T. C.

ANKARA UNIVERSITY

ENGINEERING FACULTY

COMPUTER ENGINEERING DEPARTMENT

AN INTRUSION DETECTION SYSTEM FOR NEW CYBER ATTACKS

GRADUATION PROJECT REPORT

Ayça Nur VANLI

17290128

SUPERVISOR

Asst. Prof. Ömer Özgür TANRIÖVER

ANKARA – 2021

08/01/2021

ABSTRACT

This is a report for the Ankara University Computer Engineering Department about my undergraduate research project. In this paper, a host-based intrusion detection system (IDS) and a network IDS researched and tried to implement via machine learning solutions.

TABLE OF CONTENTS

TABLE OF FIGURES & TABLES

1. INTRODUCTION

In order to talk about implementation of machine learning solutions to the malwares, the malware types and their distinctions must be clarified.

Malware is the merged version of the words malicious and software. The software can be code, scripts, active content and other software design to gain access and corrupt or abuse the victim over the cyber world. Multiple new malwares have been creating rapidly in the past couple of years. The malwares are not just harmful to the citizens, but they illustrate massive damages to the companies and countries over the decades such as Stuxnet and WannaCry.

We can classify malwares based on their behaviour of execution. We can list them as below [1]:

- Virus: Viruses are self-replicating malwares. They are passive and they spread by being transferred through vectors such as files, media or network.
- Worm: Worms are self-replicating malwares. They are active and they can spread over the network.
- Trojan Horse: Trojan Horse is an imposter software that looks like harmless but they disguise their intend and show is after being executed.
- Spyware: Spyware is a malware that aims to monitor, store and later investigate the victim's actions and abuse it to gain benefits.
- Adware: Adware is the malware that specifically created for the advertisements.
- Root Kit: Root Kit is a malware that uses their tools to disguise themselves from being detecting.
- Bots: Bots are a malware used for abusing a victim's bandwidth, memory or CPU to achieve a high impact on the attacks. They create botnets. They are also referred as zombies.
- Ransomware: Ransomware is a malware that detains the victim and releases after a ransom is paid.

Stuxnet were a worm that effects SCADA[1] systems and aims to gain control [2]. This worm had been used over the Islamic Republic of Iran's nuclear research and development facilities. The malware first detected in 2010. WannaCry is a ransomware that has been attacking the victim's files. It is encrypting the victim's files and demand ransom to release them. FedEx and Renault were two of the victims [3].

Intrusion Detection System (IDS) is a commonly used device for cyber security solutions. The aim of this device is to monitor the network traffic and alert the administrator for any malicious actions in the traffic. We can classify three types of IDS based on their input data which are Network IDS (NIDS) that takes input from the

[1] SCADA is an acronym for Supervisory Control And Data Acquisition which stores and examine the real time data.

network packets, Host-based IDS (HIDS) that takes input from the specific host's traffic, and Application IDS that takes input as high-risk applications traffic [4].

With the soaring expansion of the Internet over the years, every network has been facing a tremendous amount of traffic with an exponential growth rate. For this reason, it has become inevitable to abandon the traditional network IDS. This traditional approach to detecting an intrusion is called signature-based IDS where the IDS's database tries to match the actions with the already detected attacks. Hence, the industry started to consider a more intellectual way to deal with their big data and they started to improve the efficiency of their IDS. When we are considering big data like network traffic, Machine Learning (ML) algorithms can be a remedy o handle them. Therefore, machine learning algorithms can be designed to detect the bad intended traffic in a network. So, the other approach to detect an intrusion is called the anomaly-based approach where the IDS is capable to determine the obscure attacks based on abnormal behaviors.

2. HOST-BASED IDS

I tried to implement a HIDS for the Windows operating systems (OS). In the initial step of my project, I started with malware analysis with the specification of Windows malicious execution analysis.

I used a Microsoft 10 virtual machine on VMware Workstation 15 Pro for building a secure environment to examination of malwares without infecting my host computer and network. Then, I downloaded Python 3.7.5. I choose a year older version of Python because some of the python libraries' functionalities were not implemented in the Python 3.9.1. which was the latest version available. Then I disconnect my virtual machine from the host's network and I took a snapshot to reverse back to the uninfected phase of the machine after I did my dynamic analyses of the malwares.

For the data sets, I collected malware executables from the Zoo [5] malware data set on GitHub. This data set includes a variety of malwares. Then I collected benign samples that are in from my computer and various sources.

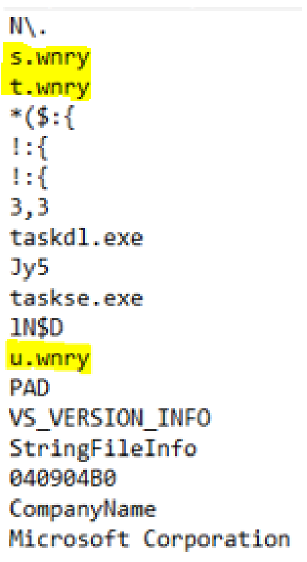
I collected my executables under the C driver with running the script below in PowerShell ISE.

2.1. Static Analysis

I choose ransomware WannaCry to analysis. I used two different tools for static analysis which are String Tool and PeStudio.

2.1.1. String Tool

After the application of the string tool to the ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe, the WannaCry can be seen with the name of the ".wnry" in the string.log file. This can be seen in the figure below.

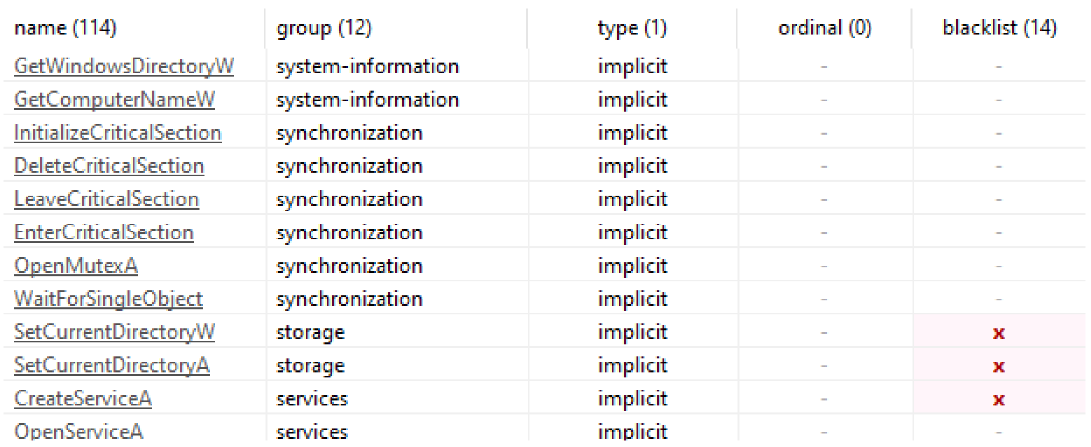


```
N\.
s.wnry
t.wnry
*($:{
!:{
!:{
3,3
taskdl.exe
Jy5
taskse.exe
1N$D
u.wnry
PAD
VS_VERSION_INFO
StringFileInfo
040904B0
CompanyName
Microsoft Corporation
```

*Figure 1: *.wnry files*

2.1.2. PeStudio

I used PeStudio to see the various aspect of the executable such as inspecting .dll calls and the whether they are blacklisted or not. This can be seen in the figure below.



name (114)	group (12)	type (1)	ordinal (0)	blacklist (14)
GetWindowsDirectoryW	system-information	implicit	-	-
GetComputerNameW	system-information	implicit	-	-
InitializeCriticalSection	synchronization	implicit	-	-
DeleteCriticalSection	synchronization	implicit	-	-
LeaveCriticalSection	synchronization	implicit	-	-
EnterCriticalSection	synchronization	implicit	-	-
OpenMutexA	synchronization	implicit	-	-
WaitForSingleObject	synchronization	implicit	-	-
SetCurrentDirectoryW	storage	implicit	-	x
SetCurrentDirectoryA	storage	implicit	-	x
CreateServiceA	services	implicit	-	x
OpenServiceA	services	implicit	-	-

