meaningful data from the algorithms used increases their relevance for the attribution task. We can imagine that accusing an actor of cyber-attacks on the basis of algorithmic results is not possible. However, less interpretable algorithms, with higher capacity, often give better results and are worth investigating.

Our contributions in this paper are twofold. Firstly, since no public dataset is available, an initial contribution is the construction and publication of a malware dataset (with over 3,500 samples by 12 different APTs), which can be used for future benchmarks. We focus on state-sponsored malware because it has a richer hierarchy of authorship; ranging from country-level to different APT groups. It is often hard to distinguish between individual authors and APT groups. An author can be part of an APT group, but can also be one individual. In the latter case, authorship attribution is not easy.

Using this dataset, we replicate the most notable experiments for malware authorship attribution and present the results (we also make the code available). This marks the second contribution of the paper.

The presentation develops along the following lines: In Section II we describe the task and introduce related work. In Section III we present the dataset, followed by the experimental results in Section IV and a discussion in Section V. Conclusions and future research can be found in Section VI.

II. Background and Related Work

The problem of malware authorship attribution raises several concerns. Firstly, since the source code of malware samples is not available, the data is limited to binary executables. This is a concern because compilation removes many stylistic features that can hold valuable information about the authors [3].

Secondly, many malware binaries are processed using origin-hiding or intent-hiding techniques, such as packers, obfuscators or techniques which hinder active analysis (e.g. sandbox detection and evasion) [1], [3]. This makes automated attribution on binary malware samples more difficult, since multiple authors may e.g. use the same packer, making their binaries look more similar.

Lastly, malware authors try to trick malware analysists into drawing erroneous conclusions by adding fake traces of authorship, like changing the timezone of the machine they are working with, using foreign languages in their code or re-using attacks from different authors [4].

Running the malware samples in sandbox environments can reveal more information than analyzing the binaries statically, because of two reasons. Firstly, some obfuscation methods are removed during runtime (due to malware unpacking). Secondly, the information that can be extracted is more sensible and easier to interpret. This data can be extracted using automated malware analysis software, which runs the samples and outputs a verbose report about their behavior. In literature, there are two ways to use these reports: using all the information contained in the reports or using only the API calls to the operating system.

In the next sections, we present supervised and unsupervised classification methods using either binary data or sandbox reports.

a) Supervised Methods: One of the first supervised learning methods [5] creates a general profile for each malware family using n-grams of the machine codes extracted using IDA Pro [6]. N-grams exploit static structure in the machine code. However, for complex malware behavior, which spans many instructions, n-grams have limited impact. Moreover, obfuscation techniques can also affect the results.

David and Netanyahu [7] feed the full report content to a Deep Neural Network (DNN) autoencoder which compresses the data to a smaller representation, which can be used for classification as well. Later, Rosenberg, Sicard and David [8] use a fully connected DNN for classification. The difference between the two is that autoencoders are trained to reconstruct the initial data from the internal representation (similar to compression and decompression) while fully connected networks are only trained to minimize the classification error. However, one disadvantage of using neural networks is that their results are hard to interpret. Several methods for post-hoc interpretation (e.g. sensitivity analysis) have been proposed. Unfortunately, they provide no guarantees that the same extracted data is used every time, for each class. An interpretable alternative to DNNs was presented by Aman et al. [9], which uses a Random Forest Classifier (RFC) on the sandbox reports.

Two publications use DNNs on the extracted API calls with similar results [10], [11].

b) Unsupervised Methods: Unsupervised methods do not require that data are labeled and can be used to group similar samples. Further analysis can uncover which groups belong to an author. One of the first publications was presented by Konrad et al. [12], who use sandbox reports in order to cluster malware samples based on families. Later, the US Cyber Genome program developed a more advance approach, which uses clustering in order to identify different building blocks inside a malware binary [13].

In our experiments we use sandbox reports from several vendors with DNNs, similar to [8], and with Random Forests, as in [9]. Moreover, we run the experiments using both full reports and API calls only.

