

Detection of malware traffic with Netflow

Bachelor Degree in Informatics Engineering, Information Technology

Author: Joël Codina Poquet

Director: Pere Barlet-Ros

Date: 26/10/2017

Abstract

Traffic classification has always been a fundamental aspect regarding the identification of applications on the network that has allowed to apply different actions or services depending on their type, such as best-effort delivery or discarding the traffic in case of a malicious application. The constant growth of the internet and the relentless pace of change on the applications have made traditional methods no longer feasible. Nonetheless, the application of machine learning methods has gathered strength over time. In this thesis, we analyze different algorithms with the aim to obtain a good classifier and design a solution capable of achieving an acceptable performance when identifying malware traffic in a real-time environment using the information provided by Netflow protocol.

Keywords: Traffic analysis, Malware, Netflow, Machine Learning, C5.0, Random Forest, Naïve Bayes.

Acknowledgement

I would like to thank my thesis' director Dr. Pere Barlet-Ros for giving me the opportunity to do this investigation project and guiding me through the process. This work wouldn't have been possible without his support.

Also thanks to all my family that has given me strengths during these 4 years path. Special gratitude to all my friends for the amazing times and all the unbelievable companionship granted during the degree.

Content

1. Context	1
1.1 Introduction.....	1
1.2 Traffic classification	2
1.3 Stakeholders.....	4
2. Formulation of the problem.....	5
2.1 Problem	5
2.2 Objectives.....	5
3. Project Management.....	7
3.1 Project Scope.....	7
3.2 Methodology and Rigor.....	7
3.3 Time schedule	9
3.3 Resources	11
3.5 Alternatives and action plan	13
3.6 Cost estimation	13
3.7 Control management	16
4 Sustainability	17
4.1 Economic	17
4.2 Social	17
4.3 Environmental.....	17
5. Data acquisition.....	19
5.1 Obtaining normal traffic captures.....	19
5.2 Obtaining malware traffic captures	19
6. Data preparation	21
6.1 Capture processing.....	21
6.2 Data Cleaning	22
6.3 IP Selection	23
6.4 The Dataset	23
7. Algorithms Used	25
7.1 Naïve Bayes	25
7.2 Decision Trees	26
8. Model Evaluation	30

8.1 Metric	30
8.2 K-fold Cross Validation	30
8.3 Binary Response	31
8.4 Multiclass Response	37
9. Optimization.....	41
9.1 Blacklist Concept	41
9.2 Using IP blacklists for optimization	41
9.4 Implementation.....	43
9.5 Optimization Results	44
10. Conclusions.	47
10.1 Summary	47
10.2 Future Work	47
Appendices.....	48
Gantt Diagram	49
11. References.....	50

Tables Index

Table 3.1: Time estimation per task.....	11
Table 3.2: Direct costs of each task according to Gantt chart	13
Table 3.3: Cost per hour of each role.....	14
Table 3.4: Hardware costs estimation.....	14
Table 3.5: Indirect Costs.....	15
Table 3.6: Final budget estimation.....	16
Table 4.1: Sustainability Matrix.....	18
Table 5.1: List of Malware Captured Used for the Dataset.....	20
Table 6.1: Binary Dataset after under-sampling	24
Table 6.2: Multiclass Dataset after under-sampling	24
Table 7.1: Versatile machine learning algorithms and factors.....	25

Tables of Figure

Figure 1.1: Barplot of the malware traffic evolution since 1984	1
Figure 1.2: Machine learning Traffic classification.....	3
Figure 3.1: Phases of Iterative Waterfall methodology	14
Figure 6.1: <i>Nfdump</i> command used to convert <i>nfcapd</i> files.....	21
Figure 6.2: Steps to transform .pcap to a CSV Netflow dataset.....	22
Figure 6.3: Summary of data before cleaning.....	22
Figure 6.4: Program that selects the Public IP from two columns.....	23
Figure 7.1: Example of a decision tree where leaves are the class.....	27
Figure 7.2: Example of how random forests make a decision.	28
Figure 7.3: Random Forest in Pseudo-code.	28
Figure 8.1: K-fold Cross Validation Pseudo-code.	31
Figure 8.2: 10-fold Cross Validation of Naïve Bayes.	32
Figure 8.3: Accuracy of Random Forest for different number of decision trees..	33
Figure 8.4: Linear relation between time and number of decision trees..	33
Figure 8.5: 10-fold Cross Validation of Random Forest with Number of Trees = 300..	34
Figure 8.6: Variable Importance for Random Forest..	35
Figure 8.7: Accuracy of Boosted C5.0 Decision Tree.....	36
Figure 8.8: 10-fold Cross Validation of Boosted C5.0 with Number of Trials = 20.....	36
Figure 8.9: Comparison for different models' performance for binary class.....	37
Figure 8.10: Confusion matrix of Multiclass Naïve Bayes.....	38
Figure 8.11: Confusion matrix of Multiclass Random Forest n = 10.....	38
Figure 8.12: Confusion matrix of Multiclass Random Forest n = 300.....	38
Figure 8.13: Confusion matrix of Multiclass C5.0 without boosting.....	39
Figure 8.14: Confusion matrix of Multiclass Boosted C5.0 with 20 trials.....	39
Figure 8.15: Comparison for different models' performance for multiclass.....	39
Figure 9.1: Blacklist optimization workflow.....	42
Figure 9.2: Accuracy over % Checked Flows.....	45
Figure 9.3: Time spent over % of Checked Traffic.....	46

1. Context

1.1 Introduction

Malware, short for malicious software, is a type of software that wants to gather sensitive information, gain access of the computer, display unwanted advertising or destroy all what is inside the computer. This term includes all the intrusive and dangerous software like computer viruses, worms, trojan horses, ransomware, spyware, adware, scareware...

In a world where malware traffic has increased every year [1] (Figure 1.1), there is a need for organizations to prevent any type of malicious software inside their systems and laptops in order to protect their critical information. This final degree project of the Facultat de Informàtica de Barcelona (Universitat Politècnica de Catalunya) aims to find a proper model that uses Netflow features and is able to detect and classify malware traffic with them.

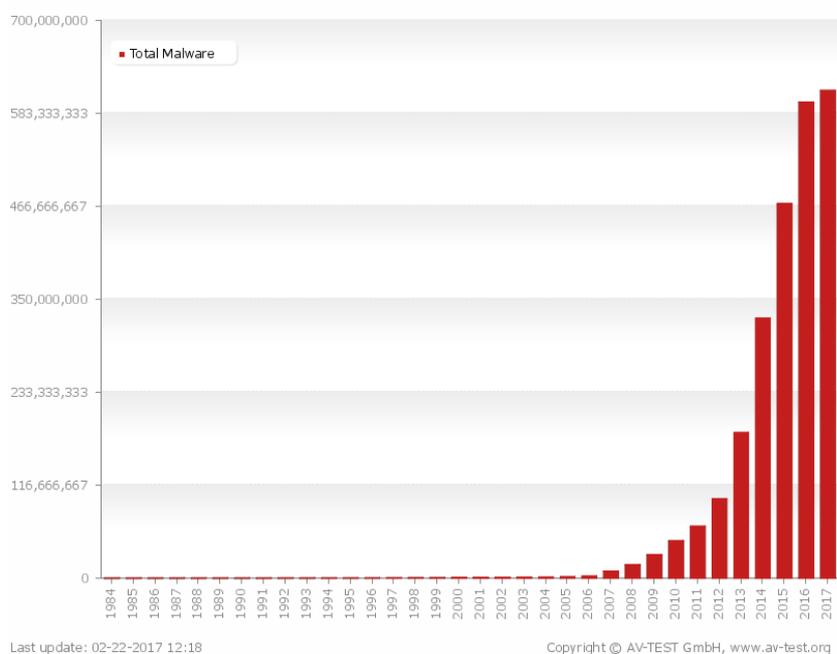


Figure 1.1: Barplot of the malware traffic evolution since 1984

Routers and switches, that support Cisco standard Netflow version 5, can collect network traffic statistics in any interface with Netflow enabled and give us the information summarized in IP Flows. An IP Flow is a unidirectional sequence of packets that share a set of 5 up to 7 IP packet values (Ingress interface, Source IP address, Destination address, IP protocol, Source port, Destination port and IP Type of Service).

1. CONTEXT

1.2 Traffic classification

Network traffic classification is used to identify applications, protocols and content that exist in a network [2]. Some actions that could be done after having classified network traffic could be monitor, analyze, optimize, control or discovery with the finality of improving the network performance. After classifying the traffic, the router could adopt different policies depending on the class they have. For example, to guarantee a good quality service in case of streaming media, to discard flows tagged as malware or to provide a best-effort delivery.

1.2.1 State of art

Traffic classification is an automated process that categorizes the network traffic making use of some parameters like port numbers or protocol. We are going to make a brief explanation of three traditional means that have been used to perform the classification: Port based, Deep Packet Inspection and statistical analysis.

The first approach is port based classification that has been used in several papers [3] which are based on the correlation between the transport layer's port number and the application or group of applications. They make use of well-known ports which have a range between 1 and 1023 since 1992 (Before it was from 1 to 255). Port descriptive information is managed by the Internet Assigned Numbers authority (IANA). They also used information of non-well-known ports used by many applications, whose information is registered in RFCs like the RFC1700 (now obsolete). The approach is supported by many devices and it doesn't require information about the application-layer payload, which doesn't compromise users' privacy. As it only requires from port information, this method is fast and low resources consuming. However, there are a lot of applications that use dynamic ports so it is only useful for applications and services that have a fixed port assigned, which are easy to cheat by changing the port number on the system.

After knowing that port based inspection is not suitable in a lot of scenarios, we introduce the second approach that would be Deep Packet Inspection (DPI). This method consists on inspecting the whole packet payload but also the headers like the ports, flags, protocol, etc providing a higher accuracy than the previous one. This method searches for signatures inside the packets that could identify the flow as a certain application. Even if this method seems to be quite good, it faces some important problems. Analyzing every packet's payload is slow and high resource-consuming. Also, nowadays there is more encrypted traffic than before; this makes the signatures almost impossible to be identified. Moreover, the signature database should be updated frequently as applications change very quickly. This method has also an ethical impact as it is used in some countries as China or Iran to censor pages and has been criticized recently because people think that analyzing the content of a packet attempts against their privacy.

1. CONTEXT

The third approach, statistical classification copes with some of the problems that have the other methods relying on statistical analysis of the traffic attributes like protocol, packet size, number of bytes, etc. It usually uses machine learning algorithms like Naïve Bayes, Decision Trees and K-means. After having trained the model, it can be a fast technique to identify the class or even unknown never seen classes. For example, this method could be applied to identify between applications like BitTorrent and Skype or between malware traffic.

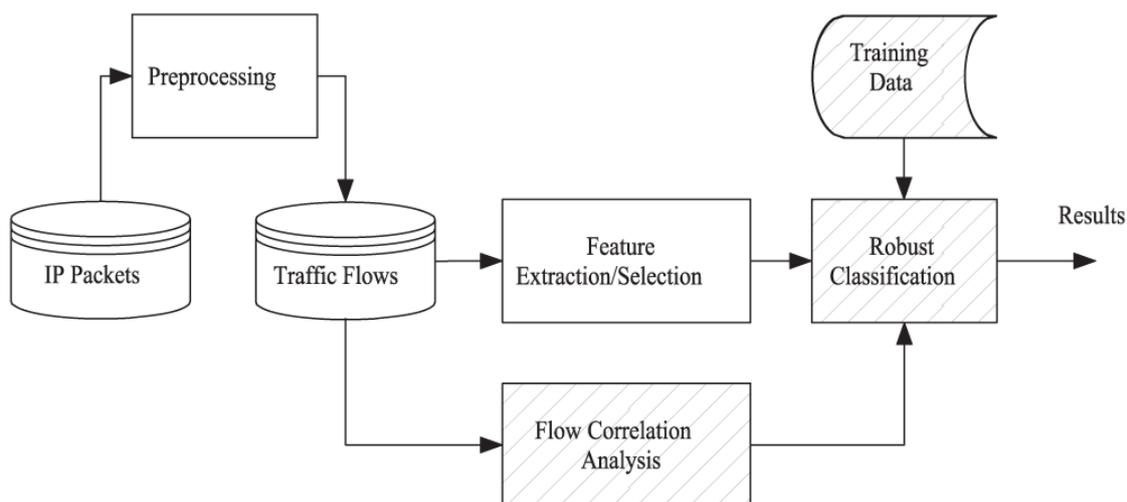


Figure 1.2: Machine learning Traffic classification

Malware traffic analysis is not new. It has been there since the creation of malicious software and the network monitoring tools. There has been plenty of work related to this field where they have tried to detect and study malware traffic by Port based and DPI methods stated above.

On the last decade, the rising in the machine learning field has led to an increased interest on trying to apply its techniques into network malware traffic analysis. Some work uses unencrypted packet content and other features with regard to classify benign and malicious traffic, but nowadays most malware uses Transport Layer Security (TLS) because traditional pattern-matching techniques can no longer be applied to its messages. Blake et al. [4] try to detect malware traffic that uses TLS using as features a set with Netflow, Sequence of Packet Lengths and Times, Byte Distribution and Unencrypted TLS Header Information, successfully achieving high accuracy. However, they used features that need deep packet inspection, which is difficult to perform in real time. Nigel et al. [5] states that comparing the accuracy of a model is not enough; computational performance, number and type of statistics calculated are vitally important and must not be ignored. Blake's work also used biased results after relying in consensus vote from VirusTotal [6] instead of a robust clustering algorithm.

1. CONTEXT

There is also a line of study that analyzed the impact of using different types of sampling techniques when the dataset is created. It stated that using the wrong technique could increase the number of biased samples affecting the whole result [7].