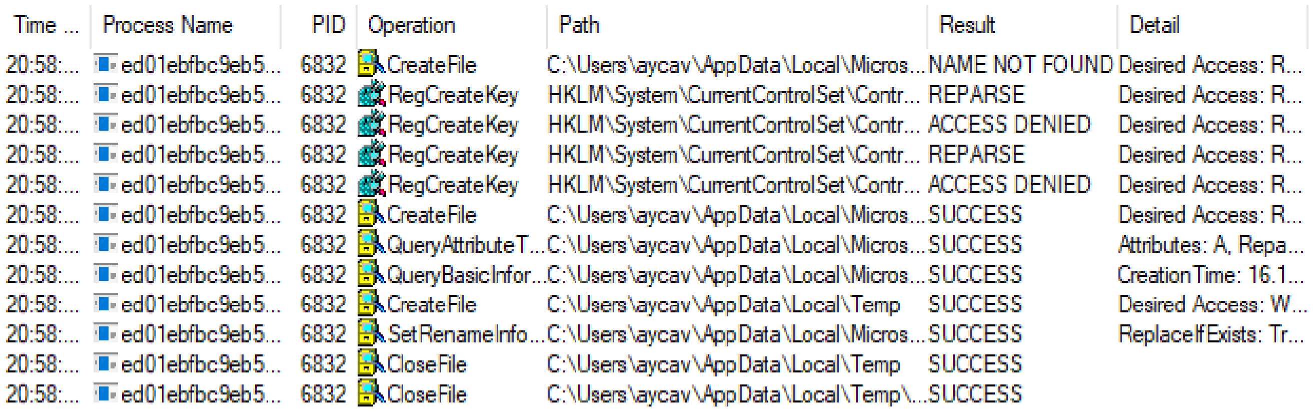
Figure 2: PeStudio

2.2. Dynamic Analysis

After the static analysis of the WannaCry, I proceed with the dynamic analysis of the executable of WannaCry.

2.2.1. Process Monitor

I used Process Monitor to see each action of the ransomware. This can be seen in the figure below.



Time ...	Process Name	PID	Operation	Path	Result	Detail
20:58:...	ed01ebfbc9eb5...	6832	CreateFile	C:\Users\aycav\AppData\Local\Micros...	NAME NOT FOUND	Desired Access: R...
20:58:...	ed01ebfbc9eb5...	6832	RegCreateKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
20:58:...	ed01ebfbc9eb5...	6832	RegCreateKey	HKLM\System\CurrentControlSet\Contr...	ACCESS DENIED	Desired Access: R...
20:58:...	ed01ebfbc9eb5...	6832	RegCreateKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access: R...
20:58:...	ed01ebfbc9eb5...	6832	RegCreateKey	HKLM\System\CurrentControlSet\Contr...	ACCESS DENIED	Desired Access: R...
20:58:...	ed01ebfbc9eb5...	6832	CreateFile	C:\Users\aycav\AppData\Local\Micros...	SUCCESS	Desired Access: R...
20:58:...	ed01ebfbc9eb5...	6832	QueryAttributeT...	C:\Users\aycav\AppData\Local\Micros...	SUCCESS	Attributes: A, Repa...
20:58:...	ed01ebfbc9eb5...	6832	QueryBasicInfor...	C:\Users\aycav\AppData\Local\Micros...	SUCCESS	CreationTime: 16.1...
20:58:...	ed01ebfbc9eb5...	6832	CreateFile	C:\Users\aycav\AppData\Local\Temp	SUCCESS	Desired Access: W...
20:58:...	ed01ebfbc9eb5...	6832	SetRenameInfo...	C:\Users\aycav\AppData\Local\Micros...	SUCCESS	ReplaceIfExists: Tr...
20:58:...	ed01ebfbc9eb5...	6832	CloseFile	C:\Users\aycav\AppData\Local\Temp	SUCCESS	
20:58:...	ed01ebfbc9eb5...	6832	CloseFile	C:\Users\aycav\AppData\Local\Temp\...	SUCCESS	

Figure 3: Process Monitor

Also, I captured the desktop prior and during the dynamic analysis of the WannaCry.

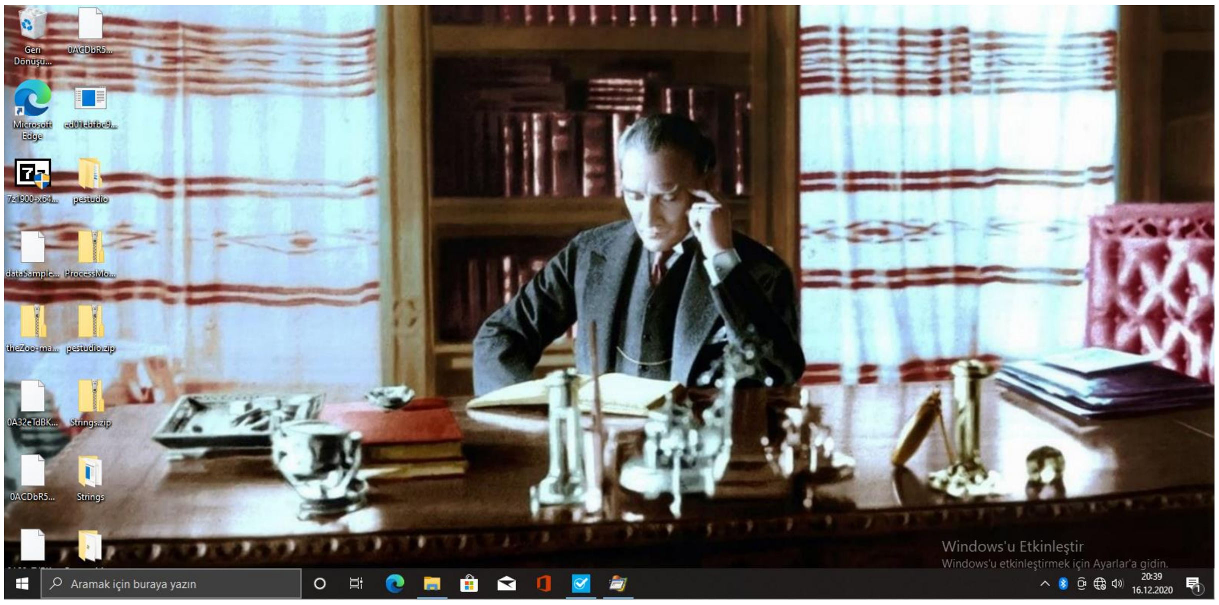
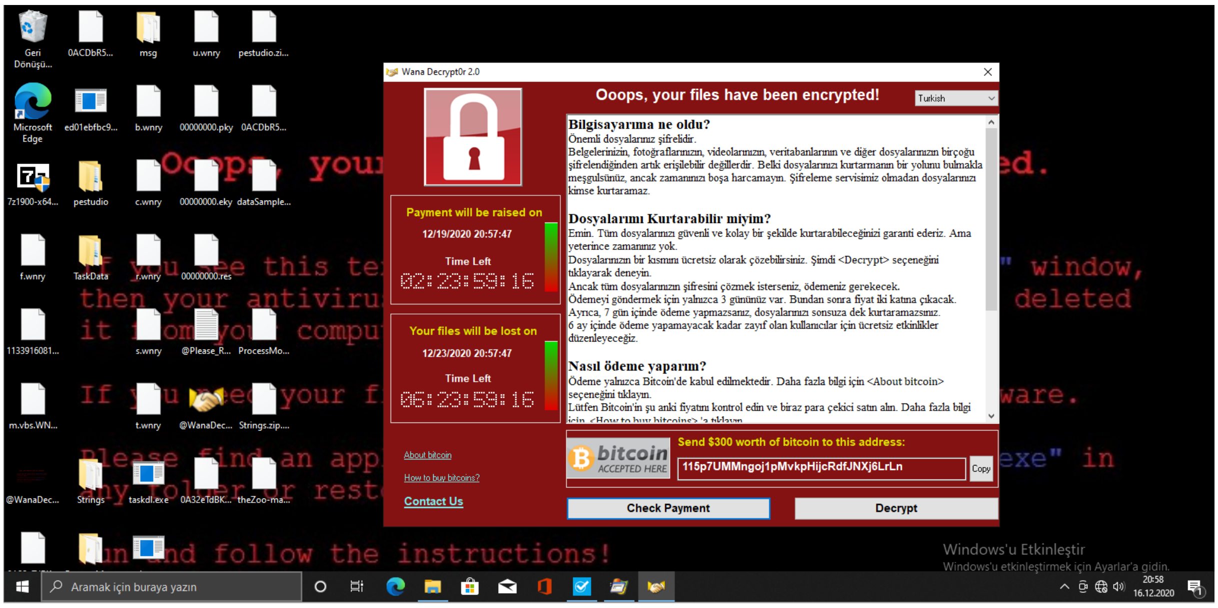


Figure 4: Prior to WannaCry



Ooops, your files have been encrypted!