III. Dataset

Since no public dataset is available, we first construct and publish our benchmark dataset. We note again that supervised learning requires *labels* associated with malware samples and that our focus is on state-sponsored malware because it has a richer hierarchy which enables us to distinguish between state actors and APT groups.

A. Data Collection

State-sponsored malware samples can be extracted from threat intelligence reports published by companies like Fire-Eye, F-Secure or Kaspersky (e.g. [14], [15], [16]). Such reports often include appendices with so-called Indicators of Compromise (IoCs), which consist of file hashes of malware samples or network traces identifying a specific piece of malware. In this way, IoCs are used as a reference to an unique malware sample or family.

By collecting threat intelligence reports, we can create an aggregated list of file hashes of malware samples and use it to download the samples from any large malware database.

We use an overview of APT groups by Fire-Eye [14] and the spreadsheet 'APT Groups and Operations' by Florian Roth [17] as starting point for finding threat intelligence reports. We investigate numerous sources of knowledge, but many of them fail in substantiating claims to which family or actor a sample belongs and are therefore neglected in order to maintain a high level of trustworthiness. Moreover, we only consider families which contain more than 30 malware samples.

For every trustworthy source, we collect all available hashes that are found and store them labeled by the concerning APT group. Using the hashes, the samples are downloaded from VirusTotal [18] - a malware database and analysis platform

TABLE I
CHARACTERISTICS OF THE COLLECTED DATASET

Country	APT Group	Family	Request	Download
China	APT 1		1007	405
China	APT 10	i.a. PlugX	300	244
China	APT 19	Derusbi	33	32
China	APT 21	TravNet	118	106
Russia	APT 28	'Bears'	230	214
Russia	APT 29	'Dukes'	281	281
China	APT 30		164	164
North-Korea	DarkHotel	DarkHotel	298	273
Russia	Energetic Bear	Havex	132	132
USA	Equation Group	Fannyworm	395	395
Pakistan	Gorgon Group	Div. RATs	1085	961
China	Winnti		406	387
Total			4449	3594

where any user can upload samples to - using their API. In total, we request 4,449 samples from VirusTotal, from which 3,594 unique samples are available for download. The samples allegedly originate from 12 different state-sponsored APT groups spread over 5 countries, namely China, North-Korea, Pakistan, Russia and the USA. All retrieved samples and an overview of all requested samples and the sources from which their file hashes are obtained can be found online at [].

We use VirusTotal, Cuckoo and VMRay to obtain behavioral analysis reports. Whereas Cuckoo (open source) and VMRay (paid) are sandbox solutions which perform dynamic analysis themselves, VirusTotal sends malware to different antivirus vendors and analysis platforms and combines a summary the reported information into a single report. In past experiments, presented in Section II, the authors use only the reports from Cuckoo. We also present the results using VirusTotal and VMRay reports, without disclosing the content of the reports.

B. Preprocessing

Before processing, we remove duplicate samples which are downloaded twice using different hash types (e.g. MD5 or SHA-1). Moreover, the sandbox reports are cleaned of data originating from buffers or process dumps present in reports from Cuckoo and VMRay. The reports from Cuckoo and VMRay may also contain direct links to the authors, based on information provided by antivirus vendors in case of VirusTotal or a set of Yara-rules in case of VMRay. Any such traces are removed before processing.

Finally, for every sandbox, the reports are converted to bags of words using the `CountVectorizer` from scikit-learn [19]. Each bag of words is limited to the 50,000 most common words, and contains numerical values[1]. This approach is similar to the approaches described in [8], [9]. The characteristics of the collected dataset after preprocessing can be found in Table I. We observe that some groups are over-represented and others are under-represented. In the next section we introduce two methods used to balance the dataset.

