Detection of Bitcoin miners from network measurements

Bachelor degree in Informatics, mention in Information Technologies.

Author: *Jordi Zayuelas I Muñoz*
Supervisor: *Pere Barlet Ros*
April 2019

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Abstract

In the last few years, cryptocurrency mining has become more and more important on the Internet activity and nowadays is even having a noticeable impact on the global economy. This has motivated a surge of cryptojacking, a cyberattack which consists on using compromised machines resources to mine cryptocurrencies for profit. In this context, it is of particular interest for network administrators to detect possible miners that are using network resources without permission. Currently, it is possible to detect them using lists of IP addresses from known mining pools, using information from DNS records, or directly performing Deep Packet Inspection (DPI) over all the traffic. However, these methods can either miss miners that use unknown mining servers or being too expensive to be deployed in real-world networks with considerable traffic volume. In this project I will present a method based on machine learning that detects cryptocurrency miners using Netflow/IPFIX network measurements. This enables to achieve an accuracy close to DPI-based techniques and considerably save resources, since it does not require to inspect all the packets' content.

Keywords: Netflow, mining, Bitcoin, Ethereum, Monero, cryptojacking, Machine Learning, Traffic analysis.

Acknowledgments

I would like to thank Dr. Pere Barlet-Ros, director of this thesis, and José Suárez-Varela for their help and guidance in this project. Their help and support made this research project possible.
I would also like to thank my family, friends, and my partner, Clàudia, for their support during the course of this degree.

Table of Contents

Abstract 2
Acknowledgments 3
1 Introduction and Context 4
1.1 Problem Formulation 4
1.2 Stakeholders 5
1.2.1 Target audience 5
1.2.2 Users 5
1.2.3 Beneficiaries 5
1.3 State of the art 6
2 Scope 8
3 Project planning 9
3.1 Methodology 9
3.1.1 Waterfall 9
3.1.2 Extreme Programming 9
3.2 Schedule 10
3.2.1 Considerations 10
3.3 Project planning 10
3.3.1 Initial work 10
3.3.2 First phase 10
3.3.3 Second phase 11
3.3.4 Third phase 12
3.3.5 Final work 12
3.4 Action plan 12
3.4.1 Gantt Chart 13
3.5 Resources 13
3.5.1 Human 13
3.5.2 Material 13
4 Budget 15

4.1 Hardware costs 15
4.2 Software costs 15
4.3 Human Resources 16
4.4 Indirect Costs 16
4.5 Total Costs 16
4.6 Budget Monitoring 17
5 Sustainability 18
5.1 Environmental 18
5.1.1 Project Put into production 18
5.1.2 Useful life 18
5.1.3 Risks 19
5.2 Economic 19
5.2.1 Project Put into production 19
5.2.2 Exploitation 19
5.2.3 Risks 19
5.3 Social 20
5.3.1 Project Put into production 20
5.3.2 Exploitation 20
6 Data Overview 21
6.1 Stratum 21
6.2 Netflow 24
7 Data Capture 26
7.1 Stratum 26
7.2 Data from other applications 27
8 Data analysis 29
8.1 First analysis 29
8.2 Learning Models 33
8.2.1 Support vector machines 34
8.2.2 CART 34
8.2.3 C4.5 34

8.2.4 Naïve Bayes 34
8.3 Weka Tests 34
8.3.1 Mining classification 34
8.3.2 Coin Classification 39
9 Results 42
10 Conclusions 43
10.1 Future work 43
11 Bibliography .... **Error! Bookmark not defined.**

1 Introduction and Context

1.1 Problem Formulation

Cryptocurrencies are alternative currencies that are designed to work in a digital context. One of these currencies is bitcoin, which is the first and most well-known decentralized cryptocurrency.

Most of this currencies are decentralized cryptocurrency. A decentralized cryptocurrency is any form of a currency which uses cryptography to verify transactions, and is not controlled by a single central server, but by all the nodes of the network working independently. Through this method it achieves to distribute the verification process over the whole network, since not a single person/institution controls the currency. Furthermore, it is protected against nodes failing, because when one node crash the others can keep on working unaffected.

During these last years, cryptocurrency use skyrocketed, and, with this, its price followed, which attracted lots of people wanting to make money.