Payment will be raised on
12/19/2020 20:57:47
Time Left
02:23:59:16

Your files will be lost on
12/23/2020 20:57:47
Time Left
06:23:59:16

Bilgisayarıma ne oldu?
Önemli dosyalarınız şifrelidir.
Belgelerinizin, fotoğraflarınızın, videolarınızın, veritabanlarının ve diğer dosyalarınızın birçoğu şifrelendiğinden artık erişilebilir değillerdir. Belki dosyalarınızı kurtarmanın bir yolunu bulmakla meşgulsünüz, ancak zamanınızı boşa harcamayın. Şifreleme servisimiz olmadan dosyalarınızı kimse kurtaramaz.

Dosyalarımı Kurtarabilir miyim?
Emin. Tüm dosyalarınızı güvenli ve kolay bir şekilde kurtarabileceğinizi garanti ederiz. Ama yeterince zamanınız yok.
Dosyalarınızın bir kısmını ücretsiz olarak çözebilirsiniz. Şimdi <Decrypt> seçeneğini tıklayarak deneyin.
Ancak tüm dosyalarınızın şifresini çözmek isterseniz, ödemeniz gerekecek.
Ödemeyi göndermek için yalnızca 3 gününüz var. Bundan sonra fiyat iki katına çıkacak.
Ayrıca, 7 gün içinde ödeme yapmazsanız, dosyalarınızı sonsuza dek kurtaramazsınız.
6 ay içinde ödeme yapamayacak kadar zayıf olan kullanıcılar için ücretsiz etkinlikler düzenleyeceğiz.

Nasıl ödeme yaparım?
Ödeme yalnızca Bitcoin'de kabul edilmektedir. Daha fazla bilgi için <About bitcoin> seçeneğini tıklayın.
Lütfen Bitcoin'in şu anki fiyatını kontrol edin ve biraz para çekici satın alın. Daha fazla bilgi için <How to buy bitcoins> 'a tıklayın

Send $300 worth of bitcoin to this address:
115p7UMMngoj1pMvkpHijcRdfJNXj6LrLn

Figure 5: During WannaCry

2.3. PE Headers

One of the most effective ways to detect malwares are using the Portable Executable (PE) information of the executables [6]. PE is a format for executables, object codes, DLLs and more in Windows OS. PE format encapsulates the crucial information for Windows OS loader to wrap the executable code.

I used Pefile library to obtained the DLLs and it's import table in each executable. I found that the experience and domain knowledge suggest that the imported DLLs benign samples to malicious ones. For instance, importing a crypto library will highly indicate a ransomware. If an execution code does not contain so many imported DLL, it can indicate a safer intent.

After I collected the DLLs, I had a obtained my corpus. The distribution of the corpus can be seen in the table below.

Executable	Corpus Size
Benign	2174
Malicious	133
Total	2307

Table 1: Corpus Distribution

When the I examined the benign corpus and malicious corpus, I noted that there are so many DLLs such as kernel32 which means stop word. For example, in the English literature word "the" is used frequently but solely it does not hold any meaning. Hence, I divided the DLLs with their corresponding term frequencies (TF) which is the number of times a word occurs in a document and Inverse Document Frequency (IDF) which is how many documents it occurs at. IDF gives common and meaningless words a small weight and important words a large weight.

With this process we allow the divergent DLLs to shine through the crowd. Also, this can make the identification of the execution easier. For instance, when the word “Socrates” is in a document, we can indicate it is a philosophy article since the word is rare. Likewise, a crypto library can indicate a ransomware.

The transformed corpus sample can be seen in the table below.

(0, 546)	0.6184229637164015
(0, 398)	0.5404658886689443
(0, 271)	0.30570931604172263
(0, 21)	0.4816549337635947
(1, 683)	0.41530100358185346
(1, 646)	0.07736845959520693
(1, 634)	0.0569686039198615

Table 2: Transformed Corpus

Then, I proceed with using classifiers. I used Random Forest Classifier. I divided the corpus into %33 test and %67 train data set. Also, before fitting the train data, I gave larger weights to the under representative class which is malicious class in our case. Then, I fitted the training data set, then I predicted the test.

It can be good to note that the costumer can be frustrated if the false positive rate (FPR) exceeds %0.01. Since this is a HIDS, users need to open files, need to execute various programs daily and if the legitimate processes be held by the HIDS, this can lead to a delay in the project management phases. Therefore, clients may prefer to have a low FPR then focusing on high accuracy. Hence, I set the constrains on classifier’s confusion matrix’s (FPR) to be less than %0.1.

Also, in order to maximize true positive rate (TPR), I choose to threshold this matrix rather than giving 0 and 1. After multiple iterations, I found our classifier that satisfies the constrains in the 43th iteration. Also, we can see that we have 0.58% true positive rate in the 44th iteration. After the 44th iteration the FPR increases.

Iteration	FPR	TPR
41	0.015256588072122053	0.14583333333333334
42	0.04576976421636616	0.3333333333333333
43	0.11511789181692095	0.375
44	0.19833564493758668	0.5833333333333334
45	0.3287101248266297	1.0