[1] Contrary to a bag of words which contains booleans, indicating that a word is present in the data, whereas the numerical variant also indicated how many times the word is present

TABLE II
AVAILABLE PERMUTATIONS

Scenario	A, B, C
Balancing	Imbalanced, Random Oversampling, Random Undersampling
Data	Cuckoo, Cuckoo*, VirusTotal, VMRay

C. Training and Test Sets

Having labeled the malware samples on two levels - namely APT level and country level - provides the possibility to come up with 3 different scenarios, which we will call scenario A, B and C. In Scenario A, we want to classify the malware samples by APT group and divide the samples over the training and test using random sampling. In Scenario B, we want to classify the malware samples by country and divide the samples over the training and test using random sampling. Scenario C focuses on finding *APT-transcending* properties for malware related to a country, and involves dividing the available samples over the training and test set in such a way that no APT group has samples in both the training set and the test set. In this way, the ML algorithm cannot derive any information about the country a malware sample belongs to by knowing to which APT group the sample belongs. Note that only samples can be used which allegedly belong to China and Russia, since these are the only two countries with malware from multiple APT groups in the dataset.

As mentioned earlier, the collected dataset is imbalanced. The imbalance may cause the classifier to be biased towards some groups. In order to overcome this phenomenon, we test oversampling and undersampling [20].

The API calls can only be extracted from the Cuckoo reports. When presenting the results, we will refer to the filtered reports which contain the API calls only as Cuckoo*.

We test each permutation between the three scenarios (A,B,C), the 3 sampling methods (unbalanced, undersampling and oversampling) and each data source, resulting in 36 unique variants of the dataset. An overview is provided in Table II.

IV. EXPERIMENTAL SETTINGS AND RESULTS

We use a smaller fully connected network than in [8] and achieve similar results. Instead of 10 layers we only use 8, maintaining the number of neurons in each layer and the ReLU activation function. In order to avoid overfitting, we use dropout and input noise with rates 0.5 and 0.2 respectively. The network is trained with the Adam optimizer. We experiment with different learning rates and find 0.001 to be the most useful value to start with. The Adam optimizer adapts the learning rate during new training epochs.

The amount of decision trees in the RFC (corresponding to [9]) is kept at 100. For all other parameters, we used the default values as implemented in scikit-learn [19].

We evaluate the results using 10-fold validation. Small variations can be observed for DNNs due to the random weight initialization. We present the accuracy averaged over 5 runs, together with the standard deviation in the following tables, ordered by the scenarios presented in the previous section. In parallel, we explore other metrics such as precision and recall. However, due to space constraints we do not present the results here. They can be accessed via the repository at [].

Table III presents the accuracy and standard deviation for Scenario A, namely classifying malware samples as belonging to different APT groups.

TABLE III
ACCURACY RESULTS SCENARIO A

	Dataset	Unbalanced	Undersampling	Oversampling
RFC	Cuckoo	**0.942** (σ: 0.007)	0.825 (σ: 0.023)	0.933 (σ: 0.006)
	Cuckoo*	**0.923** (σ: 0.009)	0.753 (σ: 0.022)	0.919 (σ: 0.006)
	VirusTotal	0.959 (σ: 0.006)	0.844 (σ: 0.021)	**0.964** (σ: 0.006)
	VMRay	0.916 (σ: 0.008)	0.775 (σ: 0.024)	**0.922** (σ: 0.010)
DNN	Cuckoo	**0.931** (σ: 0.006)	0.784 (σ: 0.018)	0.917 (σ: 0.010)
	Cuckoo*	**0.832** (σ: 0.015)	0.618 (σ: 0.047)	0.821 (σ: 0.020)
	VirusTotal	**0.969** (σ: 0.007)	0.738 (σ: 0.210)	0.966 (σ: 0.006)
	VMRay	0.871 (σ: 0.023)	0.679 (σ: 0.027)	**0.898** (σ: 0.015)

We observe that, in all cases, the RFC has higher accuracy than the neural network. This means the classifier exploits some hierarchical structures that are not learned by the DNN. We run a significance test between the results from the unbalanced, the undersampling and the oversampling and find that whenever the oversampling method performs best, the results are not statistically significantly when compared to the unbalanced dataset. This means the classifier is able to uncover a representation of the unbalanced data that can clearly separate between the different APT groups

Table IV presents the accuracy results for Scenario B, namely, classifying malware samples as belonging to different countries. As can be observed in Table I, one country may contain many APT groups. The dataset is unbalanced, with China and Pakistan having many more samples than the others. However, as can be seen in the results, oversampling or undersampling does not improve the methods. When running the same significance test described earlier, we discover the increases to be non-significant. This result suggests that even though the datasets are smaller for some classes (like North-Korea), they have distinctive traits which can describe the classes with high accuracy.