In Bitcoin and other Altcoins (Bitcoin alternatives), this decentralization is achieved by miners. These miners use their computational power to validate transactions and get new bitcoins as a reward. Because of the growth of Bitcoin market price, along with the price of other cryptocurrencies, like Ethereum, lots of people decided to start mining cryptocurrencies to try and get some benefits from the growth of this technology. [1] [2] [3] [4]

This new potential profits also attracted the interest of people of dubious morality. Cybercriminals have been installing malware to infected computers, using their resources to mine bitcoin, which allows them to get bitcoin without having to use their computational resources nor having to pay the electricity bill This practice is increasing at an alarming pace, becoming a notable threat in cybersecurity. [5]

Bitcoin uses a huge amount of computational resources, which greatly increases the electricity bill and reduces the life expectancy of the electronic assets used to do it. Because of this, we may be interested in being able to find if computers in a network we're supervising are being used to mine Bitcoin, either to find possible malware infections, employees that decided to get an extra pay at the cost of company assets installing Bitcoin miners on them, or to monitor legit Bitcoin mining machinery.

To mine bitcoins, users rarely mine alone, since it would be really difficult to get rewards while racing against rest of the Bitcoin mining network. That's why most people mine in pools. Pools are groups of miners that share computational power and then split the rewards. When mining in pools, miners will send proof of their computations (called shares) to the pool, then, when splitting the reward, thanks to this, every miner will get a reward proportional to their work. [6] [7]

With the constant expanse of the traffic in networks, both public and private, with an expected growth of 370% between 2017 and 2022, it's becoming more and more difficult to analyse the data in networks. [8] [9]

Netflow is a CISCO router feature for analysing network traffic that aggregates all packets that share some values like source and destination IP and port, protocol and type of service, in a single flow, that has some relevant information about this group of packets. Even though lots of information is lost during this process, the amount of information to process is also greatly reduced. [10]

This project's objective is to analyse network traffic flows using CISCO Netflow traffic from routers that we control to find Cryptocurrency traffic that passes through the infrastructure, and use this to identify the miners and analyse the properties of the found miners, and find out how much information about the miner we can extract from Netflow.

1.2 Stakeholders

1.2.1 Target audience

The targeted audience of this project are either network analysis companies that are interested in researching mining in their network, or companies that are interested in finding if they have infected machines with malware that is mining bitcoin, or computers that are not supposed to be running bitcoin mining software but have it running anyway.

1.2.2 Users

The users of this product would be the IT workers assigned to integrate it to the network and the analysts that are going to analyse the information.

1.2.3 Beneficiaries

The beneficiaries would be the companies that would be able to find and put a stop to unintended Bitcoin mining, which would reduce the potential loss this would produce, or the network analysis companies that would benefit from the data acquired from studying the miners on the network.

1.3 State of the art

This topic is widely researched, especially by security and networking companies. CISCO even offers filters for their Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) that create alerts when Bitcoin miner activity traffic or Stratum Bitcoin mining protocol is detected.
There are multiple methods that are used to find bitcoin miners in the network:
The first one consists on reading the network traffic and try to find known IPs from mining pools used by miners.
There are some limitations to this method, miners that are not part of a pool won't be detected because they create little traffic and don't connect to this known addresses. Thankfully, solo mining is no longer profitable and it's rarely used.
Another option is to use deep packet inspection to find signs of a mining protocol being used, using signatures of these protocols and comparing them to the network traffic.
Both of these methods add lots of computational load, since all packets of the traffic must be analysed to try to find potential Bitcoin mining traffic. This is not feasible in big networks where there are too many packets to inspect, especially in deep packet inspection, where all the contents of the packets have to be analysed too. Moreover, the simple act of capturing all network traffic requires lots of resources. [11] [12] [13] [14]
Another less expensive way to find them is to log DNS requests and search for addresses that may be related to bitcoin mining. This may not be a definitive indicator that there is mining software on the network (it could be just one user visiting the webpage of that mining pool), and, with the expanding DNS over TLS it may even be evaded completely, but it should be taken into account, and it is still used nowadays. [7] [15] [16]
There are other papers about detecting mining other methods focused to detect the miners from the endpoint, using both static and dynamic code analysis and machine learning to find browser miners. [17] [18]
In terms of analysing mining properties, there are some papers published that estimate the power consumption of the Bitcoin network as a whole, but to find an individual machine hash rate (Number of operations per

second it can achieve) and consequently its expected power consumption, or which altcoin is being mined is yet to be researched. [19]

2 Scope

In this project, instead analysing all network traffic packet by packet, Netflow traffic will be used. This relieves the router and the computer from having to copy all traffic data and analysing it, improving their performance. [20] [21]

To be able to analyse network traffic with Netflow we will need:

- A Netflow exporter, preferably a router, but for testing purposes the same computer that is running the miner along with any other software that produces network traffic will suffice.
- A collector to receive the Netflow traffic and store it in the computer.
- An analyser to look at the collected traffic and extract the relevant information.