Table 3: Classifier Iterations

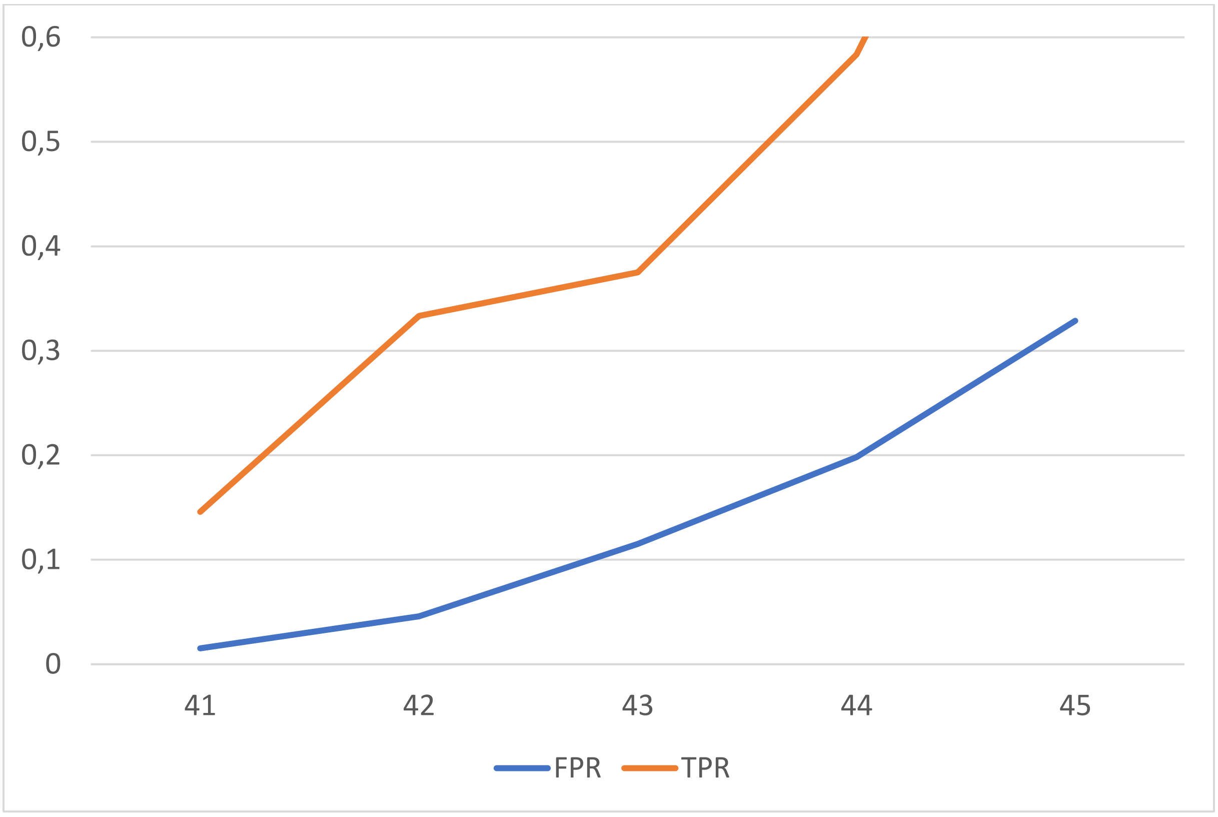


Figure 6: Iterations Performance

3. NETWORK IDS

After all these attempts, I tried to do implement a model based on CICIDS2017[2] dataset [7]. Because this data set was relatively new and it was based on real new attacks. I choose isolation forest for anomaly detection since it is one of the best unsupervised anomaly detection models. I choose to split the train, test and validation into two different approaches.

Isolation forest lies to the fact that is it easier to isolating anomaly then isolati ng normal. Also, it notes that an anomaly point's tree would be less shallow if we com pare it to a normal point's tree. Distribution of the benign and malicious samples' amo unt and percentages in the CICIDS2017 data set can be seen in the table below.

CICIDS2017	Amount	Percentage
Benign	2273097	%80.30036637024273
Malicious	557646	%19.699633629757273
Total	2830743	%100

Table 4: CICIDS2017 Distribution

When we examine the malicious samples, we can see a variety of new attack types that is illustrated in the figure below.

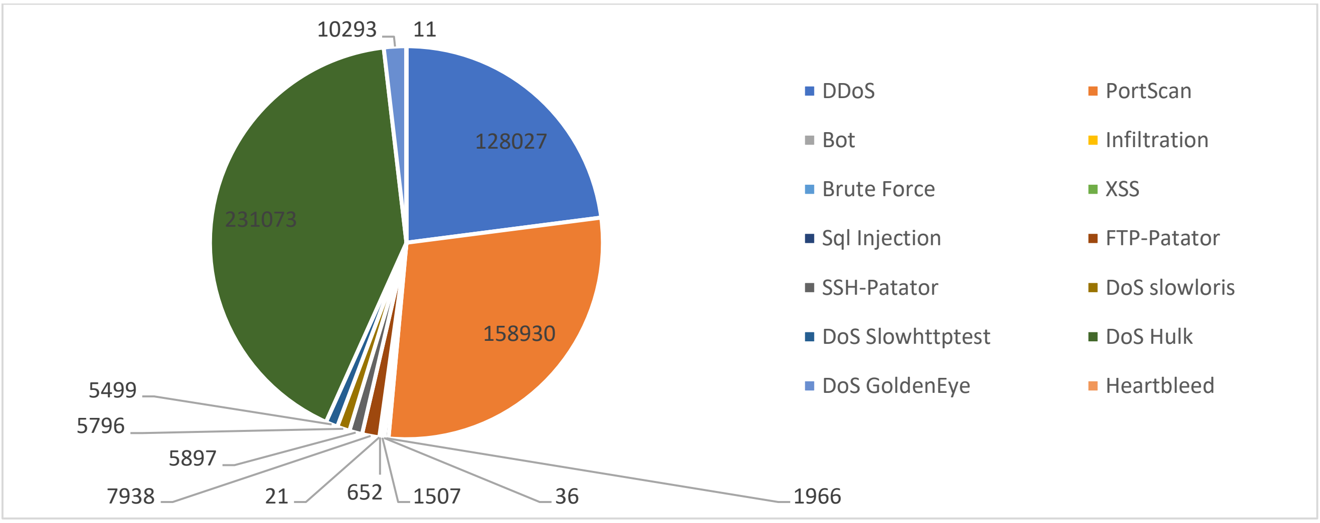


Figure 7: Malicious Data Set Types Amount

Since isolation forest does not expect object types, I enumerated each attack label. The attacks' corresponding values can be seen in the table below.

[2] CICIDS2017 is one of the newest datasets where it contains up-to-date attacks and scenarios. It is published in 2017 by the Canadian Institute of CyberSecurity.

Label Name	Number
Benign	1
DDoS	2
PortScan	3
Bot	4
Infiltration	5
Brute Force	6
XSS	7
Sql Injection	8
FTP-Patator	9
SSH-Patator	10
DoS slowloris	11
DoS slowhttptest	12
DoS Hulk	13
DoS GoldenEye	14
Heartbleed	15

Table 5: Label Enumerating

3.1. Distinctive Data Split

In this approach, I split the data in a way that the similarities between each data sets are diminished significantly. Basically, I distributed each malicious attack into a data set and tried to set a boundary between them to lower the chance of model's temptation to memories. The distribution of the data sets can be found in the figure below.

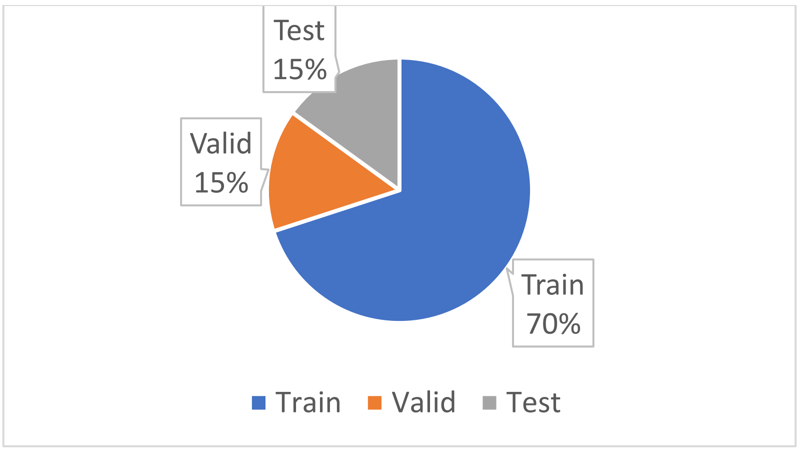


Figure 8: Split of Data Set

When we look at each data set, we can see the minimized similarity of attack s in the tables below.

Train Data Set Labels	Amount
Benign	1679249
DDoS	128027
PortScan	158930
Bot	1966
Infiltration	36
Brute Force	1507
XSS	652
Sql Injection	21
FTP-Patator	7938
SSH-Patator	3194
Total:	1981520

Table 6: Train Data Set Label Distribution

Valid Data Set Labels	Amount
Benign	228862
SSH-Patator	2703
DoS slowloris	5796
DoS slowhttptest	5499
DoS Hulk	181751
Total	424611

Table 7: Valid Data Set Label Distribution

Test Data Set Label	Amount
Benign	364986
DoS Hulk	49322
DoS GoldenEye	10293
Heartbleed	11
Total	424612

Table 8: Test Data Set Label Distribution

Data Set	Benign Samples	Attack Types	Malicious Ratio (Contamination Ratio)
Train	Yes	2,3,4,5,6,7,8,9,10	%15.25450159
Valid	Yes	10,11,12,13	%46.10078401
Test	Yes	13,14,15	%14.04246701

Table 9: Distinctive Data Set Summary

Data Set	Main Attack Type	Main Attack Type Percentage	Intersection with Train Data Set	Intersection Attack with Train Data Set
Valid	DoS	%98.61915003397208	%0.0138084996602792	SSH-Patator
Test	DoS	%99.98155167208936	%0	None

Table 10: Valid and Test Data Sets Comparison with Train Data Set

Then I started to train the isolation forest with the train dataset. I left the cont amination of the dataset default which is defined as auto in the official document [8]. T hen, I started with the validation data set.

