Comparing Machine Learning Models for Malware Detection

John Williams
Department of Computer Science
University of South Carolina Upstate
Spartanburg, South Carolina, USA
jrw9@uscupstate.edu

ABSTRACT

The threat of cybersecurity breaches and attacks continue to grow. Threats like ransomware continue to affect all sectors of business and can be extremely costly for businesses to remediate. One of the many ways to protect against cybersecurity threats is to implement high fidelity malware detection. Cybersecurity companies develop and sell anti-virus agents to protect against malicious code and executables from running on devices. To further aid the detection of malware, researchers and cybersecurity companies are tapping into machine learning techniques. This paper aims to measure the effectiveness of malware detection using mean absolute error metrics between two popular machine learning models, support vector model and random forest.

Keywords

ML, Machine Learning, SVM, Support Vector Model, RF, Random Forest, Static Analysis, Botnet, DDOS, Distributed Denial of Service, Indicators of compromise, IoC, Portable Executable Headers, PE Headers.

1. INTRODUCTION

Malware detection continues to be an important part of cybersecurity and companies are investing in building anti-virus agents to detect and stop malware. This is important to prevent destructive processes from taking place within a business's infrastructure.

Malware is a general term used to broadly describe programs or code that negatively impacts the device or data. Malware is typically categorized by the behavior of the executing process. Some descriptions of malware include ransomware, adware, worms, viruses. [1] Likewise, malware is used to accomplish different things. Malware could be used to steal, delete, or encrypt data. It can also be used to take over a device or devices to facilitate more attacks (botnet) or deny services (DDOS). [2] See the graph below:

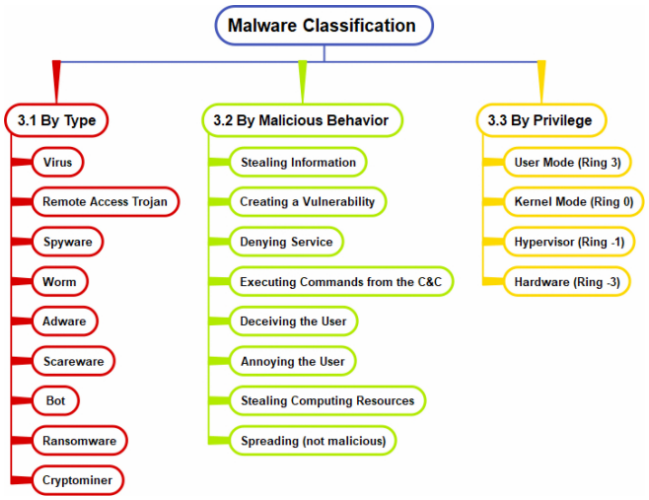


Figure 1. Malware Classification [5]

The threat landscape continues to change and threat actors are continuing to develop malware to avoid detection. It is a never-ending race between security researchers and threat actors. Both security researchers and threat actors are leveraging machine learning to develop their techniques to detect and avoid malware. Traditional detection techniques depend on classifying malicious software and using signatures to detect and block the malware before it executes. This is known as static analysis. Static analysis depends on large datasets and reputation measurements from many different vendors to work well. For example, Cisco has a threat intelligence team, Cisco Talos, that works on identifying and publishing new threats. Once a threat intelligence group

identifies files or URLs that are malicious, they publish the data as indicators of compromise (IoC). Companies that support anti-virus agents, like Microsoft Defender, take the IoCs and add them to their signature database so the anti-virus will block the file or URL when it is identified on the device.

Dynamic analysis consists of analyzing the behavior of running code or software and determining if the behavior is malicious or not. The files are executed inside of sandbox environment and the behavior is monitored. Running the file from inside a sandbox prevents damage to the device. Dynamic analysis would when look for behavior that can be attributed to malicious software. Malicious behavior could be registry changes, sending beacons, remotely executing PowerShell or curl commands.

Dynamic and static analysis have different benefits. Dynamic analysis, while resource intensive, is more effective against malware that may be obfuscated. Static analysis may miss obfuscated malware, but it is very fast and resource efficient. [3] These analysis techniques are running on anti-virus software on the device. So, resources for detection are limited to the resources the device can provide to the anti-virus software without disrupting normal device usage.