Finally, there has been some work about Netflow Analysis. Leyla et al. [8] created a program that uses Netflow Analysis and the properties of command and control (C&C) channels to be able to perform accurate botnet detection. However, they only classify botnet malware which has some patterns of crowd behavior that ease the classification of the traffic. Wagner et al. [9] also gives a real-time network traffic solution for detecting worms in fast IP networks using an entropy-based approach.

1.3 Stakeholders

According to the Project Management Institute (PMI), a project stakeholder is “an individual, group, or organization, who may affect, be affected by, or perceive itself to be affected by a decision, activity, or outcome of a project”, in other words the target audience, users and beneficiaries. The main actors or stakeholders of this project are:

- 1) Director and student: They are the ones who will enable the project to progress and reach the goals established in time, in order to successfully finish the final year project. The director Pere Barlet-Ros has the function to guide the student and recommend which paths he should take to overcome any possible issues. On the other hand, the student Joël Codina Poquet is going to implement the project and document it.
- 2) Network Analysis Companies: These companies will take profit of the investigation done in this project, by adapting and adding this knowledge to their traffic analysis tools for increasing their value and offering better and more functionalities than before.
- 3) Firewall Manufacturers: Companies that provide firewalls to protect companies could implement in their hardware the ideas of this final project in order to discard malicious traffic before the end user.
- 4) ISPs: It becomes increasingly critical for them to detect anomalous traffic to ensure quality of service and provide value-added services.
- 5) Companies with critical information: There are companies that have critical and confidential information which must not be accessed, that is why they spend a lot of money with expensive hardware that analyses all the traffic. However, with the model that this project wants to reach used as a complement, the hardware inversion could be reduced.

2. Formulation of the problem

2.1 Problem

It has been stated that malware traffic increases yearly causing a lot of trouble to small and big organizations that want to protect their data against them. There are basically two levels of protection that many companies deploy.

- 1) Antivirus/Anti-malware: Software that protects computers against a lot of types of malware. In general, enterprise licenses are not free, which causes more costs to companies, but also they don't ensure a 100% protection as they are signature-based that makes them vulnerable for non-previously detected viruses. Malware authors are always a step ahead writing new type of viruses like "metamorphic viruses" [10] which are not detected in the first instance.
- 2) Firewall protection: Hardware that monitors incoming and outgoing traffic in a network, it can filter potentially dangerous packets if they match in their database and protect the internal network with the internet. Even so, there is still a problem when newest viruses that don't match the database surpass the filters. One could expect that analyzing the payload (content of the packet), the firewall would be able to detect the malicious traffic, but in an organization the amount of packets per second is huge and analyzing each packet makes the process too slow and impossible to deploy as it would cause large queues of packets waiting to be processed. There is hardware specialized in traffic analysis but it is expensive and not affordable for medium – small companies.

Regarding the problems stated above, there is clearly a need to create a protection against malware without increasing the costs buying new hardware. To do that, one option could be the analysis of IP Flow statistics that some routers collect, without paying attention to the payload. Some objectives have been defined to try to archive this goal.

2.2 Objectives

One of the objectives of this project is the study of machine learning traffic classification to classify normal traffic from malicious traffic as a binary response but also to study whether there is any behavior pattern between the most common type of malware such as Trojans or Botnets, that the human eye will not identify but the machine learning algorithm could. In order to achieve our objectives we should define some goals:

2. FORMULATION OF THE PROBLEM

- 1) The creation of a dataset with the flow information. To do that, it will be necessary to capture benign traffic and malware traffic that would imply the use of malicious software. To obtain Netflow traffic captures, *softflowd* [11] software will be used in addition with *nfcapd*[12] and *nfdump*[13].
- 2) The creation of a proper statistical model using only the fields provided by Netflow protocol in our dataset and some transformations of its features. Applying different machine learning algorithms and evaluating them to take the best one.
- 3) The study of ways to improve the model using additional features and public information, without giving up the idea of making it fast, and implement them. This could allow creating a better classification model.

3. Project Management

3.1 Project Scope

The goal of this project is to accomplish all the objectives proposed in the previous section on time. Nevertheless, there could be obstacles that may hinder the execution of the project:

- 1) Time: Sometimes something that should have been finished by the time the project ends, lasts more than expected. The final year project doesn't give so much time for the student and he may face difficulties to finish in time. Getting a considerable training dataset is a challenge, as generating and isolating malware traffic is not as fast as generating benign traffic. Even so, the project has been bounded properly to reduce that risk.
- 2) Non-ideal conditions: In pursuance of emulating a real environment, we will capture traffic with a software instead that with a router which may cause us some trouble. The Ethernet card of the PC that will be use may not be suitable enough to capture all the packets making false data.
- 3) Malware traffic generation: For the generation of malware traffic, it will be required the execution of the malicious software in a virtual machine, as executing them in the PC is too risky. As many malware software is for windows, its installation and the generation of packets could be troublesome.

3.2 Methodology and Rigor

3.2.1 Methodology

The method that is used in this project is waterfall, a simple sequential methodology in which each task must be completed before beginning the following one. This methodology is used for some factors that make it powerful and easy to follow. It has different phases (Figure 3.1): Requirements, Design, Implementation, Verification & Validation and Operation and maintenance.

In the requirements phase, the project is defined according to the limitations and possible obstacles.

In the design and implementation phase, the project is designed to fulfill the objectives and then the practical implementation is done.

In the validation/testing phase of this project, the statistical models will be evaluated and compared, if it doesn't work as planned and there are improvements that could be

3. PROJECT MANAGEMENT

implemented to improve it, the project will go to the implementation phase. This iterative process is going to go on until having a good model.

Finally, the deployment phase is out of scope of the project because there is not enough time.

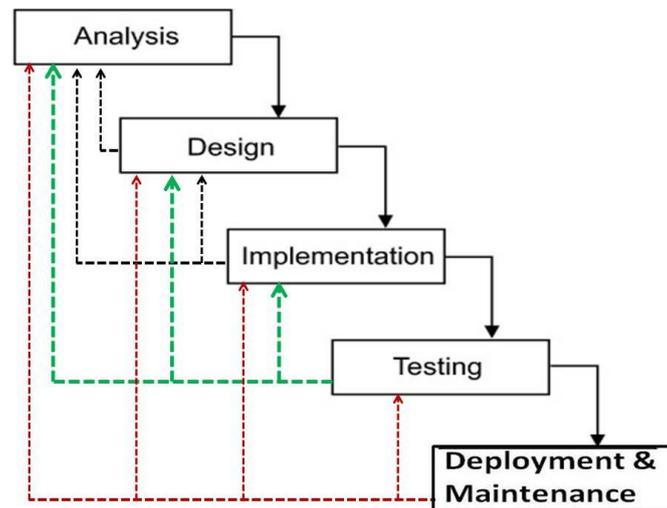


Figure 3.1: Phases of Iterative Waterfall methodology

The reason why this methodology has been chosen is because it allows preplanning every step and putting it in a sequence that ends with the finalization of the project. Another reason for this decision is that there is only one person developing the project, it doesn't make sense to parallelize work as it would be even negative as context costs additional time. For the computer parts, it is planned to use only one computer, and models use a lot of computational resources, so making things in parallel would mean sharing computer processing and it would probably take more time than in the sequential way.

3.2.2 Monitoring tools

Google drive is going to be used for sharing and saving a copy of the different documentation of the project, the code and the datasets to prevent possible disasters. For the time management, Google Calendar is used. It allows to share a calendar and to create notes in each day in order to mark for example the deadlines, meetings with the director... It provides different colors that will be used to identify the different type of tasks and its status. For the conversion to PDF of documents, Microsoft word will be the chosen tool.

3. PROJECT MANAGEMENT

3.2.3 Validation meetings

In favor of ensuring the quality of the project, there will be meetings scheduled with the final project director. He will check that all is going as planned and there are not deviations in the process.

3.3 Time schedule

The time expected for the realization of this project is 4 months in the period compressed between February 2017 and June 2017. This document presents what project management tasks should be done and when.

3.3.1 Task description

All tasks will take all human and hardware resources stated below for their accomplishment.

3.3.1.1 Project Management

It corresponds to the different subtasks and deliverables of the subject GEP. To accomplish this task, 7 deliverables that have different dedication times. The different subtasks are the following:

- 1) Context and scope of the project (24.5 hours)
- 2) Project planning (8.25 hours)
- 3) Budget and sustainability (9.25 hours)
- 4) First oral presentation (6.25 hours)
- 5) Specialization competence (8.5 hours)
- 6) Oral presentation and final document (18.25 hours)

Additional resources needed: Microsoft Office, Ganttter, FIB's racó and UPC's Atenea.

3.3.1.2 Documentation and learning process

The student should get familiar with the software tools, the working environment and gain the enough technical knowledge to make a good quality project. To do that, some learning time is required. This task can be parallelized with the Project Management one.

3.3.1.3 Data acquisition

It consists in the acquisition of data by capturing network traffic. This task has two subtasks:

3. PROJECT MANAGEMENT

- Benign traffic capture: Acquisition of benign traffic by capturing it from the network.
- Malign traffic generation and capture: As malign traffic is not the usual one, it is needed to generate it with malware software. After that, capturing it with the same process as the benign traffic. It will take more time to capture packets than benign ones.

Resources: *softflowd* and internet connection.

3.3.1.4 Data preparation

Also known as “data wrangling”, this task involves all the preprocessing of the dataset: Export of the data, cleaning of missing, featuring, reshaping the data to make it readable for the data mining software and the user.

Resources: *RStudio* will be used for the preprocessing.

3.3.1.5 Modeling

The best statistical model is chosen in this task. There are some subtasks that will be done to achieve this goal.

- Hypothesis & Modeling: In this step, machine learning techniques will be applied. The separation into training, validation and test sets.
- Evaluation & Interpretation: The model created is evaluated and compared with other models previously created.

These subtasks can be repeated as many times as needed until having the best performing model.

3.3.1.6 Optimization

This task is optional depending on the results reported on the previous task. It consists of studying and implementing optimizations to the model in the interest of making it better and more robust.

3. PROJECT MANAGEMENT

3.3.2 Time estimation

Table 3.1 shows the time in hours of each project task.

Task	Time estimation (Hours)
Project Management	75
Context and scope of the project	24,5
Project planning	8,25
Budget and sustainability	9,25
First oral presentation	6,25
Specialization competence	8,5
Oral presentation and final document	18,25
Documentation and learning process	100
Data acquisition	60
Benign traffic capture	10
Malign traffic generation & capture	50
Data preparation	10
Modeling	120
<i>Model Optimization</i>	30
Project Memory	50
Lecture	15
Total	460

Table 3.1: Time estimation per task

3.3 Resources

3.3.1 Human Resources

The most important resource to a project is its people. In this project, only one worker will be required. This person is doing all tasks to complete the project. To be able to finish it in time, the worker should have some previous knowledge in data mining and networking for the sake of interpreting the results.

The time spent for the worker in the project in a week will be between 20 – 30 hours depending on the amount of work needed to be done or if there are long waiting times for some tasks that require high hardware resources that the worker cannot do anything until they are finished. However, if there is a huge workload at some point, spending more hours is also contemplated.

3.3.2 Hardware Resources

- 1) *Acer Aspire V5 laptop with a 15.6" screen*: Hardware tool necessary to develop the project. It is going to be used for all tasks that require a computer to be done.
 - a. OS: Windows 10 and Ubuntu 16

3. PROJECT MANAGEMENT

- b. Processor: Intel Core i5-3317U 1.70GHz x64
 - c. Memory: 8GB
 - d. Capacity: 256GB SSD
- 2) *LG 32" screen (optional)*: Hardware tool that helps reducing the time spent when working with extended screen.
 - 3) *Router*: That provides internet connection to be able to capture packets in the dataset construction task.

3.3.3 Software Resources

- 1) *Softflowd*: It is a flow-based network traffic analyzer that allows exporting data in Cisco Netflow.
- 2) *Nfcapd*: is the Netflow capture daemon of the *nfdump* tools. It reads Netflow data from the network and stores it into files. The output file is automatically rotated and renamed every n minutes - typically 5 min - according the timestamp YYYYMMddhhmm of the interval e.g. *nfcapd.201107110845* contains the data from July 11th 2011 08:45 onward.
- 3) *Weka*: It is a collection of machine learning algorithms for data mining tasks that can be applied to a dataset. It also allows preprocessing and cleaning. It's useful to test algorithms in a faster way.
- 4) *RStudio*: It is an IDE for R programming language that allows executing R scripts and generating markdowns.
- 5) *Microsoft Azure Machine Learning Studio*: Online machine learning tool that allows executing data mining algorithms in the cloud. Useful when R or Weka don't have enough hardware resources or the algorithm takes a lot of time.
- 6) *Gantter*: It is a webpage that allows creating Gantt diagrams, needed for the time schedule representation of the project.
- 7) *Gmail*: Email tool used to maintain a communication between the director and the student during the project.
- 8) *Microsoft Office*: Tools that will be used to write the project memory, do the tables, calculate the costs and hours and do the presentations.

3. PROJECT MANAGEMENT

3.5 Alternatives and action plan

During the development of the project, some deviations might occur because of the number of hours for one task is insufficient and requires more time than the expected. It is possible that the student requires more time for the “welcome aboard” and get familiar with the software tools could affect the project negatively. However, this would be only in the initial part of the project.

Also, the problems stated on the previous report could cause delays in the tasks. These delays have already been calculated and the deadlines have been adjusted with the goal of having an extra time margin.

On the one hand, the fact that the project is being done by only one person, it allows to make a more flexible time planning. If there is a task that has been finished before the expected time, the student could go on with the next one without having to wait anyone. On the other hand, if he requires more time to finish one task, he could take hours of another task in order to guarantee the finalization of the project in time.

3.6 Cost estimation

In this section, there is an estimation of the project economical costs. As the project is not developing anything material, production costs have been excluded. The estimation is going to be done based on the human resources, hardware, software and other costs. The cost estimation has been linked to task description according to Gantt chart.