Then, I predict the validation data to classify them into being malicious or benign. I obtained the confusion matrix of the models over validation data set. The matrix can be seen in the table below.

True Positive	% 65.53852126958503
False Negative	% 34.08242187699554
True Negative	% 94.49144025657384
False Negative	(% 5.443891952355568

Table 11: Confusion Matrix of Validation Data Set

I repeated the same procedure for the test data set. I obtained the similar dis tribution and scatter graphs that mentioned before. Also, I obtained the corresponding confusion matrix.

True Positive	% 43.45252071244089
False Negative	% 56.2003152986952
True Negative	% 94.60719041278296
False Negative	(%5.3928095872170445

Table 12: Confusion Matrix of Test Data Set

I hypothesis that since the valid dataset shares the malicious attack sample S SH-Patator, that also has in the train data set, the true positive is higher than the test data sets since the test data set does not share any same malicious attack type.

Then, I set the contamination value from auto to %20 since it was the ratio th at the train data set has. Then I looked at the confusion matrixes of the validation data set and test data set.

True Positive	% 73.37610920106872
False Negative	% 26.62389079893128
True Negative	% 87.60388356302052
False Negative	%12.39611643697948

Table 13: Confusion Matrix of Validation Data Set

True Positive	% 54.1005601583202
False Negative	% 45.8994398416798
True Negative	% 84.61475234666536
False Negative	%15.8524765333464

Table 14: Confusion Matrix of Test Data Set

Since the true positive is increased, I wanted to examine the behaviour of the model. I graphed the true positive of the models over test and validation data with the increased contamination each time. The graph can be seen in the figure below.

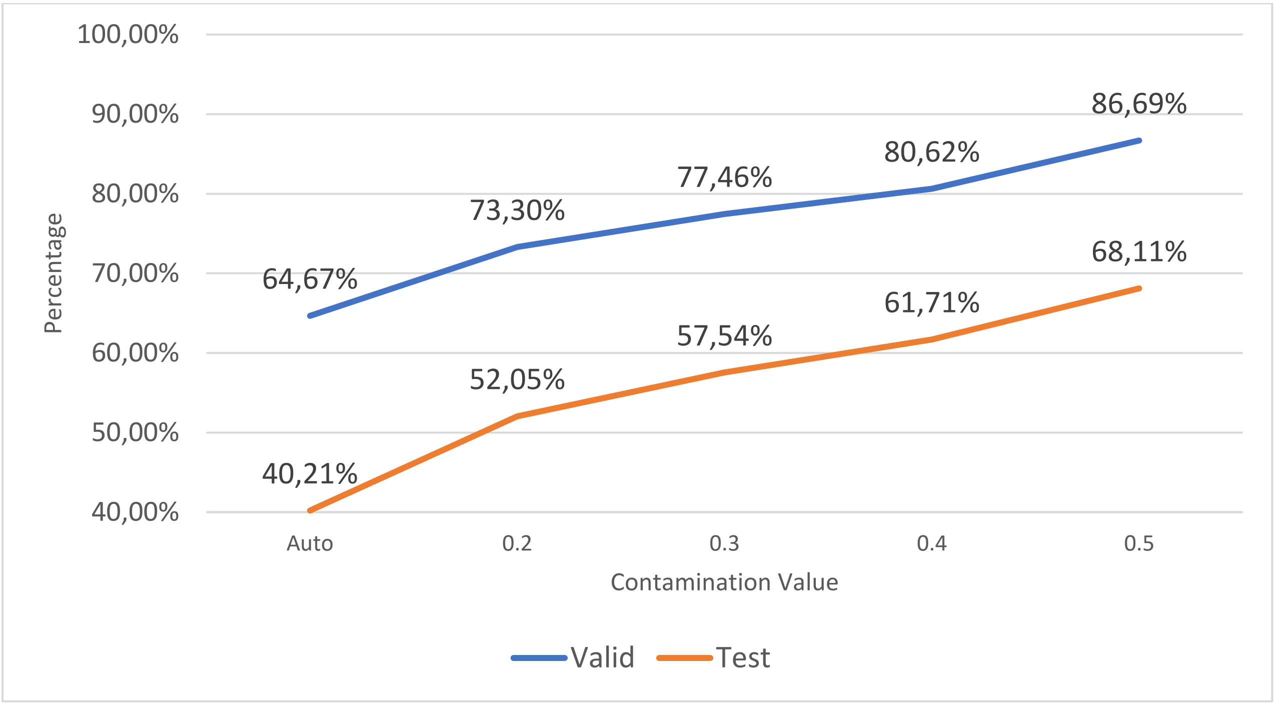


Figure 9: True Positive Changes

However, the true negative percentage shrunk down significantly with each increased contamination value to the point where I see the %57.23623843189346 for validation and %55.93420021589869 for the test data set.

This can mean that the model increased the amounts of cells it flagged as malicious and this leads to a better coverage of the malicious samples and therefore, the true positive ratio. However, since the test data does not contain any shared values with the training data and also the contamination ratio of the validation data set %46,10078401171896, it is still relevant with the data sets.

Also, it is important to note that the valid data set has more accuracy because of it share one type of attack with the train data set. On the other hand, the test data set does not have any intersection of malware types with the train set. Hence, the accuracy is lower than the valid data set's accuracy. This led me to experiment with the same model but dividing the data set in an inclusive way so that each three data set contains the all 15 malware types.

In the last model with the contamination value of 0.5, I graphed the scored that the de cision functions assigned to each validation data cell. The graph can be seen in the fi gure below. Also, for a better understanding, I scattered the scored that the decision f unctions assigned to each validation data cell. The scatter can be seen in the figure b elow.