Machine learning can be applied to both static and dynamic analysis. Machine learning models examine the extracted features of the malware and classify it. Supervised machine learning techniques depend on malware detection experts to tell the model what features to examine. Unsupervised models do not depend on predefined features, instead the models will learn the features and determine what should be examined. A neural network is an example of an unsupervised model that can extract features from raw inputs. [4]

For static analysis, SVM and RF have been used successfully to detect malware.[1]

This paper compares the effectiveness of SVM and RF models. Specifically mean absolute error metric is used to assess which model was more effective classifying executable files as malicious or benign. The models will train and test on PE Headers from the executable file dataset. The amount of time the algorithms needs to finish training is also measured and compared.

2. LITERATURE REVIEW

Literature reviewed for this paper consists of academic papers surveying malware detection with machine learning, PE headers, and malware detection using PE headers.

Malware is increasingly prevalent. [7] It is used for many different reasons, but the common thread is that malware is malicious in nature and attempts to do something negative or destructive. [6] These negative actions can be monitored and stopped. For example, if a malicious executable is downloaded onto a device by an unaware end user, then the anti-virus on the machine would be able to scan is and identify the MD5 hash. The anti-virus can then check the MD5 hash and if it is determined to belong to a malicious executable, then it will be blocked. This is one way static analysis works and the process is known as classification. The anti-virus classified the executable using the MD5 hash. MD5 hashes are just one of the many features that be extracted and analyzed. The method described above can be thwarted by more sophisticated malware. Threat actors are now obfuscating their code and making it much more difficult for anti-virus detection to catch it. [3] Detection engines (anti-virus) were further developed to analyze the behavior of malware. By monitoring the behavior, anti-virus might identify suspicious operations and stop the malware. Analyzing the malware normally takes place in a sandbox environment to keep the device safe from harm. With either solution, or a combination of both, detection is dependent on properly classifying features of the executables. One feature used is portable executable headers (PE Headers).

PE headers describe the structure of executable files. [8] The PE Header is made of the following concepts:

Attribute Certificate, Data/Time Stamp, File Pointer, Linker, Object File, Reserved, Relative Virtual Address, section, and Virtual Address.

PE Headers will most likely have these concepts but may not. There are also many other optional sections of the file format. In a relevant paper, the authors not that they focused on the first three sections in the PE Header. They extracted those features and ran their classification models using that data. [9] This methodology is dependent on some domain knowledge.

Machine learning models can be leveraged to make static analysis faster. This is because machine learning can quickly analysis lots of extracted features from the executables, like PE Headers, and use those features to classify the executable as malicious or not. The process broadly consists of gathering data samples. These data samples are just a large file of executables. Features of the executables would then be extracted using specialized software, like PeView. Next, the static features would be represented in a format, such as a matrix or list. Finally, the machine learning module would run and classify the features. See below:

Support Vector model (SVM) is a supervised machine learning algorithm. Originating from Vapnik's VC Theory, SVM has been used in many areas as a learning technique. Some of those areas include financial forecasting, marketing, estimating yields in manufacturing, facial recognition, among others. [10] SVM will use data points with either a linear or nonlinear hyperplane. A prediction model can be built as training data points are sent through the model. The result is a graph with a hyperplane separating datapoints. The classification is dependent on which side on the hyperplane the datapoint is on.

Random Forests is an ensemble learning model that using multiple decisions trees training and it returns the mean of all the decisions trees. Random forests train on various subsets of the training data and aggregate the results using a method called bagging. One of the goals with random forests was to avoid over-fitting. Decisions Trees are prone to over-fitting which leads to less accurate classification. Decisions Trees are simple in practice and become more accurate when combined as a random forest. Random Forests also benefit from being extremely simple to setup once classification decisions are made. It may require some domain knowledge to set up the classification decisions.

Both learning models are capable of training and classifying executables using PE Headers. Two papers reviewed highlight the success of the models with malware detection. [1] One study, was able to classify malware with an accuracy of 99.49% using SVM and MR.[11] This accuracy is extremely close to 100% which may indicate that it would be the better choice out of the two algorithms selected. However, random forest models have also had very high accuracy reported.[1] Time duration for training was a metric that was not reported on or written about in the

survey papers. It will be good to review the amount of time it takes to complete training for each model and see which is faster.

3. METHODOGY

Python modules will be used to work with the dataset. The modules used are Pandas, Sklearn, Numpy. Jupyter Notebooks will also be used to present the implementation workflow in a readable and repeatable way. Pandas will be used to extract and manipulate the data as needed. Numpy will be used to present the data in different forms. An example would be a histogram visualizing feature values. Sklearn will be used to train and run learning models. Sklearn can run Random Forest and SVM models.