The first goal of the project will consist on creating the data, capturing Netflow traffic made by mining software and other random traffic made by a broad variety of applications. To be able to do this, since I have no access to cisco routers that can provide Netflow traffic, the following software solutions will be used: Softflowd as Exporter, nfcapd as collector, and nfdump to transform the nfcapd files to readable data that we can analyse. With the data that we will have collected, then the analysis can start, comparing the flows created by mining with the ones created by the traffic that we are not interested in (like amount of packets in the flow, size, time that the flow lasted, etc.…), and use this information to find out a way to separate flows coming from mining from those that we are not interested in.

Then, with a way to extract the flows that come from mining, I will aim to find as much information about the miner using only the data that can be extracted from its flows.

One problem that this project has is the possibility that no access to a real network is found. That would prevent the application from being tested in real life conditions, and the project would only be tested on controlled conditions where it might react differently to an uncontrolled network.

3 Project planning

3.1 Methodology

To help with the development of the project, a schedule will be followed. As mentioned earlier in the scope of the project, it will be divided in three phases. Because of this I decided to take different approaches between the phases.

During the first two, since they are pretty straightforward, the waterfall method will suffice. It is known how to do it, the only thing left to do is to design, implement and test it.

On the other hand, the third phase of the project is anything but straightforward. There is no known solution and to find a solution we will need to apply trial and error until a solution is found (Or not, because of the research nature of this part of the project, not finding a reliable solution is a possibility). That's why I decided to use extreme programming.

3.1.1 Waterfall

The waterfall methodology consists on a set of stages that we have to get through sequentially, only starting the next stage when the one before is complete. The stages stated on the original waterfall model are:

1. Requirements (Hardware and software)
2. Analysis
3. Design
4. Implementation
5. Testing
6. Operations (installation, maintenance, etc.)

The stages that we are interested in are 1 – 5. Since we are going to use this method for the first two phases, operations won't really be important, since once testing works the next phase will start. Also, stage 1 won't be needed in the second phase, since they will use the same hardware/software. [22]

3.1.2 Extreme Programming

The third phase of the project is unpredictable, so an agile method will work much better. Even though agile methods are not designed to work individually, some of it can still be applied to this project, and, because of the nature of the last part of the project, it works really well with it.

This is because it is based on short cycles and focuses on testing. This allows for changes on the original plan of the product. Since on this phase of the product we are not sure what will work or not, this allows us to try different possible solutions and modify de project depending on what works and what doesn't. [23]

3.2 Schedule

Estimated project duration

The estimated project duration is little less than 5 months, starting on the middle of September to the end of January.

3.2.1 Considerations

Even though the first two phases of the project are pretty straightforward, because of the research nature of the last phase of this project, it is bound to need some changes during its duration. Because of that, the project planning will probably need to be modified.

This is to be expected, especially with the selected extreme programming methodology that will be used while working on this phase.

3.3 Project planning

The project is divided in three main tasks, plus the initial and final work. During the duration of each task, along the work done related to the task, the corresponding part of the final report will be added to it.

3.3.1 Initial work

The initial work consists on studying the context, scope, state-of-the-art, sustainability and budget of the project, along with creating the schedule and describing the project.

This is done during the course management course.

3.3.2 First phase

This is the first of the main tasks, and it will consist on creating the first prototype of the network analyser. This will be a simple prototype that will read the traffic information and find the data related to possible Bitcoin mining traffic. This part of the project will follow a waterfall methodology, thus it would be divided in five different subtasks; requirements, analysis, design, implementation and testing. Each subtask won't start before the one before is not completed.

The first subtask, requirements, collides with the initial work, so it will already be completed during the first task. This leaves only four of the subtasks.