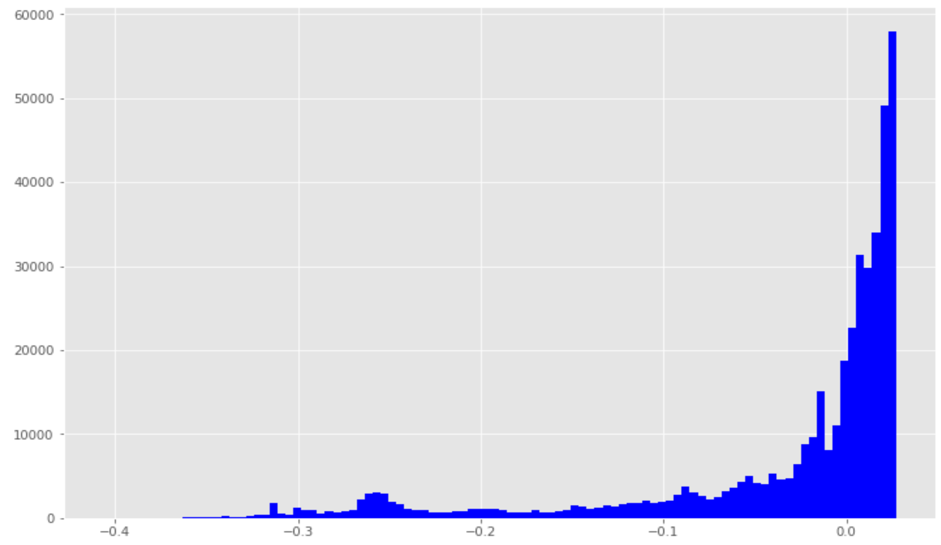


Figure 10:Validation Data Set Scores

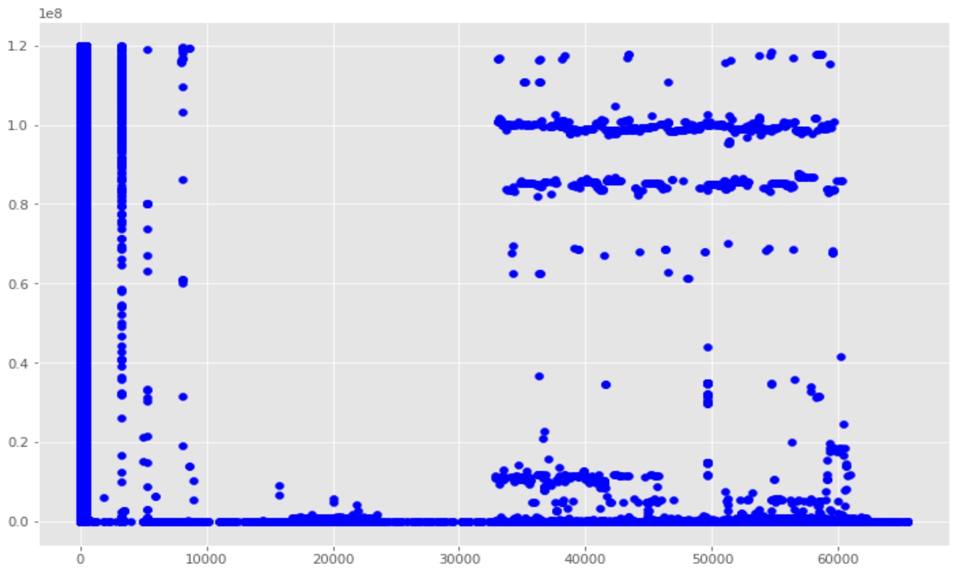


Figure 11: Validation Data Set Score Scatter

After I obtained the predictions, I illustrated into the scattered with the anomalies' edge color is showcased as red. The scatter can be seen in the figure below.

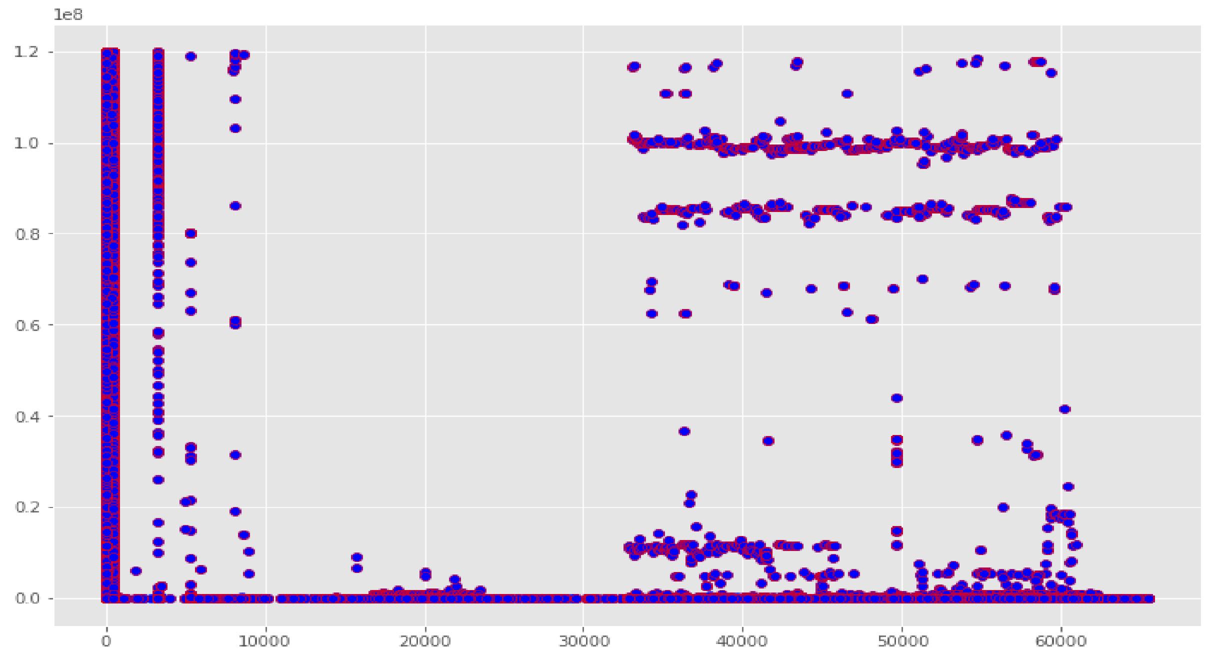


Figure 12: Validation Data Set Predictions Over Scores

3.2. <u>Inclusive Data Split</u>

I stumbled upon the other way of splitting data to preserve the contamination ratio and the malicious attack type is consistent between data sets in previous studies [9]. I tried this way too. The ratio of the split can be seen in the figure below.

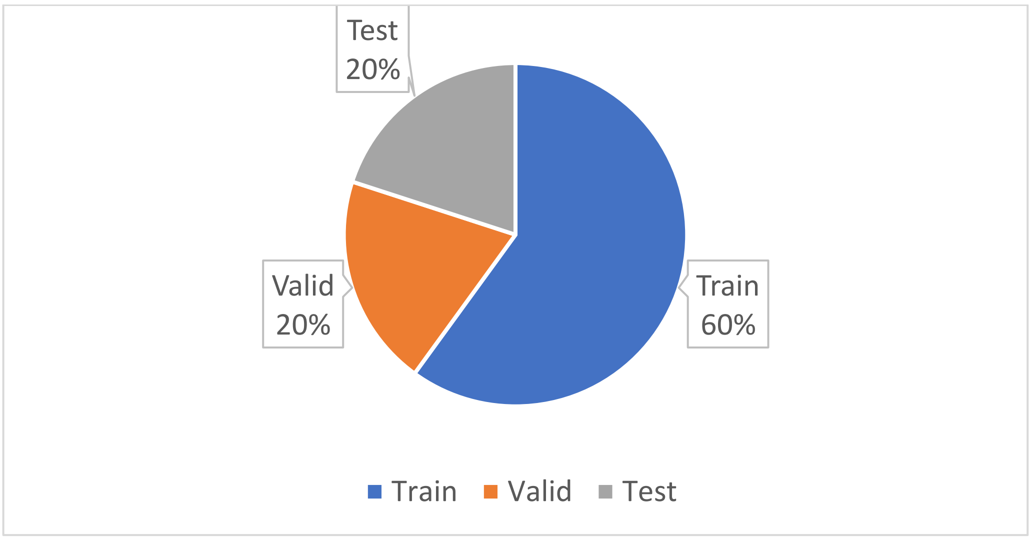


Figure 13: Split of Data Set

Each data set's distribution of the malicious attacks is very similar and inclusi on. This it shown in the table below.

Label	Train	Valid	Test
Benign	1363105	454126	454089
DDoS	76789	25648	25588
PortScan	95220	31872	31712
Bot	1204	366	386
Infiltration	22	9	5
Brute Force	929	287	291
XSS	383	147	122
Sql Injection	10	6	5
FTP-Patator	4804	1561	1570
SSH-Patator	3514	1141	1242
DoS slowloris	3473	1177	1146
DoS slowhttptest	3280	1116	1103
DoS Hulk	137797	46068	46259
DoS GoldenEye	6189	2049	2055
Heartbleed	6	2	3
Total	1696725	565575	565576

Table 15: Inclusive Data Sets

Then, I repeated the same steps that I did it to distinctive data sets. I first started with the 0.2 contamination value and work my way up into the 0.5.

The initial phase where the contamination value is 0.2, the distribution of scores and the predictions can be seen in the figures below. I would like to point out that both of these graphs showcase a different pattern from previous data sets.