The models for this paper will analyze PE Header data that was extracted from a sample dataset of executable software. Accuracy and speed of execution will be compared to determine which model is better for the classification task. The classification task is to evaluate executables and decide if the executable is malware or not. While the dataset, PE Headers, is already provided, data cleaning, preparation, and feature selection will still be needed. Below is an example figure of this process:

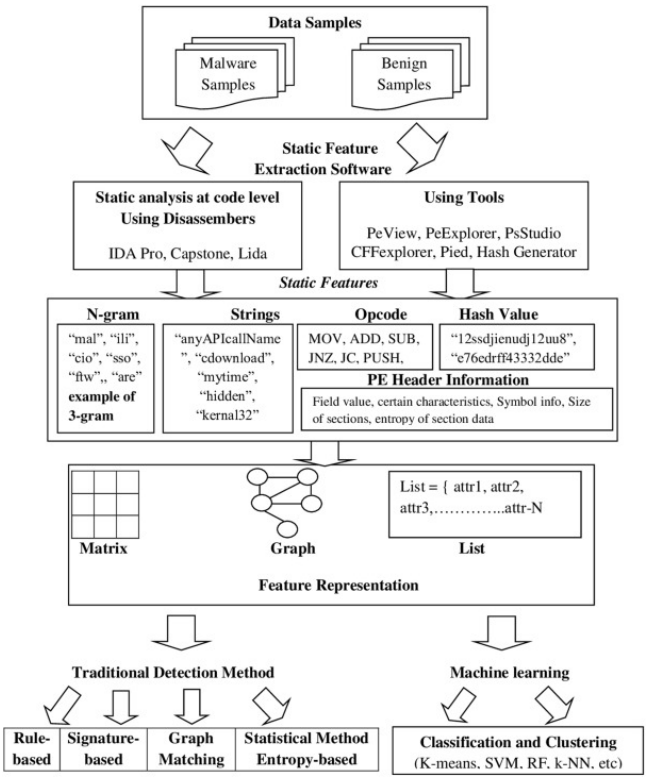


Figure 2. Malware Data Extraction and Analysis[1]

The first step will be to analyze the dataset that is being used for this paper. The PE Header data will need to be identified and extracted before any kind of training can be done. The dataset has thousands of executables and their respective data. PE Headers have multiple subsets of data fields that will be extracted and used for analysis. These data fields will be known as features.

While the data, features, is being extracted, missing data will be identified and will need to be handled. This process is known as data cleaning. The missed data can either be ignored or manipulated. The entry can be ignored by deleting it. Alternatively, the missing data can be filled with existing data from the dataset. This is done by filling in the missing data with the most common value found in the dataset for that specific data field. The process of manipulation will not be used in this paper. Entries missing the selected PE Header features will be ignored.

The dataset will then be scaled using normalization. This will be done because the data values vary greatly. Normalized data will help the models perform better and may lead to more accurate results. The results with the data not normalized and normalized will be compared. The time it takes to normalize the data will be measured.

The models will then be trained, tested, and compared using evaluation metrics.

3.1 Support Vector Model

Support Vector Models support supervised learning tasks and can classify datasets. The model will be used to classify files, using their extracted data, as malware or benign. Multiple kernel functions can be used for learning tasks. A downside to SVMs is that they need high compute resources to run efficiently. A way to avoid this high compute requirement is to use the linear SVM kernel. Multiple kernels can be used with SVM. The Radial Basis Function (RFB) kernel was used. It is the default kernel used with the Sklearn SVM module import. Below is the RFB kernel mathematical function:

$$\exp(-\gamma\|x - x'\|^2)$$

According to the Sklearn documentation, gamma and C are two parameters that need careful consideration. The C parameter controls the classification strictness. Meaning it decides how hard the model needs to work to classify the training examples accurately. Gamma controls the weight any individual training example has. [12]

3.2 Random Forest

Random Forest is a supervised learning model designed for classification. It takes decision trees and averages them to create a better prediction. This methodology of taking the average of many in called an ensemble. The averaging is a benefit because it prevents overfitting. Random Forests do however take more compute power than a decision tree model. One of the reasons why Random Forests work well is the added randomness that comes with running many individual decision trees. It is important to make sure the decisions trees are random to drive different outcomes. This means feeding the decision trees different variations of features. The Sklearn Random Forest model will split features automatically among n-defined max_features parameter.

3.3 Algorithm Comparison

Both models may result in high compute needs and training time. The benefit is a more accurate classification model where the malicious files are classified accurately.