3.6.1 Human Resources

As the project is going to be developed by only one person, who is doing all the tasks, he is going to adopt different roles along the project. These roles have different costs per hour depending on their responsibility and difficulty. We assume that the people who are contracted have some experience, so the learning task will not be contemplated in the costs.

Task	Role			Cost estimation
	Project Manager	Network Engineer	Data Scientist	
Project management	75h	0h	0h	3.000 €
Data acquisition	0h	60h	0h	2.280 €
Data preparation	0h	5h	10h	540 €
Modeling	0h	0h	170h	5.950 €
Model optimization	0h	10h	20h	1.080 €
Documentation	50h	7h	10h	2.616 €
Total	125h	82h	210h	15.466 €

Table 3.2: Direct costs of each task according to Gantt chart

3. PROJECT MANAGEMENT

	Role		
	Project Manager	Network Engineer	Data Scientist
Cost per hour	40€/h	38€/h	35€/h

Table 3.3: Cost per hour of each role

3.6.2 Hardware

Apart from the human resources costs, there are also costs associated to hardware and material. For this project, only a computer for computational calculus and a router that provides internet connection are essentially needed. However, in the interest of reducing human resources costs, a second monitor (the laptop has one integrated) has been added to reduce tasks time when working dual screen [14].

Product	Units	Price	Useful life	Amortization/hour	Hours	Total amortization	Total Cost
Acer Aspire V5	1	750 €	5	0,08 €	417	31,27 €	750 €
LG 32" Monitor	1	300 €	6	0,03 €	417	10,42 €	300 €
Router	1	40 €	6	0,00 €	60	0,18 €	40 €
Total							1.090 €

Table 3.4: Hardware costs estimation.

For the amortization per hour, it has been considered that a year has 250 working days and for each day there are 8 hours which the people work. For modeling simulations, sometimes leaving the computer turned on all the day is necessary but this has not been contemplated as it wouldn't be the usual thing.

3.6.3 Software licenses

Almost all programs in this project are Open Source and can be used without any additional costs. The only private software is Microsoft Windows 10, which is already included in the computer price listed above, and the suite Microsoft Office 2016 that costs 149€.

For the traffic capture, Ubuntu OS and *softflowd* will be used and both are free. The other software resources used for project management and modeling (*RStudio*, *Weka*, *Microsoft Azure Machine Learning Studio*, *Gantter*) are also free.

3. PROJECT MANAGEMENT

The total costs associated to software licenses are **149€**.

3.6.4 Other costs

In the project, there are other costs that don't enter in hardware, software nor human resources costs categories. These other costs could be classified as following:

- 1) Indirect costs:** In this category, all costs related with the location where the project takes place are included. They are intangible products but necessary for any worker that needs to work in an office. The office is rented for the duration of the project. For the costs, the duration of the project has been considered (4 months) and has been used to calculate the estimated costs of the following table.

Product	Price /month	Estimate Cost
Light	40 €	160 €
Internet	31 €	124 €
Office	400 €	1.200 €
Water	20 €	80 €
Total	491 €	1.564 €

Table 3.5: Indirect Costs

- 2) Contingency:** In the risk section it can be seen that a project has problems and incidences that could cause the project costs increase respect the estimated ones. In order to have some margin and prevent it, it is important to reserve an amount of money for these cases. The amount of money saved will be the 15% of the sum of all the previous costs, direct and indirect ones. In this case we would have a contingency of:

$$1.564€+149€+15.466€+1.090€ = 18.269€ \quad 15\% \text{ of } 18.269€ = \mathbf{2.740€}$$

- 3) Other incidents:** In this section, other incidents that may appear during the project are taken into account. One of the main problems could be that the computer breaks and stops working. This incidence is critical, as the computer is the main tool of the project. Depending on the level of breaking, it would be needed between 2 – 5 hours of an IT Administrator, who works for 20€/hour (Maximum 100€), to fix it or the purchasing of a new computer, which may cost an additional 750€. As the project needs to be finished in 4 months, having to send the computer to the manufacturer for fixing it is not an option, as it could take too much time. There could be also a problem with the duration of the project. It could be that the data scientist need more time in exchange for finishing its task. It may need 20 more hours that cost 35h/hour * 20 hours = 700€. Total: **800€**

3. PROJECT MANAGEMENT

3.6.5 Final budget

To sum up, it is necessary to reflect all the calculated costs together and see what the final estimated costs for the project are. This project is Open Source because is part of the final year project of a student and doesn't aim to take profit of it, that is why profit margin has not been taken into consideration.

Concept	Cost
Human Resources	15.466 €
Hardware	1.090 €
Software Licenses	149 €
Indirect Costs	1.564 €
Contingency	2.740 €
Other Incidents	800 €
Total	21.809 €

Table 3.6: Final budget estimation

3.7 Control management

The control management will be done at human resources level, as the other costs are very fixed and costs are not as big as the human ones.

There will be scheduled sessions with the project director with the intention of having a weekly control of what is being done and by whom. At the end of every week, each worker (in this case only one) should present an imputation of hours sheet where it is indicated what he has done every day and in which task has been working. This is important in order to avoid time deviations and see the current status of the project.

If the project requires more hours, money from other incidences and on the last instance from contingency could be taken to pay these extra hours. There is enough money to cover a large deviation in the worst case.

4 Sustainability

4.1 Economic

The costs have been assessed on the previous chapter. This project is viable as it doesn't have production costs because it doesn't produce any tangible product. It can be seen that almost a 75% of the costs are taken by the human resources, so the project is pretty competitive. As it is a research project, it could be replicated for similar researches in this field and escalated conveniently according to their resources.

4.2 Social

This project is important for two reasons from the social point of view:

- People, who don't have enough knowledge to be saved from hackers, may be infected with malware and see how all their personal information is stolen and in the worst case, leaked. The implementation of the results of this project in an ISP level could drastically decrease the malware traffic on the internet and protecting end-user from them.
- Nowadays, with the recent scandals, internet privacy has gained a more importance as people have realized that their personal information is not safe. Actual monitoring packet tools make use of packet payload inspection (packet content), which means that if it is not encrypted, the information is accessed. However, the method investigated in this project avoids this packet inspection, and that is positive from a user privacy point of view.

4.3 Environmental

Even if this project doesn't have a production goal, the application of the results could reduce the amount of hardware and computational costs that are spent only for deep inspection packet. There is not public information about neither how much energy is spent on malware detection nor how many computers are abandoned after being infected by malware, but it could be said that globally the environmental impact would be positive.

The malware detection would take less hardware resources; this fact has a direct correlation with the energy spent on the process. While having less abandoned PCs in early stages would help to reduce ecological footprint.

4 SUSTAINABILITY

	Project Development	Exploitation	Risks
Environmental	Consumption Design	Ecological footprint	Environmental Risks
	7	14	0
Economic	Project Bill	Viability plan	Economic risks
	4	10	-3
Social	Personal impact	Social impact	Social risks
	8	15	0
Sustainability range	19	39	-3
	55		

Table 4.1: Sustainability Matrix

5. Data acquisition

In order to start working with the supervised algorithms, it is necessary to obtain malware and normal traffic captures. The first ones are generated by some malware when it is being executed on a machine that has been infected deliberately. The second type is usual traffic captures free of malware. To apply the different statistical models and evaluate them, we need a dataset where the traffic is correctly classified between the classes. Following this step properly can make the difference between realistic results and biased results without any value for the future work. Even though the methods for capturing malware traffic and benign traffic are different, we are going to apply some processing to them with the aim of finally getting them into the same format.

5.1 Obtaining normal traffic captures

In this phase, we wanted to capture traffic that didn't contain any type of malware so a safety environment was set for the sake of avoiding possible infiltrations of this type of traffic that could bias the results. This bias could occur when classifying malicious traffic as normal and therefore giving wrong information to the algorithms that could end into a bad model.

The safe environment selected to capture the benign traffic was a company's network, which name cannot be revealed for privacy reasons, of around 30 computers connected to internet plus an anti-virus software up to date. This network is also in possession of a Firewall Fortinet that has a huge database which is updated in a weekly basis with the latest malicious webpages and blocks all the dangerous traffic that attempts to attack the vulnerable ports. This environment is desirable because it could allow us to know the project results and behavior of the solution on a real enterprise environment.

To capture the traffic, an SSH connection to the firewall with putty was established during 4 hours in three days while running the command to sniff all traffic passing through the internet interface and saving the output into a file. This file is then processed with a program named fgt2eth.exe, which is provided by the firewall's company [15] and converted into a .pcap file that could be read for many network tools. *Putty* software was gathered.

5.2 Obtaining malware traffic captures

Unlike regular traffic that can be easily generated by just sniffing a lot of machines connected and working through the internet, malware traffic has some particularities that make it difficult to obtain in huge quantities:

- On a huge network, there is one/a few machine/s infected.
- It is difficult to isolate as there are other processes running on the computers that could also generate traffic to the internet.
- Some type of malware like Trojans, don't generate too much traffic as their behavior could.

5. DATA ACQUISITION

Also, there are a lot of different malwares even of the same type [16]. As we want the model to be robust, we should ensure to take many different viruses so we increase the variability of the samples hence after training the model with many different malwares we reduce the prediction bias towards one malware in particular.

Malware Name	Type	Size .pcap (MB)
BunddleApp	Trojan	259
DrAutoit	Trojan	116
DridexA	Botnet	80
Dyreza	Trojan	11
Kelihos	Botnet	190
LuminosityLink	Trojan	59
MalwareLocky	Ransomware	256
Miuref	Trojan	215
Necurse	Botnet	57
Neris	Botnet	20
NewStorm	Trojan	774
OpenCandy	Adware	111
Pony_Botnet	Botnet	7,7
PurpleHaze	Botnet	235
TAOBAO	Adware	320
Tinba	Botnet	15
TrojanMSIL	Trojan	15
TrojanSpyBanke	Trojan	2,9
Yakes	Trojan	887
WauchosZ	Trojan	10
WisdomEyes	Trojan	31
WormNetspy	Trojan(Spyware)	1100
TOTAL		4771,6

Table 5.1: List of Malware Captured Used for the Dataset

As stated previously, malware traffic is harder to capture, which is the reason for executing some of them on a virtual machine and taking some others from *stratosphereips* [17] and *contagiondumb* [18], in order to collect a considerable amount of different malware captures. After that, we could gather 4.7GB of malware captures from the different types of viruses specified in the Table 5.1.

6. Data preparation

6.1 Capture processing

We need to be able to apply the different machine learning algorithms successfully, so we first need to prepare the data to be “readable” for them. On the previous phase, the different traffic captures were obtained but there was a lack of a router that could produce Netflow traces automatically by itself; the traffic was captured on .pcap extension.

As one of the main goals of the project consisted on knowing how good is the predictive behavior of the different models on Netflow traffic, we first need to generate the Netflow traces from the .pcap files. We simulated that ideal scenario by using some network tools available on Ubuntu.

The environment was prepared by installing the operating system Ubuntu in a virtual machine and the three necessary tools used for this process (*softflowd*, *nfcapd* and *nfdump*). *Softflowd* semi-statefully tracks traffic flows recorded by listening on a network interface or by reading a packet capture file, so we pass our .pcap files as parameter and the IP and port of the host who will receive the flows. In this case as we want the same Ubuntu machine that is executing the program to collect them, we are going to put the loopback ip 127.0.0.1 (localhost) and an arbitrary non well-known port as 9995.

Before executing *softflowd*, it is necessary to set up the host to listen to the port used to send the flows; we open a second console and executed the tool *nfcapd* specifying the same port and host as in *softflowd*. This Netflow capture daemon reads Netflow data from network and stores it into files. After having executed *nfcapd*, we run *softflowd* and all the Netflow data is stored in files. After doing this step, we have the malware and normal traffic on different files but still not in a “readable” extension.

Finally, we are going to process the files saved by *nfcapd* and transform them into a csv (Comma Delimited Values) file. This format is one of the most used ones in the machine learning paradigm as it allows easily separating the different columns and it is readable for the tool we are going to use afterwards. To do this transformation, we use a tool included in the same package as *nfcapd*, *nfdump*. It reads the Netflow data from files stored by *nfcapd* and processes the flows according the options given. For the options, we select a custom format that gets the usual features of Netflow (Duration, Protocol, Source Address, Source Port, Destination Address, Destination Port, Flags, TOS, # of packets, Bytes, # of Flows) but also some features product of the original ones that *nfdump* calculates for us that are Bytes per Second (Bytes/Duration), Packets per Second (Packets/Duration) and Bytes per Packet (Bytes/# of Packets). We redirect the output of the command into a .csv file.

```
nfdump -r nfcapd.201709220859 -o 'fmt:%td %pr %sa %sp %da %dp %flg %tos %pkt %byt %fl %bps %pps %bpp' -q
```

Figure 6.1: *Nfdump* command used to convert *nfcapd* files

6. DATA PREPARATION

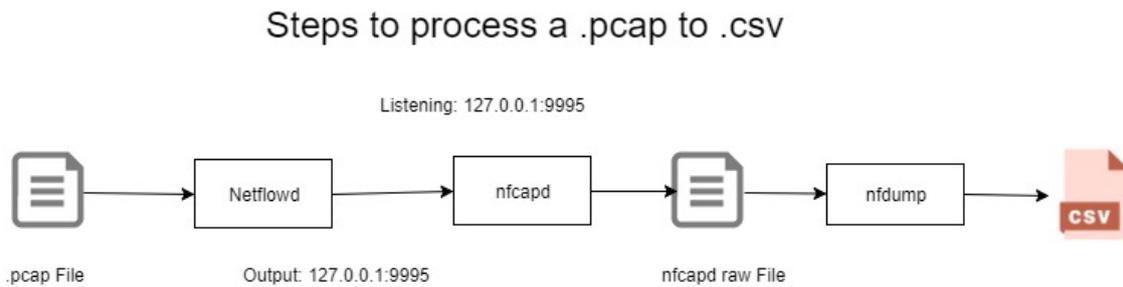


Figure 6.2: Steps to transform .pcap to a CSV Netflow dataset.