In Analysis the problem and technologies will be studied to have better knowledge of the task. Some part of this is also studied in the initial task, so a short amount of time will be needed to complete this task.

Designing the prototype will be the next subtask, and it will take into account the information discovered during the previous subtask to lay out what will be the base of the project.

After designing the prototype, it has to be created. This will be done on the implementation subtask, where the design arranged before will be followed to get a working prototype.

At last, to finish this first phase, the prototype has to be tested to find and fix all bugs that may be found in the code and to make sure the project works as intended.

3.3.3 Second phase

After finishing the first phase, we will be able to start the second phase.

This will consist on taking the prototype created on the first phase and with its output, find the differences with the data of the true positives and the false positives, and then use this information to finely tune the prototype to reduce the false positives to a minimum. Since this task also follows the waterfall methodology, it will follow a similar organization.

The requirements of this task will be the same of those of the first task, so we can skip this subtask and start with the analysis.

Unlike in the first phase, in the second phase analysis will be much more important. In this subtask, the output of the prototype will be analysed to find similarities between true positives and ways to discern the false positives. This will require analysing a large amount of data, therefore, it will take longer.

With these similarities and divergences, the next design subtask will start. An efficient way to implement the changes to the prototype will be created.

Then, on the next subtask, implementation, the prototype will be modified to add this changes.

To finish the prototype of the analyser and this second task, the modified prototype will need to be tested.

3.3.4 Third phase

This third phase consists on studying how to find information about the miners that are found with the analyser and how to find more information about them. This is more unpredictable and it is not clear what will be the result in the end. Because of this it will follow an extreme programming methodology.

Since this task being unpredictable, it will be given the longest amount of time, to maximize the probabilities that a working solution is found. It is also possible that after finishing the task the result just that it is not feasible to find enough information from the miner using just the flows of its traffic, and deep packet inspection should be used instead, which would be outside the scope of the project.

This task will be divided in 7 day cycles in which the possible solutions will be implemented and tested. In every cycle the task will be replanned, a new solution will be designed, and tested, then, the result will be analysed to see if it fits with the objective of this task.

3.3.5 Final work

After finishing the third phase, the final report of the project will be finished and reviewed, and a manual for using the product will be made. Then, the project will end.