SVM will be significantly slower than Random Forest but it may be worth it if the classification task is more accurate. Similarly, if the Random Forest classifies the dataset with greater accuracy and quicker than SVM, Random Forest will be the better choice.

One other significant difference between the models is the mentioning of dimensional spaces. Sklearn documentation notes that SVM is effective with dataset that include high dimensional spaces. Dimensionality refers to the number of features a dataset might have. One way to measure dimensionality would be to see if features exceed observations. An example of this would be one person compared to the possible combinations of genes in that one person. The dataset in this example does not meet that high dimensionality definition.

4. IMPLEMENTATION

The implementation of the models is done in Google Colaboratory. The following python modules are used: Sklearn, Pandas, Time, Numpy, Matplotlib. Sklearn is a python module designed for machine learning. The majority of the model will be ran using Sklearn. Pandas is a data analysis module. It is used to load the data and perform any cleaning, formatting, Numpy is added as additional module to work with Pandas. Matplotlib is used to plot data and visualize it.

4.1 Dataset

The dataset for this paper is from a GitHub account, *PactPublishing/Mastering-Machine-Learning-for-Penetration-Testing*. The dataset is a CSV consisting of 138,047 samples of data extracted from executable files. Below is a list of the data fields, presented in the CSV as column headers.

```
'Name', 'md5', 'Machine', 'SizeOfOptionalHeader', 'Characteristics',
'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode',
'SizeOfInitializedData', 'SizeOfUninitializedData',
'AddressOfEntryPoint', 'BaseOfCode', 'BaseOfData', 'ImageBase',
'SectionAlignment', 'FileAlignment', 'MajorOperatingSystemVersion',
'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfImage',
'SizeOfHeaders', 'CheckSum', 'Subsystem', 'DllCharacteristics',
'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes', 'SectionsNb',
'SectionsMeanEntropy', 'SectionsMinEntropy', 'SectionsMaxEntropy',
'SectionsMeanRawsize', 'SectionsMinRawsize', 'SectionMaxRawsize',
'SectionsMeanVirtualsize', 'SectionsMinVirtualsize',
'SectionMaxVirtualsize', 'ImportsNbDLL', 'ImportsNb',
'ImportsNbOrdinal', 'ExportNb', 'ResourcesNb', 'ResourcesMeanEntropy',
'ResourcesMinEntropy', 'ResourcesMaxEntropy', 'ResourcesMeanSize',
'ResourcesMinSize', 'ResourcesMaxSize', 'LoadConfigurationSize',
'VersionInformationSize', 'legitimate'],
```

Figure 3. File Column Headers/Features

These column headers are used a features in the learning models. Fortunately, domain knowledge is not required to complete the machine learning models and evalute their success despite. Histograms will be generated for each feature and malicious files will be compared with bengin to see if there are any visual patterns. Addtionally, the feature values have exremely large ranges which may complicate the model training. Scaling the features may be done if model performance is low. The “Name”, “md5”, “legitimate” features will be removed from the dataset. The “legitimate” feature will be used for labeling. If the value in the “legitimate” cell for a file is 1, the feature is benign. If it is 0, the feature is malware. This binary output will be used to measure the predication accuarcy of the models. The predicted “legitimate” values will be compared with the actual.

Without scaling the dataset, SVM performed poorly during intial testing. The dataset contained large varitions in values which contributed to the poor performance. Scaling was done using the preprocessing.StandardScaler Sklearn module. Standard Scaling removes the mean in the feature and then divides feature values by their standard deviation.[13]

5. EXPERIMENTAL SETUP

The setup to evaluate the models involves data analysis, training the model, using the model against a test subset of data, and evaluating the results.

For data analysis, the data was loaded into a Pandas Dataframe from a CSV. The columns, which represent the features, were evaluated. The dataset consisted of 57 features. Three features, Name, md5, and legitimate were removed. Name and md5 needed to be removed because the model would not run because the values were not integers. The rest of the columns were displayed in Figure 3. The dataset consists of 138,047 file samples. See the makeup below:

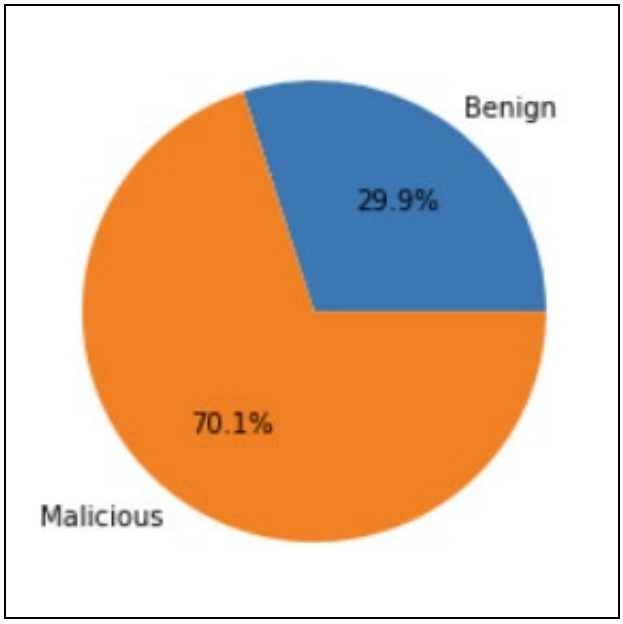


Figure 4. Malicious and Benign Pie Chart

70.1 percent of the files are malicious and 29.9 are benign. This information is provided by looking at the legitimate column in the dataset. The legitimate column is removed from the training and testing set because the column is used to measure the accuracy of classification tasks. A function from Sklearn called train_test_split was used to divide the dataset in four subsets. A training set for X, validation set for x, a training set for y, and a validation set for y. The fit function from Sklearn will be run for both libraries using the subsets from train_test_split. See below:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error


forest_model = RandomForestRegressor(random_state=1)


forest_model.fit(train_X, train_y)
```

Figure 5. Imports and Fit Functions.

A similar import and fit function is called for the SVM function. Once the models are fitted, the model will be used to predict whether the valuation data subset is malware or not.

Accuracy in predicting whether the file samples are malware or not will be measured using a Python Module from Sklearn called mean absolute error. It measures the mean absolute error of the prediction by taking the forth subset of data, valutation y, and the prediction values from vaulation x as parameters. It may be more accurate to call valuation y the true values. Mean aboslute error is the measuring the amount of values that were incorrect. So if 85% of the predictions were correct, the mean absolute error would be 15%. See and example below:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error


y_true = [1,0,1,0,1,0,0,0,1]
y_prediction = [0,1,0,0,1,1,0,1,1]
mean_absolute_error(y_true, y_prediction)

0.5555555555555556
```

Figure 6. Mean Error Example

In this example, the mean error is 0.55 or 55%.

A second metric for measuring the algorithms will be compute time. The time it takes to complete the training and prediction is very important in machine learning. This study will use the Time Python module to evaluate the compute time it takes to complete the training and prediction. CPU power is normalized by using Google Cobalortory.

6. RESULTS ANALYSIS

The initial results of this study showed that the Random Forest model was better suited for the classification task. The RF model was faster and more accurate than SVM.

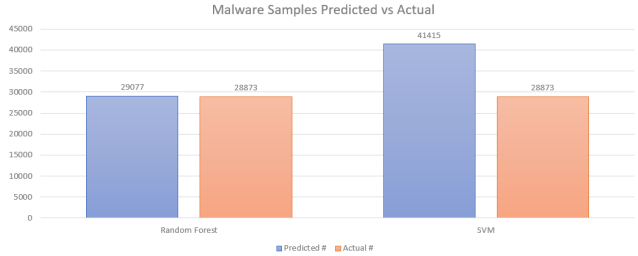


Figure 7. Initial Predicted verses Actuall Malware

These initial results were poor and it appears that the sample size for each were not the same. The RF model test sample size was not equal between the test samples and was not scaled. Below is the initial accuarcy.

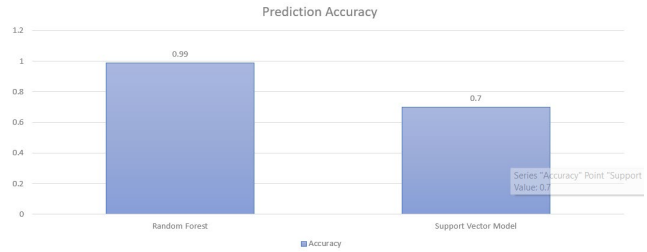


Figure 8. Initial Accuracy

The dataset was modified to ensure better testing. The test sample was set to 10% of the total dataset and was scaled. The final results showed that SVM was more accuracte but significantly slower than RF.