6.2 Data Cleaning

After having the csv files, they are now readable by the tool we are using in order to do the models and analysis; in our case we are using *RStudio*. However, there are still a few things to do before we can apply the different machine learning techniques. Figure 6.3 shows what can be found in the data after being read, in this case for the traffic malware csv file.

```
> summary(Malware)
  v1      v2      v3      v4      v5      v6      v7      v8      v9
0.001 : 306475 UDP :2486526 192.168.1.115:1481435 Min. : 0 192.168.1.115:1035717 53 :1120027 ..... :2487567 Min. :0 Min. : 1
0.002 : 396085 TCP :1488667 8.8.8.8 : 537368 1st Qu.: 53 8.8.8.8 : 545038 25 : 486379 .AP_SF : 444386 1st Qu.:0 1st Qu.: 1
0.003 : 193464 ICMP : 876 8.8.4.4 : 501896 Median :51198 8.8.4.4 : 504792 80 : 371621 ...S. : 399527 Median :0 Median : 2
0.018 : 135382 IGMP : 165 10.0.2.104 : 370597 Mean :33427 10.0.2.104 : 340488 5355 : 201338 .A.S. : 148597 Mean :0 Mean : 4
0.000 : 129869 1494 : 3 192.168.1.118: 177208 3rd Qu.:58227 224.0.0.252 : 201340 1900 : 57086 .APRSF : 143536 3rd Qu.:0 3rd Qu.: 4
0.021 : 92070 (other): 8 (other) : 907730 Max. :65535 (other) :1348859 (other):1739783 (other): 352621 Max. :0 Max. :332380
(other):2920417 NA's : 107517 NA's : 107528 NA's :107528 NA's : 107528 NA's : 107528 NA's :107528 NA's :107528 NA's :107528
  v10      v11      v12      v13      v14
152 : 389078 1 :3976165 0 : 489313 0 :1486807 50 : 792461
50 : 271606 M : 69 135 : 361121 500 : 277753 125 : 218520
125 : 265799 NA's: 107528 1.0 : 90838 1000 : 273967 57 : 182797
114 : 121436 400000 : 86120 666 : 187353 47 : 123429
228 : 114973 200000 : 71154 8 : 150970 114 : 120493
(other):2813342 (other):2877688 (other):1599384 (other) :2538534
NA's : 107528 NA's : 107528 NA's : 107528 NA's : 107528
```

Figure 6.3: Summary of data before cleaning.

We realize that there are missing values that are represented with a “NA”, this means that when the data was captured it was not possible to get the value. However, these missing values could be important for the data quality of the dataset and it is recommended to try to apply an imputation method like PCA or MCA [19] instead of just deleting them as we could lose important samples. But in this case, after analyzing the rows with missing values, it has been found that those are the same for each feature so they don't add any extra information to the dataset and can be deleted. We also considered as errors flows with duration 0 and protocols different of TCP, UDP, ICMP and IGMP.

Additionally, columns have been named for a better understanding of the dataset. TOS column was deleted as it only presented one value (0) for both malign and normal traffic so it was irrelevant for the analysis.

6. DATA PREPARATION

Furthermore and going on with the data cleaning, some features like the ones that carry out the ports information are being treated as numerical variables when they are factors, so we make the transformation. We do the same but in the other direction for Duration, Bytes, BPS, PPS and BPP that were treated as factors instead of numerical.

We also add another column that will be the class. This is very important as the class allow us to train the models or validate the results comparing the class with the model predictions. We are going to copy the malware dataset and we are going to prepare one for the binary response, classifying all of them as malware, but also a second dataset for a multiclass response, differentiating between Malware-Trojans and Malware-Botnets so we can evaluate the two types of responses in the future.

6.3 IP Selection

Currently, our dataset has two features which are IP Source and IP Destiny. Each one of these fields contains an IP that can be a public IP or a private IP address [20]. As private IPs are not unique and are equal on each network, they should not be used for the analysis. We are going to compare the two features and select only the public IP of both, this public IP will be added as a new column to the dataset while the other two are deleted. The public IPs will not be relevant on the first part of the project but they will be more relevant later on for the model optimization.

```
if len(sys.argv) == 1:
    print("Please use as argument the name of the file to compare, the two IP rows should be in 3th and 5th")
else:
    numLinea = 1
    with open(sys.argv[1], 'rt') as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='"')
        next(csvfile)
        for row in reader:
            sys.stderr.write("Reading line #" + str(numLinea) + "\n")
            ipOrigen = ipaddress.ip_address(row[2])
            ipDestino = ipaddress.ip_address(row[4])

            if not ipOrigen.is_private:
                print(ipOrigen)
            elif not ipDestino.is_private:
                print(ipDestino)
            else:
                print("NA")
            numLinea = numLinea + 1
```

Figure 6.4: Program that selects the Public IP from two columns

6.4 The Dataset

After having cleaned and performed the transformations stated on the previous points, we have 180.067 observations of normal traffic and 3.845.870 observations of malware traffic for the binary response. We can see that if we join them as they are, it would produce an unbalanced class dataset. Some studies have concluded that working with class imbalance

6. DATA PREPARATION

could generate several issues to the results [21]. Standard classifiers in these cases tend to ignore the classes with fewer observations as they are overwhelmed by the biggest ones.

There are some clever re-sampling methods that could help us decrease and fix the class imbalanced problem. Some studies have analyzed the different solutions like random oversampling by generating new synthetic samples for the small class or random /directed under-sampling concluding that random under-sampling had a better performance than oversampling on decision trees algorithms [22]. We reduced the benign dataset using the random under-sampling getting a subset of 180.067 observations.

Class	# of Observations
Malware	180.000
Benign	180.000
TOTAL	360.000

Table 6.1: Binary Dataset after under-sampling

For the multiclass dataset, we did also under-sampling as provided on table X.

Class	# of Observations
Malware-Botnet	90.000
Malware-Trojan	90.000
Benign	90.000
TOTAL	270.000

Table 6.2: Multiclass Dataset after under-sampling

7. Algorithms Used

In this section we are going to introduce the supervised algorithms that have been chosen to be applied on the dataset with the purpose of creating the models. The selection of the algorithms has been done by having into account factors like if they supported Multi class (Necessary for our classification), their interpretability and if they accepted a mixed predictor. The following table shows the algorithms that have been proven to be versatile applied in many data science projects and they have been used on past literature regarding traffic classification.

Classification algorithms	Multi-class Support	Mixed Predictor Support (numeric/categorical)	Interpretability level
Decision Trees (C5.0, Random Forests)	Yes	Yes	Easy
k-Nearest Neighbor	Yes	No	Hard
Naïve Bayes	Yes	Yes	Easy
Support Vector Machines	No	Yes	Hard (if the kernel is not linear)

Table 7.1: Versatile machine learning algorithms and factors.

We wanted an algorithm that had multiclass support but easier to interpret, so k-Nearest Neighbor and SVM (Support Vector Machines) were discarded.

The following supervised algorithms have been implemented:

- Naïve Bayes
- Random Forest
- C5.0 Decision Tree

7.1 Naïve Bayes

Naïve-Bayes is a method based on the Bayesian theorem. It analyses the relationship between each feature of each observation with the class of that instance to compute a conditional probability between the feature values and the class.

Let's assume X is a vector of characteristics (x_1, x_2, \dots, x_n) , for learning we estimate the likelihood of our data or probability that observation (x_1, x_2, \dots, x_n) belongs to class c_i [x_i is feature i of observation x]:

$$p(x_1, x_2, \dots, x_n | c_i)$$

It also calculates the simply proportion of elements in class j , $P(c_j)$.

7. ALGORITHMS USED

One of the key points used on Naïve-Bayes method that makes it viable for huge datasets is the assumption that the attributes are independent, so the probability results on the product of the probabilities of each attribute. This assumption makes it easier to compute, getting the following equality:

$$P(x_1, x_2, \dots, x_n | c_j) = \prod_i P(x_i | c_j)$$

For classifying, given an observation (x_1, x_2, \dots, x_n) we should look for the class that has the maximum probability of belonging to that class, so the class assigned to a new observation is:

$$C_{newobs} = \operatorname{argmax}_{c_j \in \mathcal{C}} \frac{P(x_1, x_2, \dots, x_n | c_j) \times P(c_j)}{P(x_1, x_2, \dots, x_n)} = \operatorname{argmax}_{c_j \in \mathcal{C}} P(c_j) \times \prod_i P(x_i | c_j)$$

The formula stated above can only be applied if the attributes X_i are qualitative or nominal. Probabilities are estimated by counting the frequency of a value in a feature. Numerical variables could have a huge number of different cases so a frequency distribution cannot be estimated. In these cases there are two ways to cope with this problem. The first one consists on discretizing or making these numerical variables qualitative dividing them into intervals hence reducing the number of different cases. The second way named kernel density estimation, is modelling the attributes with a continuous probability distribution using multiple Gaussian distributions, which is more effective than just using one.

7.2 Decision Trees

The main idea of a decision tree is to understand the relationships in large collections of data, i.e., multiple observations of variables, whose structure is difficult to determine. The main way a decision tree is applied is by using the properties of an observation as questions and going over the tree until we reach a leaf identifying the observation in a certain class. The structure of a decision tree is built by a machine learning algorithm on the training phase.

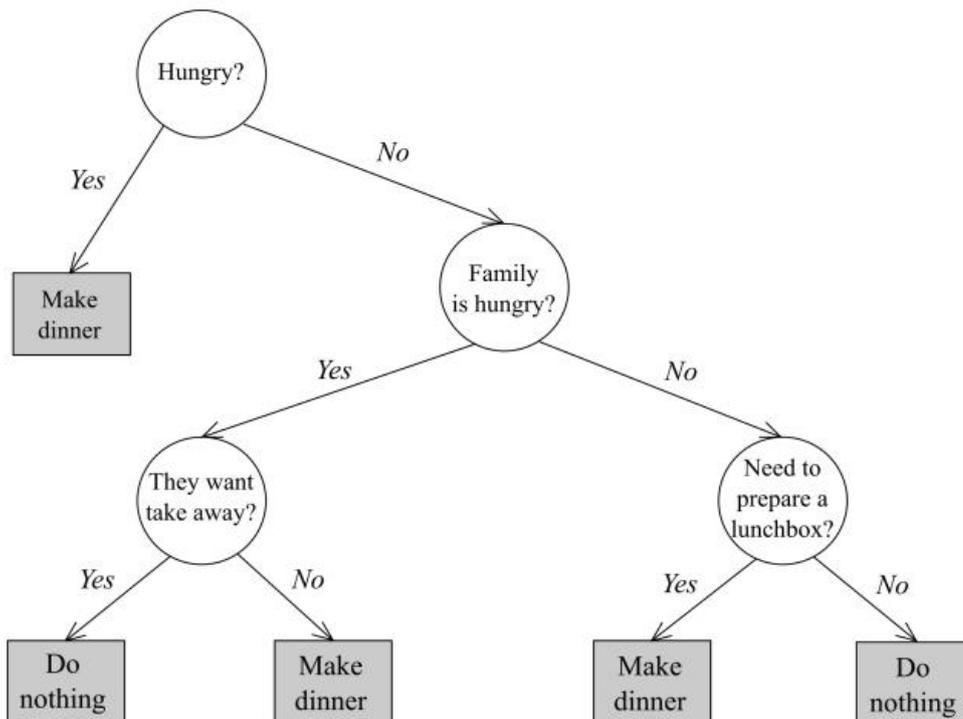


Figure 7.1: Example of a decision tree where leaves are the class.

The algorithms that we are using are based on these decision trees.

7.2.1 Random Forest

By their definition [23], random forests are a combination of decision tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. Using a random selection of features to split each node gets lower errors compared to Adaboost [24].

For prediction, each decision tree predictor votes for a decision that is pooled with the other decision trees' responses, getting as definitive the decision with the higher rate.

7. ALGORITHMS USED

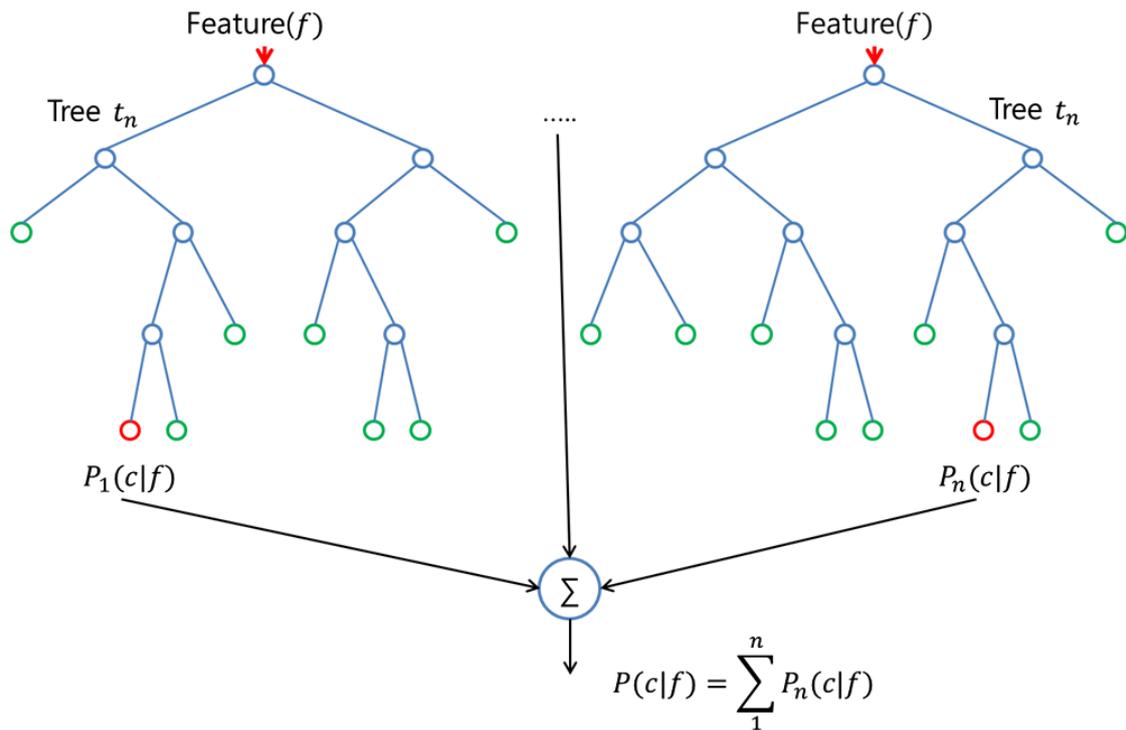


Figure 7.2: Example of how random forests make a decision.