TABLE IV
ACCURACY RESULTS SCENARIO B

	Dataset	Unbalanced	Undersampling	Oversampling
RFC	Cuckoo	**0.948** (σ: 0.005)	0.910 (σ: 0.008)	0.937 (σ: 0.011)
	Cuckoo*	**0.947** (σ: 0.002)	0.902 (σ: 0.013)	0.936 (σ: 0.004)
	VirusTotal	0.960 (σ: 0.006)	0.935 (σ: 0.013)	**0.971** (σ: 0.008)
	VMRay	0.926 (σ: 0.008)	0.907 (σ: 0.016)	**0.935** (σ: 0.011)
DNN	Cuckoo	**0.945** (σ: 0.006)	0.893 (σ: 0.018)	0.924 (σ: 0.005)
	Cuckoo*	**0.894** (σ: 0.009)	0.809 (σ: 0.020)	0.878 (σ: 0.015)
	VirusTotal	0.977 (σ: 0.011)	0.943 (σ: 0.010)	**0.977** (σ: 0.004)
	VMRay	**0.932** (σ: 0.009)	0.861 (σ: 0.026)	0.930 (σ: 0.008)

Once again, we observe that RFC performs marginally better that DNNs in some cases. This means that less complex representations are sufficient for country-level authorship attribution, but can also be a consequence of the dataset. Using unstructured representations, such as the bag of words used in this case, might be better suited to algorithms which search for simpler connections, such as small trees. This discussion will be extended in Section V.

Table V presents the results for Scenario C, namely, searching for APT-transcending properties of malware by dividing the Chinese and Russian samples in such a way that no APT group has samples in both the training and test dataset. This evaluation is meant to test the classifier's power to generalize outside the APT groups it was trained with and it is based on the assumption that different APT groups share common characteristics. This assumption is not unrealistically, given that different APT groups belong to one country. However, the drop in accuracy reported in Table V invalidates this assumption when using supervised learning.

TABLE V
ACCURACY RESULTS SCENARIO C

	Dataset	Unbalanced	Undersampling	Oversampling
RFC	Cuckoo	0.427 (σ: 0.052)	0.452 (σ: 0.052)	**0.470** (σ: 0.033)
	Cuckoo*	0.566 (σ: 0.052)	0.511 (σ: 0.026)	**0.577** (σ: 0.067)
	VirusTotal	0.650 (σ: 0.056)	**0.727** (σ: 0.019)	0.664 (σ: 0.056)
	VMRay	**0.523** (σ: 0.045)	0.370 (σ: 0.052)	0.498 (σ: 0.095)
DNN	Cuckoo	0.439 (σ: 0.040)	**0.494** (σ: 0.107)	0.397 (σ: 0.030)
	Cuckoo*	0.487 (σ: 0.070)	0.562 (σ: 0.044)	**0.572** (σ: 0.050)
	VirusTotal	0.688 (σ: 0.073)	**0.721** (σ: 0.065)	0.718 (σ: 0.052)
	VMRay	**0.722** (σ: 0.026)	0.693 (σ: 0.056)	0.653 (σ: 0.031)

We can observe that the performance diminishes in all cases when information about an APT group is removed from the training set. In particular, the information from the Cuckoo reports seems to be dropping faster than others. While Cuckoo reports usually obtain higher accuracy than VMRay in previous scenarios, it seems to be tightly linked to data specific to particular APT groups and not sufficient to generalize. Contrary, VirusTotal and VMRay are able to extract some information which can help generalize between different APT groups. However, they still experience severe loss of accuracy.

Another phenomenon which can be observed in this case is that oversampling and undersampling can lead to better results. In particular, when running the significance test mentioned earlier we observe that VirusTotal performs significantly better in the undersampling case. This means that the unbalanced datasets lead to overfitting in Scenario C and poor generalization.

Lastly, we note that RFC also maintains stable performance when compared to DNNs.