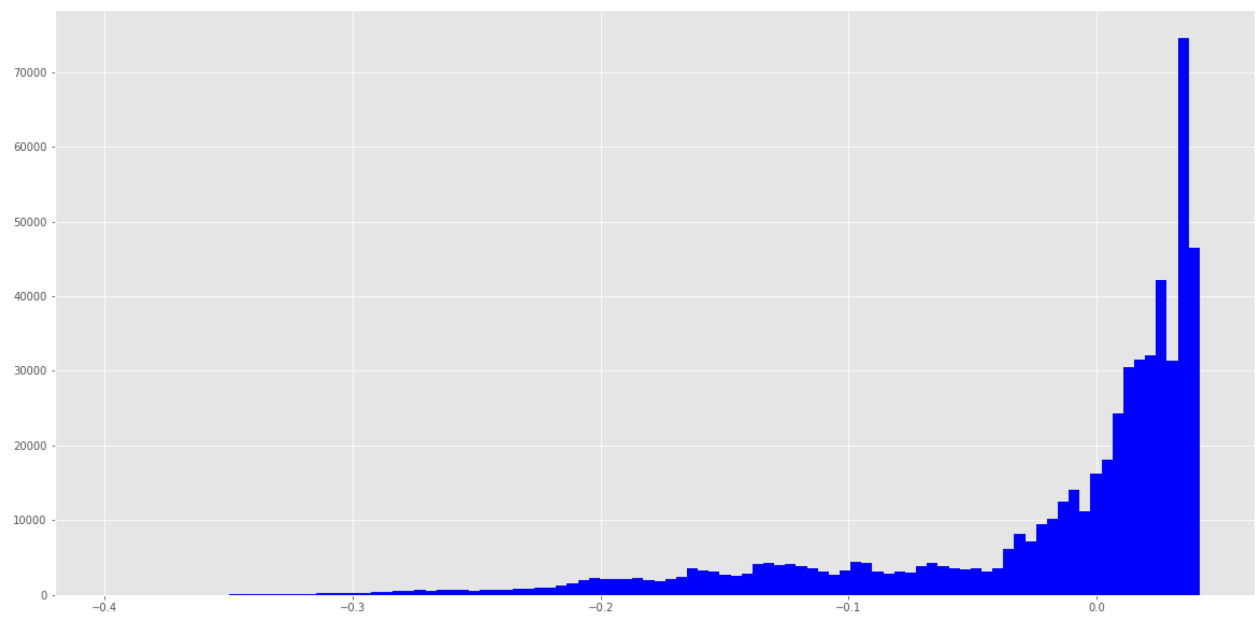


Figure 14: Distribution of Scores

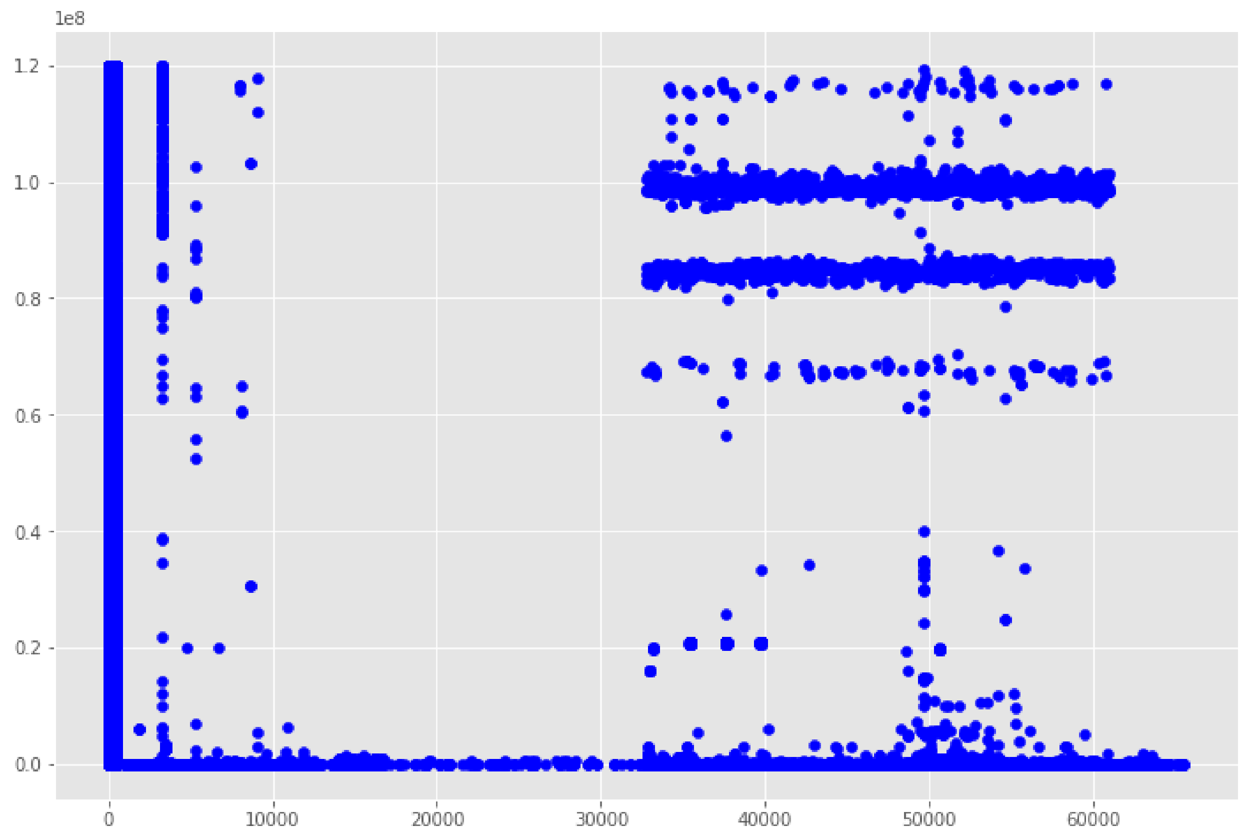


Figure 15: Inclusive Data Set Score Scatter

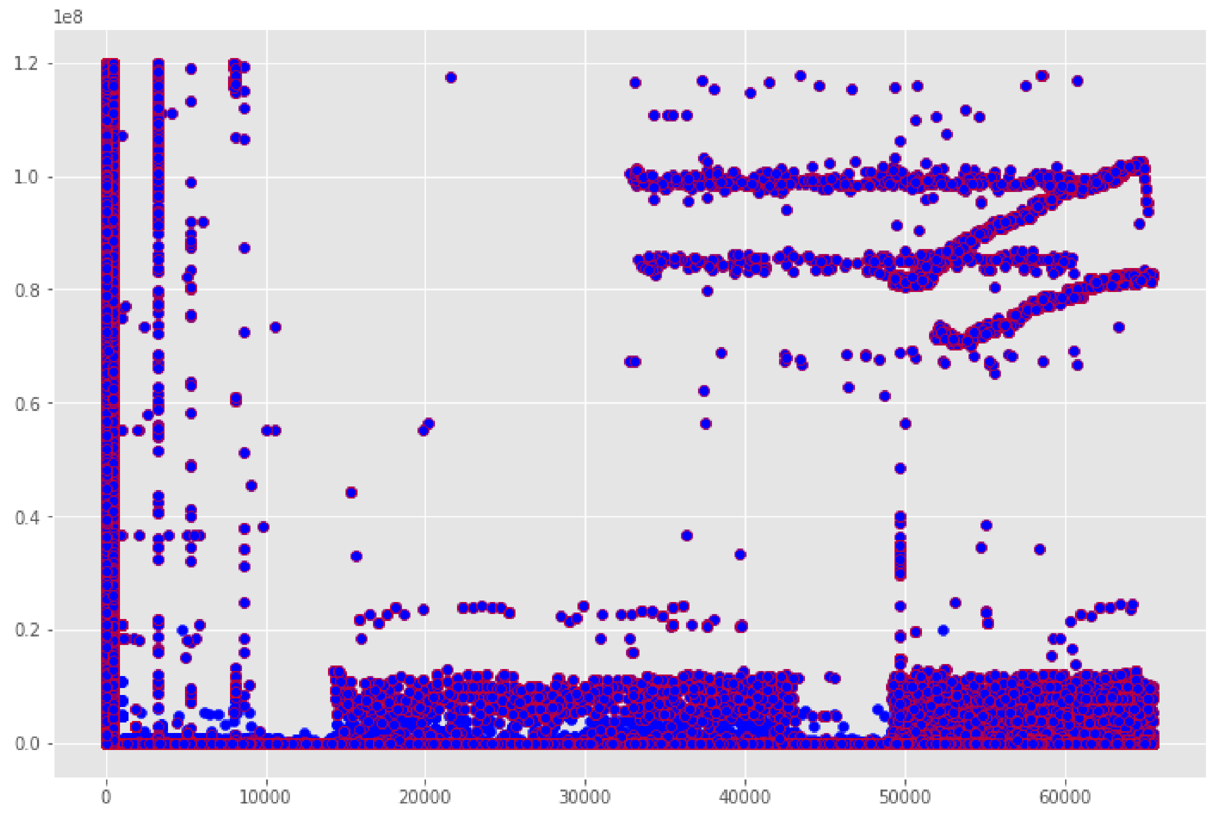


Figure 16: Inclusive Data Set Score Predictions Scatter

Then I started to increase the contamination value and the model's response to it. It can be seen in the figure below. In this experiment, I faced with way below true positives then the distinctive data set.

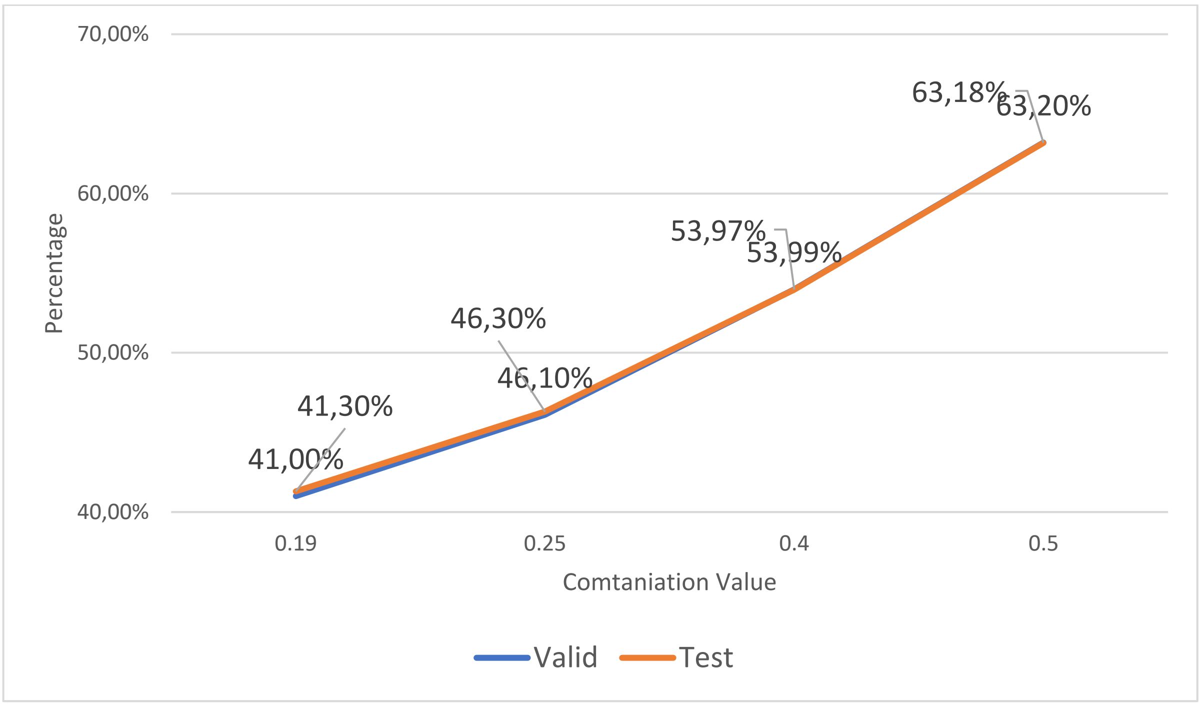


Figure 17: True Positive Changes

CONCLUSION AND FUTURE WORK

In the initial phase of my project, I tried to implement a HIDS that can be used in the computer s who use Windows OS. Benign and malware samples were gathered. For malwares, WannaCry was selected to do malware analysis. Static and dynamic analysis made. After the analysis, the PE headers were obtained and the DLLs imports were examined. DLLs examined and the top frequent DLLs use to identify malwares and Random Forest Classifier is used for the classification. Since the malwares are minority in comparison to the benign samples, higher weight was given to the malwares. Then, the model's threshold is iterated until it satisfies the constraints. For future works, more experiments can be done with the classification algorithms that can be found in Waikato Environment for Knowledge Analysis (WEKA) [10].

After the HIDS, the implementation of NIDS was started. I used CICIDS2017 data set that contains 14 different attacks and benign network traffic. I used Isolation Forest for unsupervised anomaly detection. I conducted two different approach for splitting the data set for train, validation and test data sets. In first approach, I designed validation data set to have only 1 intersected attack types. However, in the test data set, there were no attacks that included in the train data set. I achieved higher accuracy in validation data set then test data set. Which is interesting because both validation and test data set mainly consist of DoS attacks. In the second approach, I split all 14 malware types in test, validation and test data set. I expected higher accuracy like I gained from the validation test in the first approach but interestingly, the accuracy for both data sets