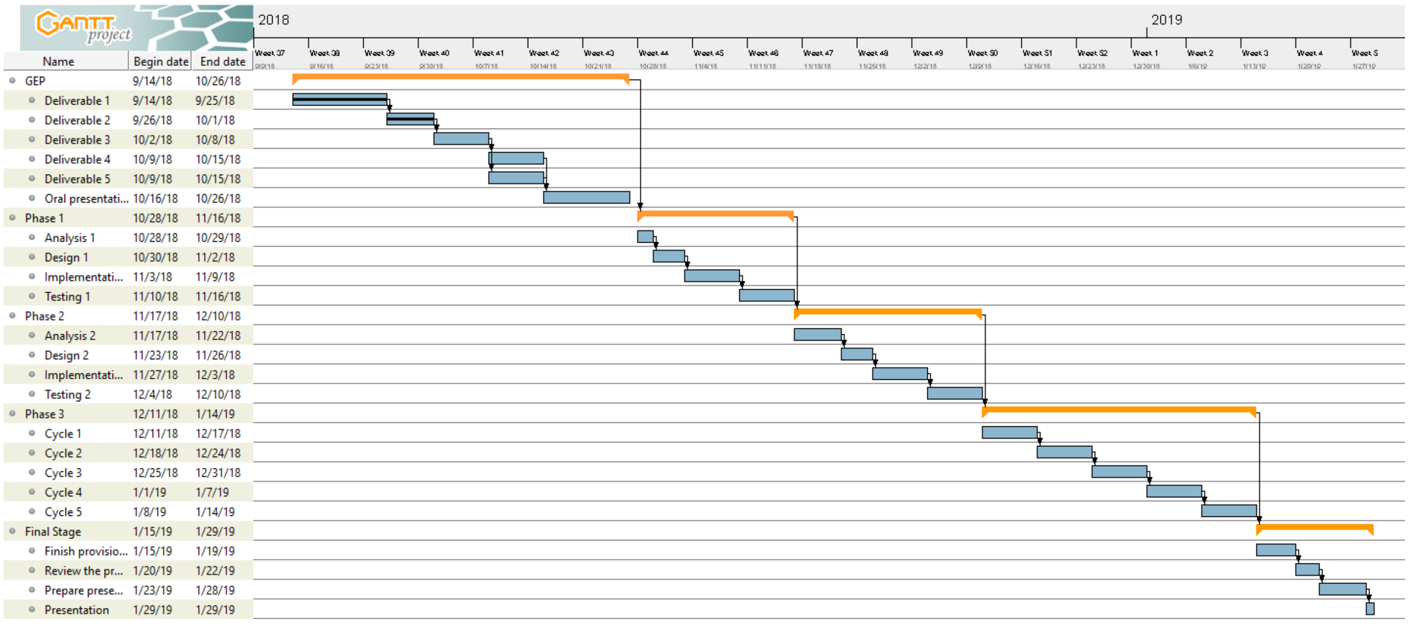
3.4 Action plan

This project will try to adhere to the following Gantt chart as close as possible, but if during the project we find that a task is not taking as long as it was thought, once this task is finished the next one will start before its beginning date, so that, in case something goes wrong there is some time left to fix it and the rest of the tasks don't have to be delayed.

At least once a month there will be a meeting with the project manager to make sure that the project is in the right path.

The estimated amount of working hours of the project will be 25 hours/week. Like this, the total amount of work is approximately 470. In case some problem arises that delays any part of the project, the working hours would be expanded to 30 hours/week until catching up with the schedule.

3.4.1 Gantt Chart



Name	Begin date	End date
GEP	9/14/18	10/26/18
Deliverable 1	9/14/18	9/25/18
Deliverable 2	9/26/18	10/1/18
Deliverable 3	10/2/18	10/8/18
Deliverable 4	10/9/18	10/15/18
Deliverable 5	10/9/18	10/15/18
Oral presentati...	10/16/18	10/26/18
Phase 1	10/28/18	11/16/18
Analysis 1	10/28/18	10/29/18
Design 1	10/30/18	11/2/18
Implementati...	11/3/18	11/9/18
Testing 1	11/10/18	11/16/18
Phase 2	11/17/18	12/10/18
Analysis 2	11/17/18	11/22/18
Design 2	11/23/18	11/26/18
Implementati...	11/27/18	12/3/18
Testing 2	12/4/18	12/10/18
Phase 3	12/11/18	1/14/19
Cycle 1	12/11/18	12/17/18
Cycle 2	12/18/18	12/24/18
Cycle 3	12/25/18	12/31/18
Cycle 4	1/1/19	1/7/19
Cycle 5	1/8/19	1/14/19
Final Stage	1/15/19	1/29/19
Finish provisio...	1/15/19	1/19/19
Review the pr...	1/20/19	1/22/19
Prepare prese...	1/23/19	1/28/19
Presentation	1/29/19	1/29/19

Figure 1 - Gantt chart. Created using GanttProject.

3.5 Resources

3.5.1 Human

Approximate number of man hours to be dedicated to each task:

Initial work	150
Phase one	70
Phase two	80
Phase three	125
Final work	50

Table 1 - Work hours for each task

3.5.2 Material

For this project I will need the following resources:

Hardware:

- Custom desktop computer
- NVidia GeForce 670
- Laptop MSI MS-16GD

Software:

- Debian Linux
- Cgminer
- Softflowd
- Flowtools

- Windows 10
- Microsoft Office word 2010
- Microsoft Office excel 2010
- GanttProject

4 Budget

As mentioned on the previous deliverable, this project will require not only man hours, but also hardware and licensed software. In top of that indirect costs should be added. All this doesn't come for free, thus, a budget should be created to be able to manage all these costs and estimate how much this project will cost.

4.1 Hardware costs

This project requires two computers, one of them equipped with a graphic card to mine bitcoin. A computer has a life expectancy of 4 years. With this we can estimate the Amortization of each product.

The Desktop and the graphic card will be used during all three phases of the project, where the software will be created and tested. On the other hand, the MSI laptop will be used during all the tasks to document the results.

Hardware	Cost (€)	Useful life (Years)	Amortization (€)
Desktop Computer	600	4	20.54
NVidia GeForce 670	200	4	6.85
MSI MS-16GD	700	4	22.66
Total:	1500	-	50.05

Table 2 - Hardware costs

4.2 Software costs

While most of the software used is open source, Microsoft Word and Windows 10 will need a paid license. The useful life of the Operative system windows 10 is expected to last until October 2025, when it will stop to receive updates. The useful life of Microsoft Word is around 3 years.