This algorithm is usually more robust against noise when more decision trees are built and, in the same way, the higher the number of decision trees, the higher the accuracy we get. Making more trees has a greater time consuming cost, so we should maximize the trade-off between time/accuracy.

Algorithm 3 Random Forest in Pseudo-code

- 1: **procedure** RANDOM FOREST
 - 2: **for** 1 to T **do**
 - 3: Draw n points D_l with replacement from D
 - 4: Build full decision/regression tree on D_l
 BUT: each split only consider k features, picked uniformly at random
 new features for every split
 - 5: Prune tree to minimize out-of-bag error
 - 6: **end for**
 - 7: Average all T trees
 - 8: **end procedure**
-

Figure 7.3: Random Forest in Pseudo-code.

7. ALGORITHMS USED

7.2.2 C5.0 Decision Tree

The last method we are going to apply is C5.0 Decision Tree, a successor of the C4.5 Decision tree presented by Quinlan in [25] that has been already used on several papers [<http://personals.ac.upc.edu/pbarlet/papers/autonomic-retraining.jnsm2013.pdf>] proving its better performance than its antecessor at classifying traffic.

Several papers have analyzed it highlighting that this algorithm is faster than the C4.5 algorithm, having shorter training times. It also manages the memory usage more efficiently. Its rule sets have lower error rates on unseen cases and also builds smaller decision trees in comparison with C4.5. Moreover, C5.0 allows us to weight different cases and misclassification types and it includes an option that automatically winnows the attributes to remove those that might be unhelpful.

8. Model Evaluation

8.1 Metric

In order to measure the quality of the different models, we need a metric that allows us to compare the different model performances. There are a lot of metrics that could be used for evaluating a machine learning algorithm (RMSE, LogLoss, Accuracy, ROC Curve...). The metric chosen for this project has been the accuracy, which could be expressed in the following way:

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

where:

Assuming the positive class is the Normal traffic and the negative class the malware traffic.

- TP (True Positives): The number of correctly identified normal traffic flows.
- TN (True Negatives): The number of correctly identified malware traffic flows.
- FP (False Positives): The number of incorrectly identified normal traffic flows.
- FN (True Negatives): The number of incorrectly identified malware traffic flows.

8.2 K-fold Cross Validation

After knowing the metric that we are going to use, we need an estimator to get it. Instead of using the conventional method that would mean splitting the dataset into 70% training and 30% validation, which could cause a significant lost in testing capability, we are using a technique named cross-validation.

There are two types of cross-validation: exhaustive and non-exhaustive. The main difference between each other is that exhaustive learn and test all possible ways to divide the original sample into a training and validation set while the non-exhaustive doesn't compute all the ways of splitting the original sample.

We are using a type of non-exhaustive cross validation that is k-fold cross-validation. This method for estimating the accuracy of the models consists on splitting the dataset into k subsets. One single subsample is saved for using it as validation data, while the other k-1 subsets are used for training data. This process is repeated k times while saving a different subset for validation each time.

The k results, in this case the accuracy obtained by this process, are averaged to produce a better estimation with a lower variance than a single hold-out set estimator. The advantage of this method over the conventional one is that all data is used for training and testing while each observation is only used once for validation.

K-fold Cross Validation: Pseudocode

1. Divide the data into K subsets (folds)
2. For each of the K folds
 - 2.1 Train the model with all folds except one that has never been excluded before.
 - 2.2 Test the error with the excluded fold.
 - 2.3 Store the validation error.
3. Average the K validation errors stored.
4. Compare the validation errors of the different models and pick the lowest.

Figure 8.1: K-fold Cross Validation Pseudo-code

8.3 Binary Response

First of all, we are going to compare the different models and their predictive performance when facing a binary class (Normal or Malware). For the three algorithms, it doesn't make sense to use the feature of the Public IP as leased IPs usually change with time and including it would cause a biased training dataset that would classify normal flows that have an IP that was used by malware when the model was trained, incorrectly. For validating the models we are going to apply a 10-fold cross validation that is the most used one.

8.3.1 Naïve Bayes

After applying the 10-fold Cross Validation to Naïve Bayes, we see that it is the fastest algorithm of the three types used as it only calculates probabilities and doesn't need to create any costly data structure. We can see in the Figure 8.2, the results obtained for each one of the folds and the average accuracy of the model.

8. MODEL EVALUATION

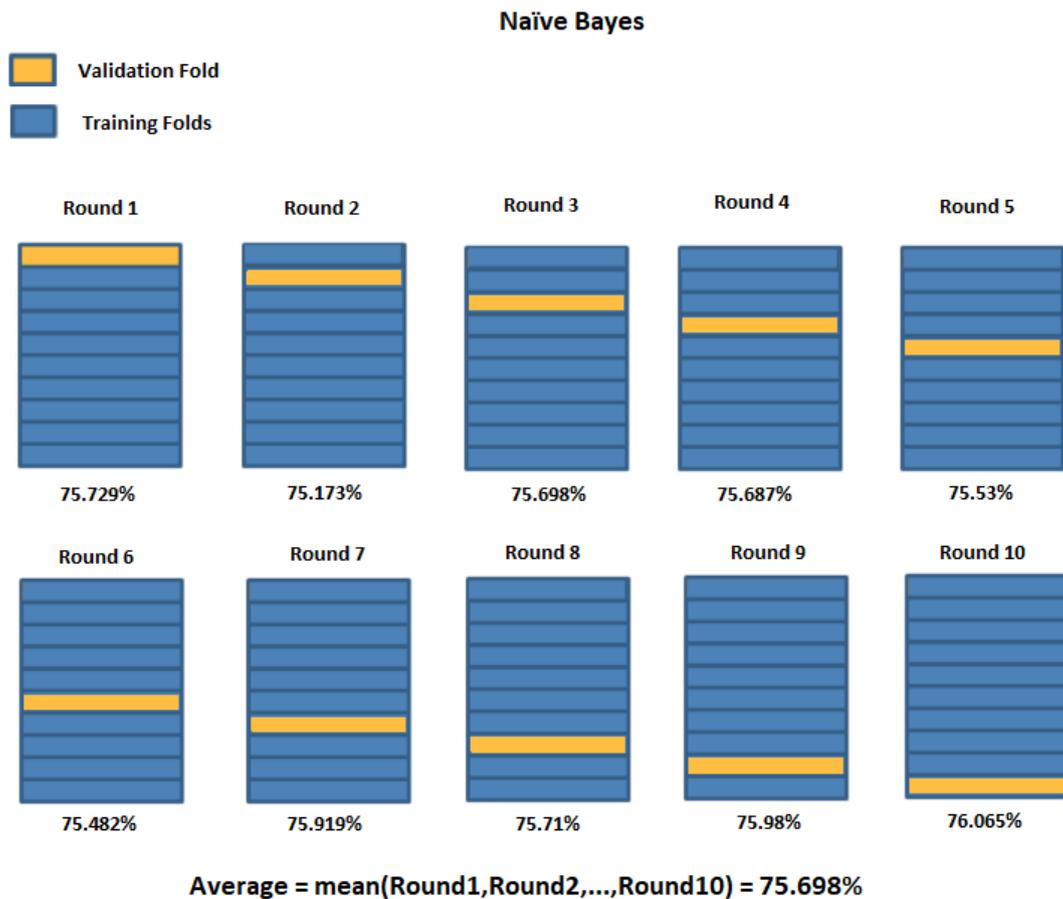


Figure 8.2: 10-fold Cross Validation of Naïve Bayes

As we can see, even if the time cost of training the model is really low, the accuracy is far away from the other two algorithms while having a 75.698% of global accuracy. Training phase on Naïve Bayes took 12.32 seconds being the lowest of the three algorithms.

8.3.2 Random Forest

For the second algorithm used, we are facing an issue that random forest implemented in R has and it is that they don't accept qualitative features with more than 32 levels [26], so we are excluding the two features regarding the port origin and port destination from the model.

The R implementation allows us to specify the number of decision trees built for the majority vote. Figure 8.3 shows how the accuracy increases when we increase the number of decision trees.

8. MODEL EVALUATION

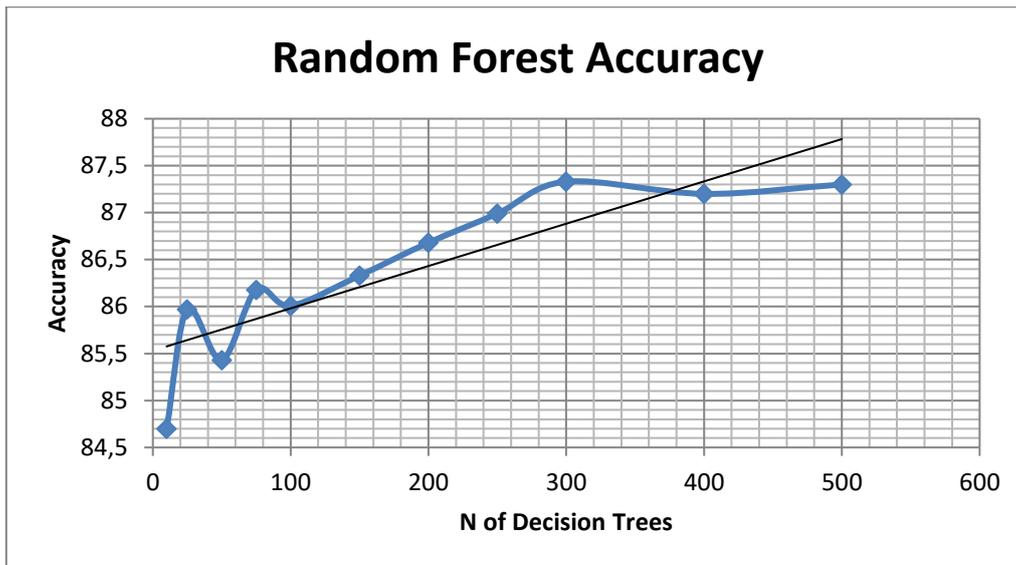


Figure 8.3: Accuracy of Random Forest for different number of decision trees.

As we can see, the accuracy increases until we get into a certain number of trees (Approximately 300 decision trees). However, it doesn't increase anymore but it decreases a little bit instead. By the nature of this algorithm, which takes random seeds to grow the trees, we have executed each model 5 times and have done the mean of the medium values. Figure 8.4 represents how time increases linearly with the number of trees.

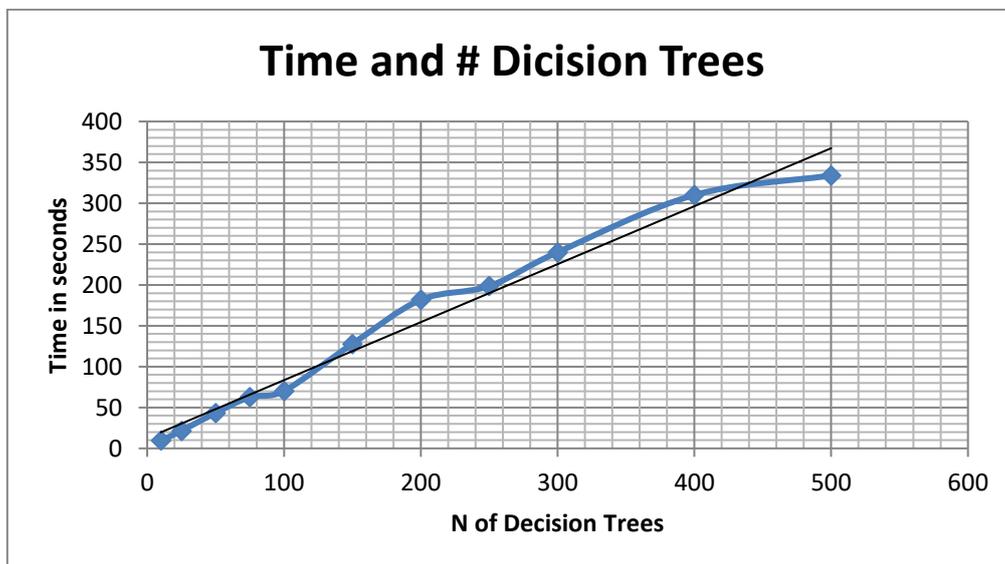


Figure 8.4: Linear relation between time and number of decision trees.

With a linear relation, it doesn't make any sense to increase the number of decision trees after 300 as it has maximized the trade-off between accuracy and time. The final accuracy is 10% greater than in Naïve Bayes.