were much lower than the first approach which was contrary to the discoveries from the first approach. For future works, the cause between these situations can be further examined. Also, a signature-based detection can be merged with the anomaly-based detection that are implemented in NIDS. Lastly, the NIDS can be tested in the real work situations [11].

BIBLIOGRAPHY

[1] A. J. C. I. A. J. P. D. Anitta Patience Namanya, "The World of Malware: An Overview," in *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, Barcelona, 2018.

[2] Wikipedia, "Stuxnet," 19 12 2020. [Online]. Available: https://en.wikipedia.org/wiki/Stuxnet. [Accessed 08 01 2021].

[3] Wikipedia, "WannaCry ransomeware attack," 07 01 2021. [Online]. Available: https://en.wikipedia.org/wiki/WannaCry_ransomware_attack. [Accessed 08 01 2021].

[4] A. A. N. J. M. S. SH Kok, "A Review of Intrusion Detection System using Machine Learning Approach," *International Journal of Engineering Research and Technology,* vol. 12, no. 1, pp. 8-15, 2019.

[5] S. S. Yuval tisf Nativ, "theZoo - A Live Malware Repository," GitHub, 2020.

[6] M. S. Belaoued M., "A Real-Time PE-Malware Detection System Based on CHI-Square Test and PE-File Features.," *IFIP Advances in Information and Communication Technology,* no. 456, pp. 416-425, 2015.

[7] «Intrusion Detection Evaluation Dataset (CIC-IDS2017),» Canadian Institute of Cybersecurity, Canada, 2017.

[8] F. a. V. G. a. G. A. a. M. V. Pedregosa, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research,* no. 12, pp. 2825--2830, 2011.

[9] E. Sharova, *Testing Isolation Forest on Non-Financial Data - 2 KDDCUP99 Sets¶*, London: PyData, 2018.

[10] J. W. G. Z. Jinrong Bai, «A Malware Detection Scheme Based on,» *The Scientific World Journal,* no. 2014, p. 11, 2014.

[11] N. A. A. E. O. K. M. M. R. Zouhair Chiba, "Newest collaborative and hybrid network intrusion detection framework based on suricata and isolation forest algorithm," in *The Fourth International Conference on Smart City Applications* , Casablanca, Morocco, 2019.

[12] Microsoft, "Microsoft Malware Classification Challenge (BIG 2015)," 2015. [Online]. Available: https://www.kaggle.com/c/malware-classification/data.

[13] E. Tsukerman, "Cybersecurity Data Science," Udemy, 2020.