V. DISCUSSION

We question several assumptions about the dataset and comment on the overall applicability of the methods presented.

Firstly, in order to reason about malware attribution, there is an implicit assumption that a malware sample is used by at most one actor, enabling the possibility to assign a sample to one APT group. However, in real-world scenarios multiple actors may use (parts of) the same sample. For example, several core elements of a piece of malware could have been bought on the dark web or stolen from other actors [4].

The results from Scenario C, presented in Table V show that, in the dataset used, not many samples which belong to different APT groups share common characteristics (otherwise we would expect a smaller drop in accuracy). Nevertheless, we advice to consider this assumption when doing attribution.

Secondly, the dataset used to benchmark the two approaches is based on claims by anti-virus companies and malware researchers. Although most claims about authorship are substantiated in corresponding malware analysis reports, they can not be proved with irrefutable evidence. We advice that, in this case, the ground truth is based on beliefs or reasons of the authors of malware analysis reports. Using such evidence for state malware attribution is not recommended.

Thirdly, the bag of words approach removes context and structure from the sandbox reports. For example, whenever a JSON sandbox report contains boolean information, the boolean values are separated from the property, limiting their information gain. Moreover, the bag of words involves a practical limit to the number of words used. When this approach is used for a long report, it can lead to loss of valuable information. In this paper we tested the approaches in literature because they seemed to work unreasonably well using these assumptions. There results presented in Section IV are consistent with the ones reported in literature, which means even simpler approaches can lead to good results and suggests adding more structure to the features used may increase the performance.

Lastly, although the classifiers reach high accuracy (and perform well for other metrics presented in the repository []), the information which can be extracted from the classifiers is far from usable. We started with the assumption that RFC are interpretable models and DNNs are only used as a comparative benchmark. However, when manually analyzing the trees generated by RFC we found them difficult to interpret and reason about. Moreover, since the bag of words approach does not involve any structure, in some cases the trees are not consistent with the opinions of human analysts.

All in all, although the techniques can reach high accuracy in the first two scenarios from Section IV, their suitability for malware attribution is questionable. The strict requirements for state-sponsored malware attribution decrease this probability even more. We remind that accusing an actor of maliciously designing and deploying malware against another actor, be it state related or not, requires unquestionable and irrefutable evidence and so, we advice caution when using supervised learning methods.

VI. Conclusions and Future Work

Technical complications, such as the lack of source code, intentionally fake traces and techniques which hinder malware analysis, cause authorship attribution of malware to be hard. Finding solutions to cope with such complications would contribute to the ability of attributing malware in a reliable manner.

With the help of the IoCs in numerous threat intelligence reports, we construct and publish a dataset of APT malware, consisting of over 3,500 samples from 12 different APT groups. This dataset can be used for future benchmarks.

Using this dataset, we verify the performance two supervised learning methods (one based on a DNN and one on a RFC) on a variety of scenarios, sampling methods and data sources.

We discover that a RFC works slightly better in most of tested cases than a DNN. Moreover, oversampling and undersampling generally do not lead to improvements of the classification results. The scenario where we split Chinese and Russian samples in such way that no APT group has samples in both training set and test set provides the insight that it is unrealistic to assume based on the used dataset that different APT groups within a country share common characteristics.

In future work, we plan to investigate if preprocessing strategies other than a bag of words could improve the classification performance, since the current approach does not take the context of the words in account. Additionally, we would like to examine whether the quality of the input data could be improved, by studying to what extent the samples are correctly ran by the different sandboxes.

Apart from that, the dataset could be expanded by adding more samples or new features for the current samples. Such expansion of the dataset enables one to examine whether new features can be found which distinguish between different APT groups or whether the classification algorithms could also distinguish between authors of criminal context malware and state-sponsored malware.

As said earlier, substantiating the classifications done by a RFCs is hard, let alone providing proofs about the fact that the outcome of a DNNs is reliable enough to act upon. Therefore, it would be of great benefit if new or improved ways regarding the interpretations and substantiating of ML algorithms could be found, since strong evidence is needed in cases involving malware authorship attribution.