This costs are mostly linked to the first and last tasks, where work will be highly focused on working on the project report, but will be used in the rest of the tasks when the results are written on the report.

Software	Cost (€)	Useful life (Years)	Amortization (€)
Microsoft Word	79	3	3.41

Windows 10	145	6	3.13
Total:	1500	-	6.54

Table 3 - Software costs

4.3 Human Resources

This project is developed entirely by a single person, so all work hours fall on a single worker. The price/hour is relative to the technical difficulty of each task.

Task	Price/Hour (€)	Work Hours	Cost (€)
Initial work	25	150	3750
Phase one	30	70	2100
Phase two	30	80	2400
Phase three	35	125	4375
Final work	25	50	1250
Total:	-	475	13875

Table 4 - Human resources

4.4 Indirect Costs

The indirect costs are those that are not specific to this project, but have to be paid anyway. In the next table lay those expenses:

Concept	Price month (€)	Months	Cost (€)
Optic fibre	70	5	350
Electricity	45	5	225
Total:	-	-	575

Table 5 - Indirect costs

4.5 Total Costs

This table shows the expected total cost of the project, 10% of the original expected cost of the project has been added to cover any possible unexpected expense that might arise.

	Cost (€)
Hardware	50.05
Software	6.54
Human resources	13875

Indirect Costs	575
Unexpected expenses	1450.65
Total:	15957.25

Table 6 - Total costs

4.6 Budget Monitoring

To make sure that the project stays close to the plan, the budget will be reviewed and updated every time one of the phases of the project is finished, adding any costs created by unexpected events. This will allow keeping an eye to the budget and to find possible deviations and correct them.

A small amount of deviations is expected, since the planning of the project is prone to changes, what would affect the budget in terms of man hours. In terms of hardware and software costs, there shouldn't be any deviance.

5 Sustainability

In the following chapter, the sustainability of this solution will be analysed, in terms of its economic, environmental, and social sustainability.

5.1 Environmental

5.1.1 Project Put into production

During the production of the project, a large use of computational resources was used to mine cryptocurrencies. This is bad because cryptocurrency mining uses all the power of the computer, which reduces the useful life of the hardware and wastes lots of electricity. Reducing the useful life of the hardware means having to renew the equipment more often; this creates lots of waste in the long run.

To minimize this environmental impact, the computer only mined when it was necessary to create traffic, and it was not left mining just to try to get some of this currency. In addition, the computer used will actually be the same computer that I usually use on my everyday life. Furthermore, the miner was throttled down when mining to reduce the carbon footprint.

5.1.2 Useful life

The results of this research will help to find unwanted miners in networks. These miners consume lots of energy and reduce the life expectancy of the hardware being used. As explained before, this is bad for de environment, and finding them will help to stop them, and with that stop this waste of resources.

The current solutions to this problem use deep packet inspection, which is really computationally expensive, and, using it, while is better than leave the miner alone, still uses lots of resources, which in turn uses a lot of energy, which increases the footprint of the product. With the solution proposed on this project, the miners would be found through the Netflow traffic instead, which is way smaller than the traffic it represents and easier to analyse. Because of this, while creating a minimal overhead to the machines running the program, allows us to find the miners and to stop them.

It is difficult to estimate the amount of resources that this project would save during the course of its useful life, because for this we would need an estimate of what amount of cryptocurrency mining comes from unwanted miners, which is, in terms of current research, unknown.

5.1.3 Risks

Beforehand is difficult to know how much mining data will be needed. There exists the possibility that the needed amount of data is more than expected, in this case, the amount of time needed to mine will have to be extended, thus increasing the amount of time that the computer will be mining and wasting resources and increasing the carbon footprint of the project.

5.2 Economic

5.2.1 Project Put into production

The project expected budget was approximately 16000 €. This is not so much for a project in information technology, but depending on the context where it's going to be applied, the project may be too expensive for the its use.
During the project, it was fond that the changes of traffic between a miner using its full power and minimal power are insignificant. This helped reduce the amount of electricity and resources wasted on mining by throttling down the speed of the miners.