8. MODEL EVALUATION

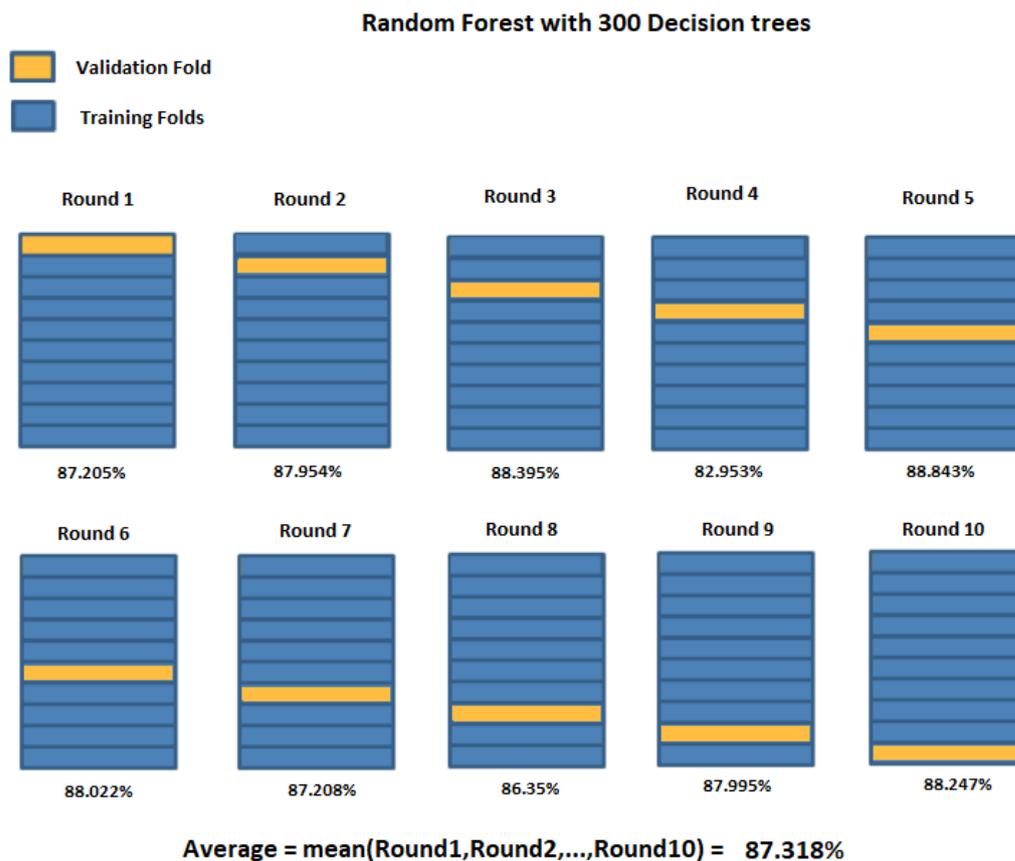


Figure 8.5: 10-fold Cross Validation of Random Forest with Number of Trees = 300.

Finally, we estimate the most significant variables of the model by computing the mean decrease of accuracy over all out-of-bag errors (Prediction errors) averaged, when a given variable is permuted after training. Figure 8.6 shows how BPP and Duration are the most significant ones while the Protocol is the least important.

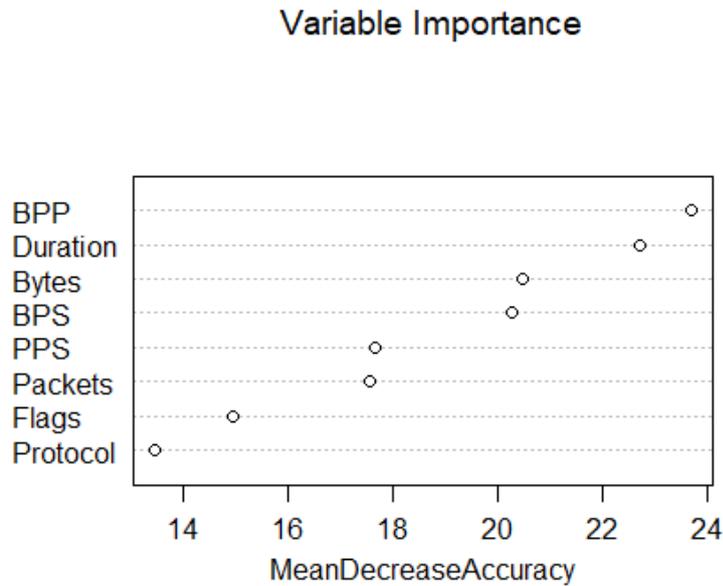


Figure 8.6: Variable Importance for Random Forest.

8.3.3 C5.0 Decision Tree

Concerning the implementation of the C5.0 decision tree, we are going to use the R package 'C50' [27]. This package allows us to fit classification tree models using Quinlan's C5.0 algorithm and also perform boosting iterations (trials) to improve the trees and give them more accuracy. Boosting builds an ensemble of classifiers combining them to get a robust one. The final classifier aggregates the weak classifiers by voting, but each classifier's vote is weighted in function of its accuracy. On previous literature, it has been proven that a boosted C4.5 or the newest version C 5.0 outperforms the same version without boosting [28] for binary class. In the Figure 8.7, we test the accuracy of the model for different number of boosting iterations so we can maximize the performance of the model.

8. MODEL EVALUATION

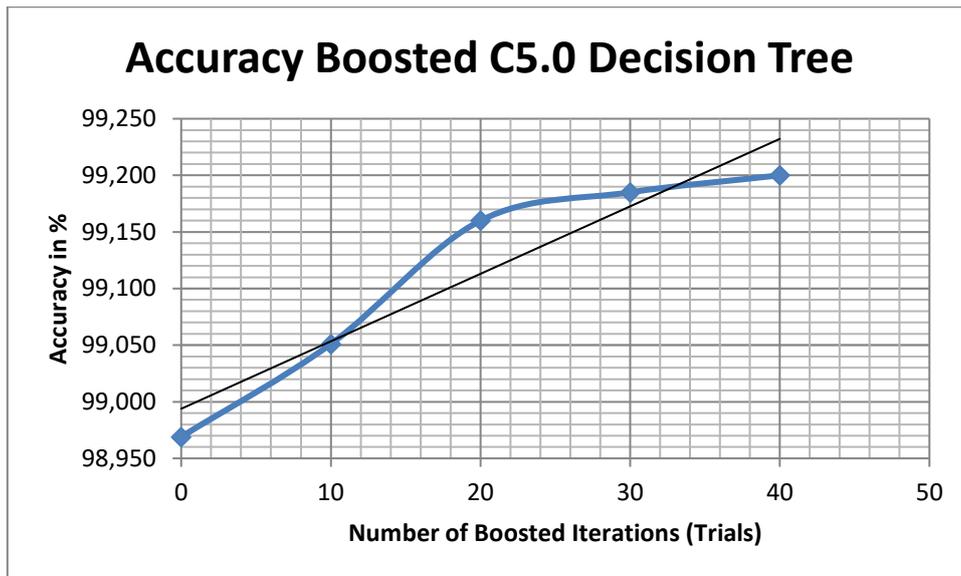


Figure 8.7: Accuracy of Boosted C5.0 Decision Tree.

The performance of the model increases after applying boosting iterations. At 20 trials, the algorithm has just a tiny improvement for the time it takes to perform for higher number of trials (Linear cost) so we are going to use 20 for the following parts of the project. We can see in Figure 8.8 the accuracy of each of the folds and the average one for a Boosted C5.0 Decision Tree with 20 trials.

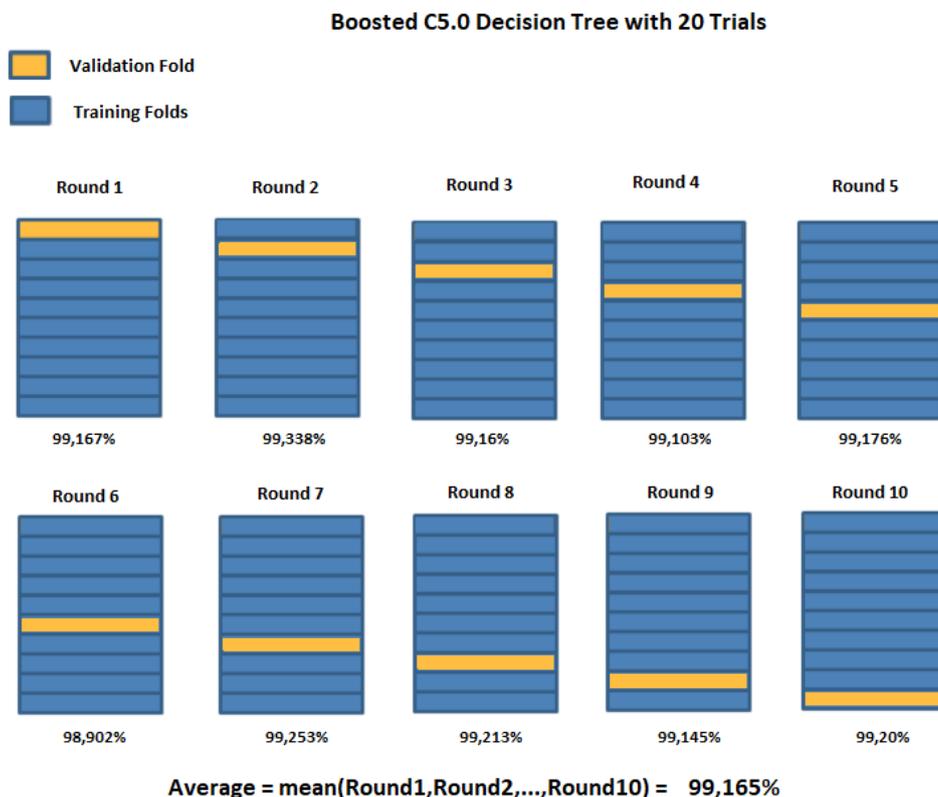


Figure 8.8: 10-fold Cross Validation of Boosted C5.0 with Number of Trials = 20.

8. MODEL EVALUATION

8.3.4 Summary

We have performed the validation of the different machine learning algorithms and now having the metric, we can compare in the interest of knowing which model has the best performance for a binary classification of malware traffic by just using Netflow. Figure 8.9 shows a bar plot with the different models applied with their accuracy obtained by the mean of the 10-fold cross validation. To point out, we also include a second bar plot comparing the different training times for each model as in a real scenario it is important to keep the model updated fast enough with the latest patterns performing retraining of the model.

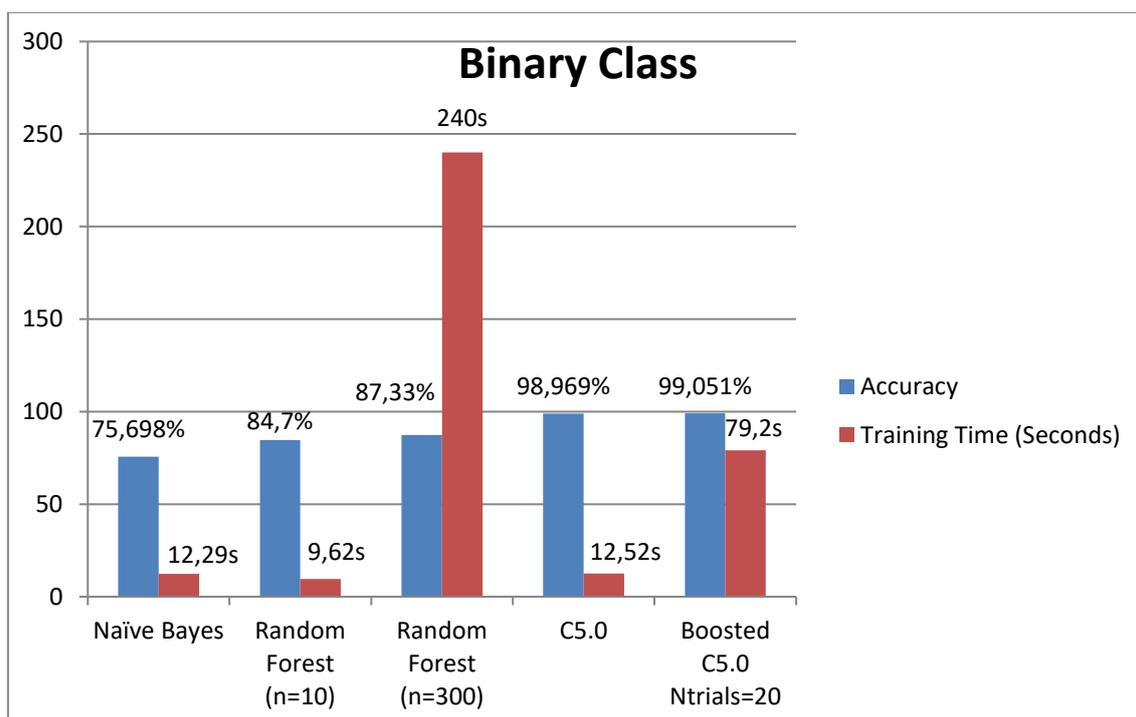


Figure 8.9: Comparison for different models' performance for binary class.

8.4 Multiclass Response

After having evaluated the different models for the binary class, we are going to evaluate them with a multinomial classification where instances are classified in one of the more than two existing classes, in our case the three classes are Normal, Malware-Trojan and Malware-Botnet. As it was explained on Section 6.4, we are going to work with a balanced dataset where all classes have the same amount of instances. One of the main goals of this experiment is to know if there are differences on the behavior patterns between Trojans and Botnets that could be detected and predicted by a machine learning algorithm.

For coherence with the previous study, we are going to validate the models also with 10-fold cross validation.

8. MODEL EVALUATION

8.4.1 Naïve Bayes

On the binary classification, Naïve Bayes achieved the worst performance of all the algorithms. Now, we are going to study its performance when trying to classify with three classes. After applying the 10-fold cross validation, we obtained a global accuracy of 57.103%. These are poor results compared with the same method on the binary classification. Figure 8.10 shows the confusion matrix of one of the folds that could give us an idea of what is happening:

	MALWARE-BOTNET	MALWARE-TROJAN	NORMAL
MALWARE-BOTNET	2210	6200	602
MALWARE-TROJAN	195	8411	33
NORMAL	489	6813	8026

Figure 8.10: Confusion matrix of Multiclass Naïve Bayes.

In a confusion matrix, the columns are the predictions while the rows mean the correct values. The accuracy can be obtained by dividing the correct predictions, which are always the diagonal of the matrix with the total number of observations. In this case, it shows that the model is classifying many botnets and normal traffic as Trojans.