References

[1] A. C. Islam, F. Yamaguchi, E. Dauber, R. E. Harang, K. Rieck, R. Greenstadt, and A. Narayanan, "When coding style survives compilation: De-anonymizing programmers from executable binaries," *2018 Network and Distributed System Security Symposium (NDSS)*, 12 2017.

[2] M. Vasilescu, L. Gheorghe, and N. Tapus, "Practical malware analysis based on sandboxing," in *2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference*, 9 2014, pp. 1–6.

[3] S. Alrabaee, P. Shirani, M. Debbabi, and L. Wang, "On the feasibility of malware authorship attribution," in *Foundations and Practice of Security*. Springer International Publishing, 2017, pp. 256–272.

[4] B. Bartholomew and J. A. Guerrero-Saade, "Wave your false flags! deception tactics muddying attribution in targeted attacks," in *Virus Bulletin Conference*, 2016. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2017/10/20114955/Bartholomew-GuerreroSaade-VB2016.pdf

[5] A. Pektaş, M. Eriş, and T. Acarman, "Proposal of n-gram based algorithm for malware classification," in *SECURWARE 2011 - 5th International Conference on Emerging Security Information, Systems and Technologies*, 01 2011, pp. 14–18. [Online]. Available: https://pdfs.semanticscholar.org/cdcf/c68ce8b6eaaf0d8964dffd99105c566b6027.pdf

[6] H. Rays. [Online]. Available: https://www.hex-rays.com/products/ida/

[7] O. E. David and N. S. Netanyahu, "Deepsign: Deep learning for automatic malware signature generation and classification," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 7 2015, pp. 1–8.

[8] I. Rosenberg, G. Sicard, and E. David, “End-to-end deep neural networks and transfer learning for automatic analysis of nation-state malware,” *Entropy*, vol. 20, no. 5, p. 390, 5 2018.

[9] N. Aman, Y. Saleem, F. H. Abbasi, and F. Shahzad, “A hybrid approach for malware family classification,” in *Applications and Techniques in Information Security*. Springer, 2017, pp. 169–180. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-10-5421-1_14

[10] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, “Dl4md: A deep learning framework for intelligent malware detection,” in *Proceedings of the International Conference on Data Mining (DMIN)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016, p. 61.

[11] W. Huang and J. W. Stokes, “Mtnet: A multi-task neural network for dynamic malware classification.” Springer, 7 2016, pp. 399–418. [Online]. Available: https://www.microsoft.com/en-us/research/publication/mtnet-multi-task-neural-network-dynamic-malware-classification/

[12] K. Rieck, P. Trinius, C. Willems, and T. Holz, “Automatic analysis of malware behavior using machine learning,” *J. Comput. Secur.*, vol. 19, no. 4, pp. 639–668, 12 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2011216.2011217

[13] A. Pfeffer, B. E. Ruttenberg, L. Kellogg, M. Howard, C. Call, A. O’Connor, G. Takata, S. N. Reilly, T. Patten, J. Taylor, R. Hall, A. Lakhotia, C. Miles, D. Scofield, and J. Frank, “Artificial intelligence based malware analysis,” *arXiv preprint*, 2017.

[14] “Advanced persistent threat groups.” [Online]. Available: https://www.fireeye.com/current-threats/apt-groups.html

[15] F-Secure, “Whitepaper: The dukes: 7 years of russian cyberespionage,” 2015. [Online]. Available: https://www.f-secure.com/documents/996508/1030745/dukes_whitepaper.pdf

[16] K. Lab, “Whitepaper: The nettraveler (aka ’travnet’),” 2013. [Online]. Available: https://kasperskycontenthub.com/wp-content/uploads/sites/43/vlpdfs/kaspersky-the-net-traveler-part1-final.pdf

[17] F. Roth, “Apt groups and operations.” [Online]. Available: http://apt.threattracking.com/

[18] “Virustotal.” [Online]. Available: https://www.virustotal.com/

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[20] T. Mitchell, B. Buchanan, G. DeJong, T. Dietterich, P. Rosenbloom, and A. Waibel, “Machine learning,” *Annual review of computer science*, vol. 4, no. 1, pp. 417–433, 1990.