5.2.2 Exploitation

If it is to be used on a single small network, it may not be a good idea, and using any of the existent solutions might be a better alternative, since the winnings on energy and hardware would be minimal in comparison. On the other hand, in case that the system is to be sold and used in lots of networks, or to be installed on a big network, this solution might be a better alternative than the existing others, since this solution uses less electricity, and, on the long run the saved energy will be worth more than the budget of the project.
Deep packet inspection is a method that is not deployed in most companies. On the other hand, Netflow is commonly already implemented in many networks nowadays. This avoids the need to create a new extremely expensive security solution able to inspect the contents of all the transmitted traffic, adapting an already existing solution instead, reducing the costs of adding a solution to find miners in the network.

5.2.3 Risks

When hardware is used for mining, its useful life is reduced. If the mining software is not configured properly, there is the risk that the hardware burns out and stops working. This would mean that new hardware would be needed, adding to the cost of the project.

5.3 Social

5.3.1 Project Put into production

While doing this project I learned about multiple protocols and concepts about networking, like Netflow, how mining and pools work.
I also learned to evaluate the sustainability of a project and all its related concepts, like social, environmental and Economic sustainability.
Lastly, I acquired experience in the academic world and research, which has been an invaluable experience.

5.3.2 Exploitation

The problem addressed in this project is already resolved in other less efficient ways, so, in terms of users, they wouldn't notice. The only ones that would notice are the ones paying the bills, so, socially speaking, it wouldn't make any changes.
On the other hand, the third part of the project, investigating the found miners would make the life of analysts much easier, since with it you would be able to find the power used and what kind of machine is mining bitcoin.
The only collective that will be affected in a wrong way by this project are the operators of illicit mining operations that might be found, which will have to stop their operations and may get charged for them.

6 Data Overview

6.1 Stratum

The data we are interested in is the one coming from stratum protocol. This is a protocol used by most pools to communicate between the miner and the pool server, and consists of a set of instructions that the server can send to the miner, and another set of requests that the miner can make to the server.
This protocol is implemented on top of TCP, and there is no port that is commonly linked to it, thus making the destination port useless when trying to detect stratum connections.
This protocol uses JSON for all its methods.
From client to server, the most used calls are mining.subscribe, mining.authorize, mining.extranonce.subscribe and mining.submit.
From server to client are mining.set_difficulty and mining.notify
Client – Server:

- **mining.subscribe:** The client asks for jobs to the server so it can start mining.
- **mining.authorize:** Authentication from a miner in the connection towards the server. (There can be more than one miner in a single stratum connection).
- **mining.extranonce.subscribe:** Asks to the server for another nonce to be used when mining.
- **mining.submit:** The client, after finding an answer for the block with the difficulty set by the server, sends the information about the found nonce to the server, thus getting a share of the block.

Server – Client:

- **mining.notify:** The server sends all the information needed to start mining the current block, including the custom id, used as identifier when submitting shares, the hash of the previous block, all the data about the transactions and the merkle branch.
- **mining.set_difficulty:** Sets the difficulty the miners should use for mining. This can vary, and depending on the difficulty a share was mined it will be worth more or less.

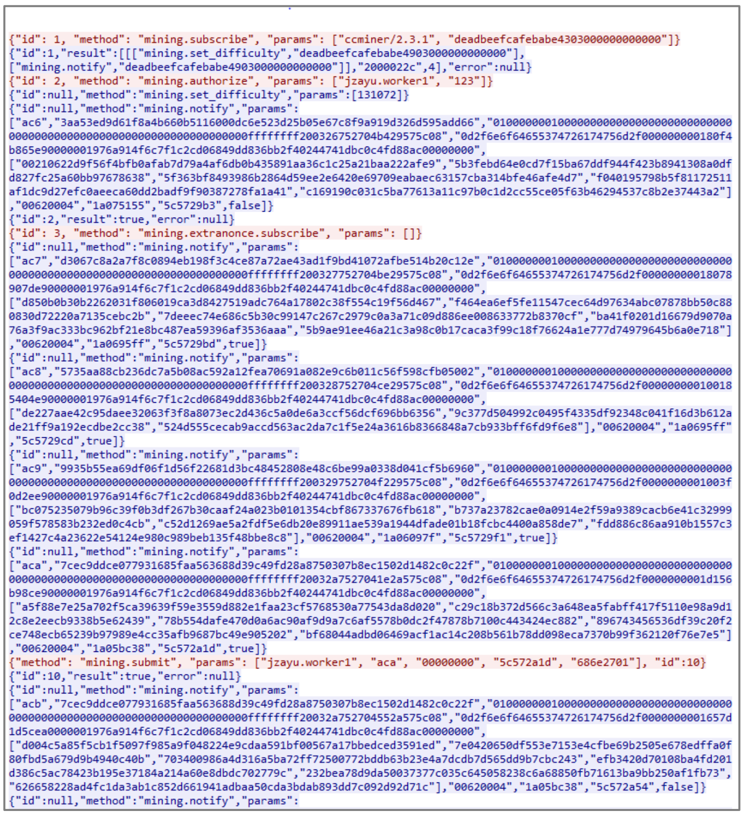
[14] [24]