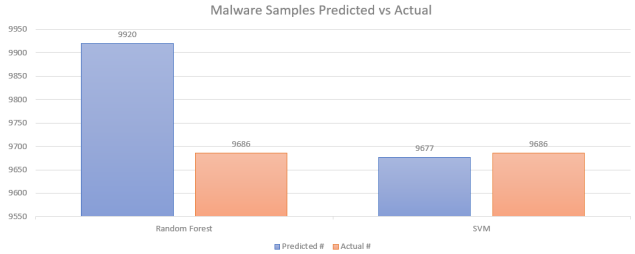


Figure 9. Final Predicted verses Actual Malware

The accuracy of the SVM model improved greatly. See below.

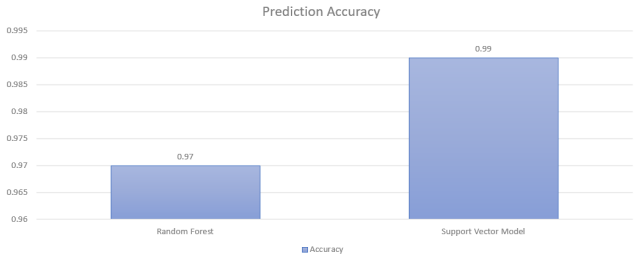
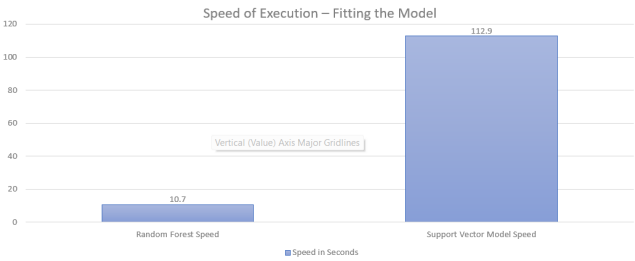


Figure 10. Final Accuracy

Finally, the speed of the RF was significantly faster than the SVM. Specifically, it was about 8 times faster than SVM.



7. CONCLUSION

To conclude, it was found that the dataset transformation have a drastic impact on performance of the models. Some models like the RF model perform well without scaled data but SVM performance suffers without scaled data. With malware identification being so important, it would probably be in the best interest to design a model that is as accuracy as possible. Because of that, the SVM model would most likely be best suited for the classification task. If speed is more important than accuracy, then

RF would be the best model to work with. This is because the speed of the RF model was 8 times faster than the SVM model.

8. REFERENCES

[1] Jagsir Singh, Jaswinder Singh, A survey on machine learning-based malware detection in executable files, Journal of Systems Architecture, Volume 112, 2021, 101861, ISSN 1383-7621, DOI: https://doi.org/10.1016/j.sysarc.2020.101861.

[2] Damodaran, A., Troia, F.D., Visaggio, C.A. et al. A comparison of static, dynamic, and hybrid analysis for malware detection. J Comput Virol Hack Tech 13, 1–12 (2017). DOI: https://doi.org/10.1007/s11416-015-0261-z

[3] Singh J., Singh J. Challenges of malware analysis: Obfuscation techniques Int. J. Inf. Secur. Sci., 7 (3) (2018)

[4] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, C.K. Nicholas, Malware detection by eating a whole EXE, The Workshops of the the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018 (2018), pp. 268-276

[5] Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2020, September 30). Dynamic Malware Analysis in the Modern Era—A State of the Art Survey. ACM Computing Surveys, 52(5), 1–48. https://doi.org/10.1145/3329786

[6] Han W., Xue J., Wang Y., Huang L., Kong Z. MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics Comput. Secur., 83 (2019), pp. 208-233, 10.1016/j.cose.2019.02.007

[7] AV-TEST K. Malware statistics and trends report, AV-TEST (2020) https://www.avtest.org/en/statistics/malware

[8] Microsoft Docs, 2022, https://docs.microsoft.com/en-us/windows/win32/debug/pe-format

[9] Edward Raff, Jared Sylvester, Charles Nicholas, Learning the PE Header, Malware Detection with Minimal Domain Knowledge, 2017, https://doi.org/10.1145/3128572.3140442

[10] Kyung-Shik Shin, Taik Soo Lee1 , Hyun-jung Kim, An application of support vector machines in bankruptcy prediction model, Expert Systems with Applications 28 (2005) 127–135, https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.2804&rep=rep1&type=pdf

[11] Huda S., Islam R., Abawajy J., Yearwood J., A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection, (2018), 10.1016/j.future.2017.12.037

[12] Sklearn Documentation for SVM. https://scikit-learn.org/stable/modules/svm.html

[13] Sklearn Documentation for StandarScaler. https://scikit - learn.org/stable/modules/preprocessing.html#preprocessing-scaler