8.4.2 Random Forest

For random forest we are going to follow the same approach as in the binary classification. We are going to test the model for 10 decision trees but also with the best configuration that was 300 decision trees. Figure 8.11 shows the confusion matrix obtained for $n = 10$.

	MALWARE-BOTNET	MALWARE-TROJAN	NORMAL
MALWARE-BOTNET	3906	41	4982
MALWARE-TROJAN	363	5784	2649
NORMAL	409	59	14786

Figure 8.11: Confusion matrix of Multiclass Random Forest $n = 10$.

We can observe that now curiously, the model has a lot of false positives (Trojans and Botnets classified as Normal traffic). The results are pretty different to the previous ones and the accuracy is 74.10%.

For the $n=300$, the accuracy increases 0.06% from the previous execution with 10 trees, getting 74.16%. Figure 8.12 shows a confusion matrix pretty similar to the previous one.

	MALWARE-BOTNET	MALWARE-TROJAN	NORMAL
MALWARE-BOTNET	3891	42	4996
MALWARE-TROJAN	358	5788	2650
NORMAL	414	59	14781

Figure 8.12: Confusion matrix of Multiclass Random Forest $n = 300$.

8. MODEL EVALUATION

8.4.3 C5.0 Decision Tree

After looking the results of the binary classification for C5.0 and boosted C5.0, we want to know if they perform as good as they did. To make the comparisons, we are going to use the same configurations chosen to make the bar plot of the Figure 8.9, C5.0 and C5.0 with 20 boosting iterations (Trials).

On Figure 8.13, we observe the confusion matrix of the C5.0 models without applying any booster iterations; we can see that comparing with the previous models it classifies correctly almost all observations. The accuracy of the model is 98.25% which is pretty similar to the one we had for binary response but a little bit lower.

	MALWARE-BOTNET	MALWARE-TROJAN	NORMAL
MALWARE-BOTNET	8751	149	38
MALWARE-TROJAN	180	8436	87
NORMAL	28	94	15216

Figure 8.13: Confusion matrix of Multiclass C5.0 without boosting.

On the other hand, after executing the same model with 20 boosting iterations we obtain an accuracy of 98.22%, worse than without boosting.

	MALWARE-BOTNET	MALWARE-TROJAN	NORMAL
MALWARE-BOTNET	8697	212	29
MALWARE-TROJAN	164	8460	79
NORMAL	24	78	15236

Figure 8.14: Confusion matrix of Multiclass Boosted C5.0 with 20 trials.

8.4.4 Summary

We have observed how the same models have less accuracy when they try to predict a multinomial class, which makes sense as it is more complex than a binary one. Figure 8.15 shows us a bar plot with the different accuracies of all the models. We can observe how boosting doesn't improve the accuracy in this case but it also manages to decrease it. Execution times are a little bit higher than in the binary class for its added class.

8. MODEL EVALUATION

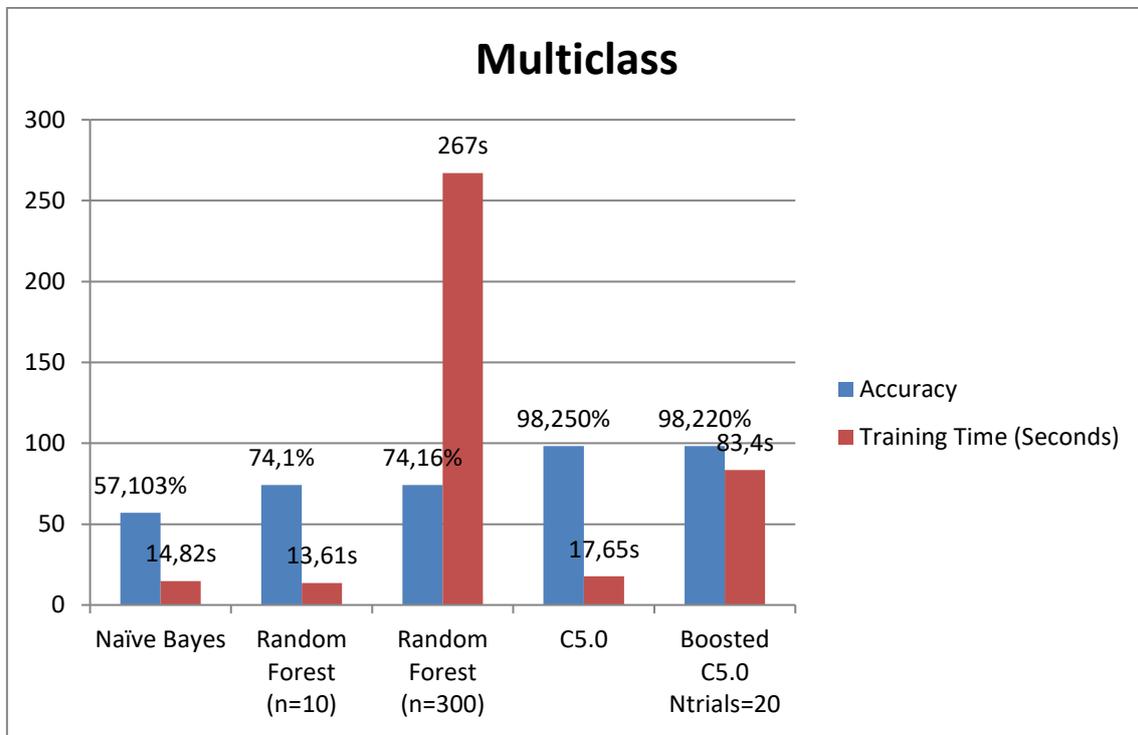


Figure 8.15: Comparison for different models' performance for multiclass.

9. Optimization

One of the main problems of statistical malware traffic classification is that models have been trained with the most recent malware samples which were attacking at that moment, but malware programs are constantly changing, every day there are new malware programs with different attack patterns. There also exist metamorphic viruses that can transform their behavior based on the ability that they can translate, edit and rewrite their own code. This type of malware is especially dangerous for computers because antivirus scanners cannot detect a type of virus that changes its internal structure on every computer it infects being unable to obtain a clear signature. That is why our models have a lower performance when a new malware is presented. To cope with an environment that change constantly, we need to apply model retraining with the latest malware flows to keep it updated.

On this last section, we are going to study how we could use the feature Public IP, that we obtained on Section 6.3 and was not useful for statistical classification.

9.1 Blacklist Concept

Traffic Blacklists are a basic control access that could contain IP addresses, hashes, domain names, URLs, etc. These blacklists contain potentially dangerous addresses, which traffic should be denied. These blacklists can be implemented locally in various points of a network architecture depending of what we want to block, for example in a DNS server to block domain names or in a firewall to block IPs.

There are private blacklist maintained by companies for their own products like antivirus, firewalls or even for search engines to avoid displaying malicious webpages. But there are other organizations that maintain public blacklists that can be downloaded or consulted on their webpage or via API REST performing a GET to the service.

9.2 Using IP blacklists for optimization

Usually when a Trojan infects a computer, it is expected that the intention of the attackers is stealing information of that machine. One of the main things that they will do is to program the malware to send it to some Domain Name System (DNS) address or a pool of IPs where they have machines waiting for the data. However, it is not rare for a malware to be spread into the network with the goal of infecting as many computers or servers as possible. But this is a double edge sword, as the more spread the malware is, the easier that the attackers get blacklisted into an IP Blacklist or a DNS based blacklists (DNSBL).

9.3 Solution

The approach proposed to obtain retrained models that are able to face new malware versions is the following:

- 1) Check the Public IP of the flows that go through our network.
- 2) Build a dataset with the flows with blacklisted IPs.
- 3) Test that dataset of blacklisted flows with the actual model.
- 4) Flows that were predicted as normal traffic are included into the training dataset for a retraining of the model labeled as malware.

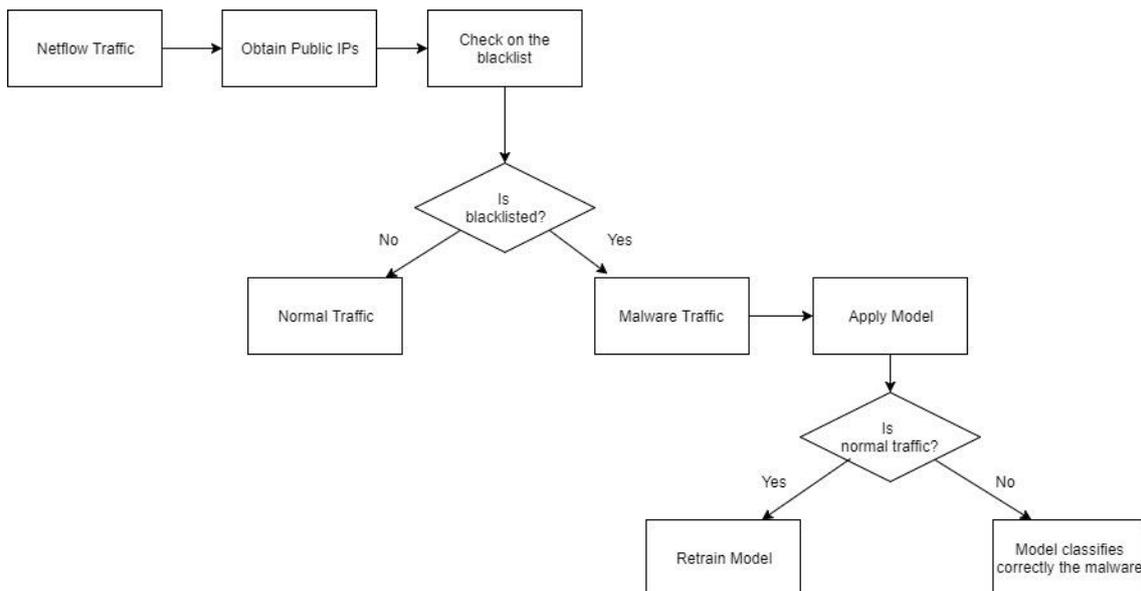


Figure 9.1: Blacklist optimization workflow.

We should point out that the solution is limited for optimizing a binary response. Blacklists don't differentiate between IPs consulted by different types of malware so we cannot retrain the model tagging the flows on the three classes specified on Section 8.4.

9.4 Implementation

After capturing the network traffic from a medium size network of about 30 computers and 5 servers during a whole hour, 92.342 flows were generated. This means that for this size 25 should be processed every second approximately. That is why it is important for traffic analysis tools to be as fast as possible; a slow method could cause a bottle neck on the packet queues that would make them to be discarded.

For this solution, we have tried do an implementation the nearest to what we could find on the real world and besides having some hardware limitations, as fast as possible in order to process the maximum number of flows in one hour.

The implementation consists on a cache and a database with blacklists. First of all, the cache in our case is implemented via software but that it could be also a small memory module implemented via hardware to any machine who wanted to use this method. Caches are used to serve information faster than if we had to search the information on a data base, disk or memory. We will take advantage of the temporal locality of the Public IP as we can see on Figure 6.3 that there are a lot of flows that make requests or connect to the same Public IP, this will allow us to save a lot of time by avoiding searching already checked IPs. We also have the database with a stored blacklist that is not stored on the machine but stored in an outside server. We are going to consult the IPs via a GET request to an API [29].

When a flow passes through the network, the first step is looking if it is located on the cache that has an access time of X less than the access time to the database, which takes X to give an answer. The cache stores the IPs that were checked on the blacklist database and the number of blacklists where the IP appears, allowing us to make a filter considering for example malicious traffic if it appears in more than X blacklists. In case of having 0 occurrences, it means that the IP has not been found on the database hence it would potentially be normal traffic.

If the IP is not found on the cache, we do a GET with the IP to the API to check if it appears on the database. The result of this operation is saved on the cache for the next time the same IP appears.

Even if searching with the API is more time consuming task in comparison of checking a database that is stored locally, we would have to update it every few time if we want it to be actualized with the last malicious IPs (IPs can change with the time) but also increase the cost of buying new storage for it.

9.5 Optimization Results

In this section we are going to expose the studies that have been done while analyzing if the solution could be viable in a real world scenario. We have done a study of how the trade-off between accuracy and time spent within the solution is affected by changing the proportion of traffic that is checked by the blacklist. For the sake of getting unbiased and realistic results, we flush the cache for every one of the different tests; this is done because we would get faster times for each new test as the IPs used would be on the cache avoiding then the call to the API.

For the model selection, we are going to use Boosted C5.0 with 20 trials as it is the model that had the better performance for binary classification and training time is not a big issue as models can be serialized so we can be retraining one model while using the old one until the retraining phase is finished.

Applying the blacklist we obtain more accuracy than applying the model to predict the flows but model predictor is much faster than the blacklist (6559 times faster), that is why we should evaluate the accuracy and the time spent of the whole solution so we can study and maximize the trade-off. Total time that could be expressed as following:

$$\begin{aligned} \textit{Total Time} = & \textit{Time of the cache} + \textit{Time of the blacklist} \\ & + \textit{Time of the model prediction} + \textit{Retraining Time} \end{aligned}$$

For the experiment, we merged traffic generated by a new unseen malware Sathurbot [30] with also new normal traffic that wasn't used for training the model, resulting into a balanced 1000 flow dataset that couldn't be bigger for limitations of time (API takes 1,1 seconds per flow to collate with the blacklist). This allows us to test the performance of our model when is tested with traffic that has a different behavior than the one that was used for training. In Figure 9.1 it can be seen the increase of the accuracy when the solution is applied for different % of Flows checked from the traffic.

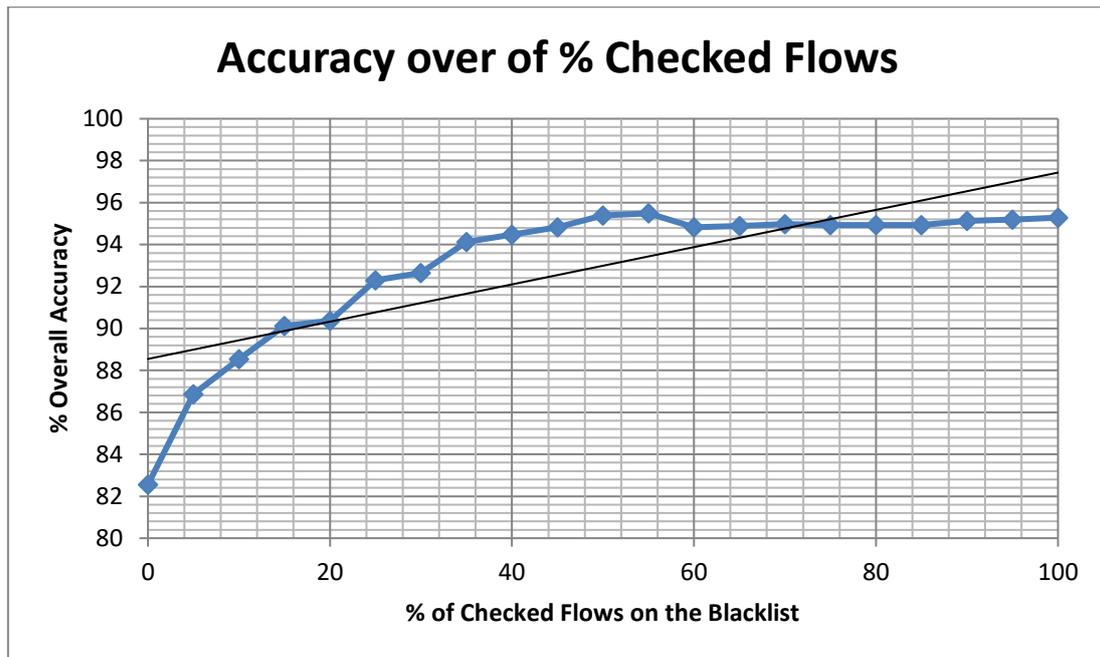


Figure 9.2: Accuracy over % Checked Flows.

As shown, testing the model without applying any optimization and checking the 0% of the new traffic through the blacklist gets a lot less accuracy than what we obtained in Section 8.3 with an 82.55% of accuracy. However, when flows start to be checked by the blacklist and the model is retrained with those flows that it incorrectly predicted, the accuracy starts increasing until getting a maximum of 95.48% accuracy when checking the 55% of the traffic through the blacklist without needing more retraining samples, much better compared with the one obtained without applying any retraining to it. Now we should compare the time spent for the whole solution. Figure 9.2 shows us the total time spent in seconds. Time has a linear cost with the number of flows checked on the blacklist, which means that the process is bottlenecked with the time spent calling the API and getting the response.

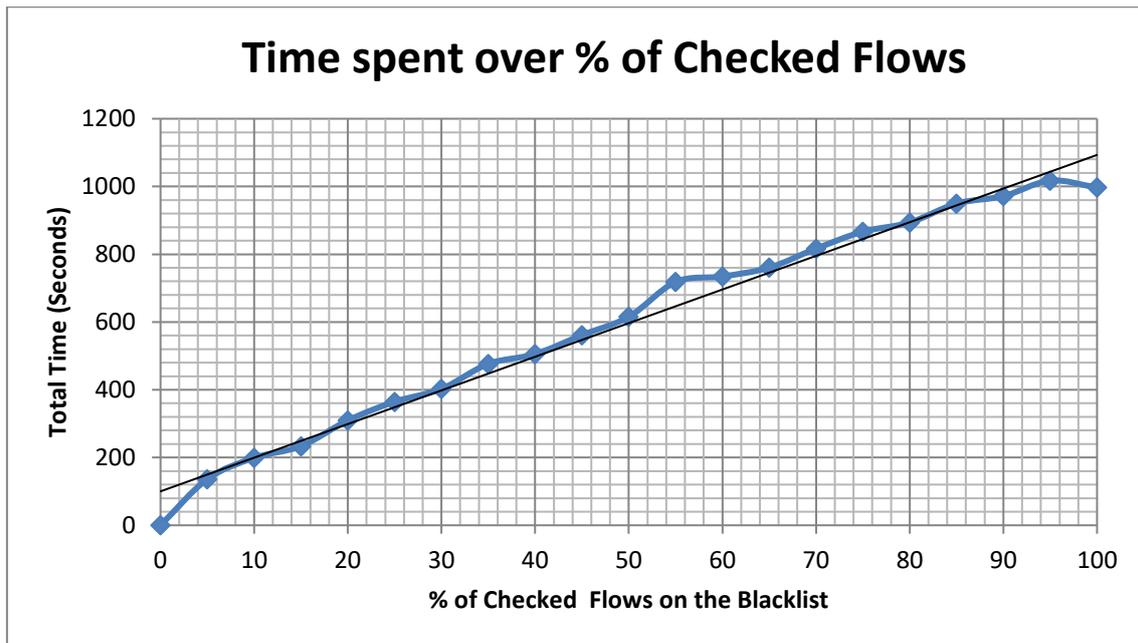


Figure 9.3: Time spent over % of Checked Traffic.

We could deduce that it doesn't make sense to check a higher percentage than 55%, as the accuracy is not increased while the cost in time goes on increasing with the percentage of checked Flows.

The solution could be adjusted into a real time traffic assuming the worst case scenario that would be that all IPs are new. In that case, the maximum percentage of flows that should be checked is:

$$\% \text{ of checked flows} = \max\left(55, \frac{\frac{\text{Mean Access API time per flow}}{3600 \text{ seconds}}}{\text{Mean of flows per hour}} \times 100\right)$$

We can observe that it is a very simple operation that can be made pretty fast. This allows us to calculate the % of checked flows dynamically and change it as a parameter of the solution. That is why we could adapt the traffic classifier to the quantity of traffic that is passing through the network at any moment without compromising or slowing the whole network.

10. Conclusions

10.1 Summary

The aim of this thesis was to study the different machine learning algorithms to classify malware traffic using only the information provided by the Netflow protocol. At first, we evaluated the predictive capacity of the models for the binary class trying to see if they could classify with more or less accuracy the traffic between normal and malware ones. After that, we tested the models for a multiclass prediction differentiating between normal, Trojan and Botnet traffic. In both cases we achieved the highest accuracy with a boosted version of the C5.0 decision tree.

Later on, optimizations were applied using additional resources to obtain a solution that could be updated and adapted to the changing environment of the network and get a great performance. After doing so, we can conclude that we achieved the goal. We obtained a high accuracy for both, binary but also multiclass classifications. In addition, the proposed solution obtained very good results improving the accuracy with respect to when we applied only the model and could also be applied to traffic classification in real time.

10.2 Future Work

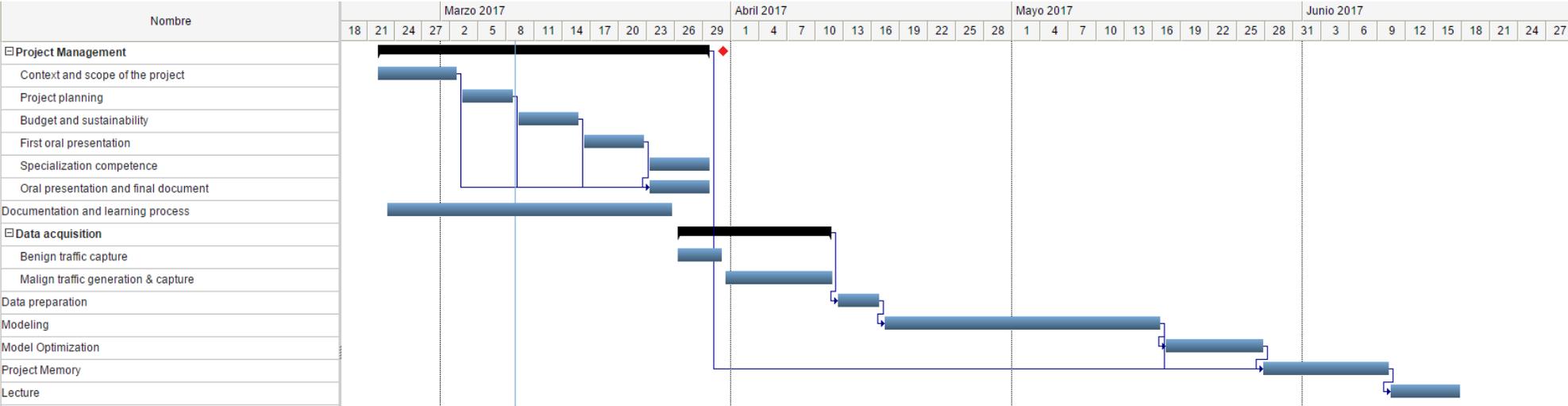
For future work, the implementation was done without using any type of parallelization that could speed the process up. We were limited to one instance that was processing all the flows being unable to handle the 100% of them. The read and processing of different flows could be parallelized on different instances in order to be able to process more flows and adapt the solution to bigger environments.

The study of different ways to decrease the time of accessing the blacklist database could give a significant speed-up on the process. For example, partially downloading a blacklist (with the X most recent IPs) and storing it into memory as second cache. The effects of having a smaller blacklist stored in local could be also analyzed.

Moreover, splitting the normal traffic into more classes defining each one for a benign application could be performed to see if the accuracy decreases after including more classes or if the model can still differentiate them along with Trojans and botnets.

Appendices

Gantt Diagram



11. References

- [1] GmbH, A.-T. (2017) AV-TEST – the independent IT-security institute.
- [2] Atique, Z. and Harkut, D. (2017). An Overview of Network Traffic Classification Methods. International Journal on Recent and Innovation Trends in Computing and Communication, 3(2), p.482.
- [3] Schneider, P. (2017). TCP/IP Traffic Classification Based on Port Numbers.
- [4] Anderson, B., Paul, S. and McGrew, D. (2016) Deciphering Malware's use of TLS (without Decryption).
- [5] Williams, N., Zander, S. and Armitage, G. (2006) 'A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification'
- [6] Free online virus, Malware and URL scanner (no date) Available at: <https://www.virustotal.com>
- [7] Mai, J., Ye, T., Zang, H. and Sridharan, A. (2006) Is Sampled Data Sufficient for Anomaly Detection? Available at: <http://conferences.sigcomm.org/imc/2006/papers/p17-mai.pdf>.
- [8] Bilge, L., Balzarotti, D., Robertson, W., Kirda, E. and Kruegel, C. (2012) DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale Netflow Analysis. Available at: <https://seclab.nu/static/publications/acsac2012disclosure.pdf>
- [9] A. Wagner and B. Plattner (2005) Entropy based worm and anomaly detection in fast IP networks - IEEE Xplore document.
- [10] Alam, S., Traore, I. and Sogukpinar, I. (2014) Current Trends and the Future of Metamorphic Malware Detection.
- [11] *Softflowd* - fast software Netflow probe Available at: <http://www.mindrot.org/projects/softflowd/>
- [12] Manpages.ubuntu.com. (2017). Ubuntu Manpage: *nfcapd* - Netflow capture daemon. [online] Available at: <http://manpages.ubuntu.com/manpages/xenial/man1/nfcapd.1.html>
- [13] Manpages.ubuntu.com. (2017). Ubuntu Manpage: *nfdump* - Netflow display and analyze program. [online] Available at: <http://manpages.ubuntu.com/manpages/xenial/man1/nfdump.1.html>
- [14] Kang, Y. and Stasko, J. (2017). Lightweight Task/Application Performance using Single versus Multiple Monitors: A Comparative Study.

11. REFERENCES

- [15] Kb.fortinet.com. (2017). Troubleshooting Tool: Using the FortiOS built-in packet sniffer. [online] Available at: <http://kb.fortinet.com/kb/viewContent.do?externalId=11186>
- [16] McAfee.com. (2017). Recent Malware - McAfee Labs Threat Center. [online] Available at: <https://www.mcafee.com/threat-intelligence/malware/latest.aspx>
- [17] Garcia, S. (2017). Stratosphere IPS. [online] Stratosphereips.org. Available at: <http://stratosphereips.org/category/dataset.html>
- [18] Contagiodump.blogspot.com.es. (2017). contagio. [online] Available at: <http://contagiodump.blogspot.com.es/>
- [19] Liu, Y. and Gopalakrishnan, V. (2017). An Overview and Evaluation of Recent Machine Learning Imputation Methods Using Cardiac Imaging Data.
- [20] Iana.org. (2017). IANA IPv4 Special-Purpose Address Registry. [online] Available at: <http://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml>
- [21] Chawla, N., Japkowicz, N. and Kotcz, A. (2004). Editorial:Special Issue on Learning from Imbalanced Data Sets. ACM SIGKDD Explorations Newsletter, 6(1), p.1.
- [22] Drummond, C. and C. Holte, R. (2017). C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling beats Over-Sampling.
- [23] Breiman, L. (2001). Random Forest. Available at: <https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf>
- [24] Schapire, R. and Freund, Y. (1996). Boosting.
- [25] Salzberg, S. (1994). C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. Machine Learning, 16(3), pp.235-240.
- [26] Random Trees R implementation, (2006). Breiman and Cutler's random forests for classification and regression.
- [27] C5.0 R implementation, (2006). C5.0 Decision Trees and Rule-Based Models. [online]
- [28] Quinlan, J. (2017). Bagging Boosting and C4.5. [online] Available at: <http://home.eng.iastate.edu/~julied/classes/ee547/Handouts/q.aaai96.pdf>.
- [29] Whoisthisip.com. (2017). IP Blacklist Check | Check if the IP is Blacklisted on DNS. [online] Available at: <https://www.whoisthisip.com/ipblock/>

11. REFERENCES

[30] Mcfp.felk.cvut.cz. (2017). Index of /publicDatasets/CTU-Malware-Capture-Botnet-303-1. [online] Available at: <https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-303-1/>