After a short capture made while mining bitcoin I was able to analyse the connection between the miner and the server.

As it can be seen in Figure 2 and Figure 3, the server transmits most of the data, while most of the packets of the client are Keep-Alive and ACK packets that transmit no information, and when they transmit data it's still a small amount, since its methods transmit little information and are only used at the start of the connection or rare occasion that a share is found or a new nonce is needed.

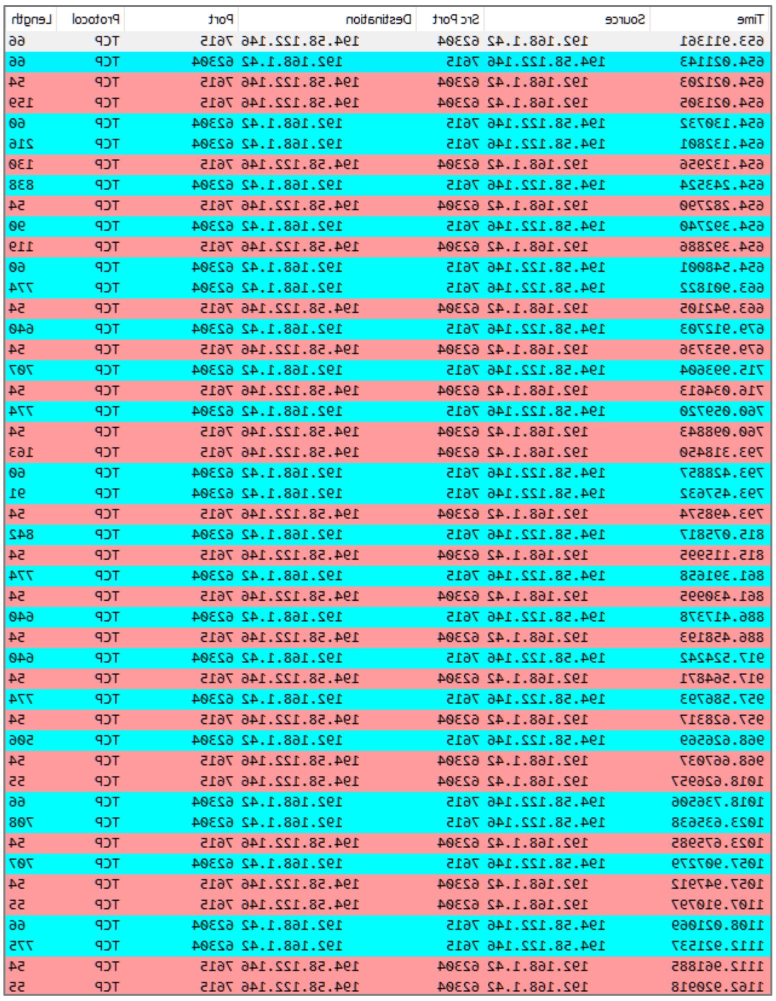
On the other hand, the server transmits data on a steady pace, since it keeps transmitting all the data needed to mine the block and refreshing it to add the new transactions to it through the mining.submit method.

All this information obtained now will be useful later when dealing with the Netflow traffic.



```
{"id": 1, "method": "mining.subscribe", "params": ["ccminer/2.3.1", "deadbeefcafebabe4303000000000000"]}
{"id":1,"result":[[["mining.set_difficulty","deadbeefcafebabe4903000000000000"],
["mining.notify","deadbeefcafebabe4903000000000000"]],"2000022c",4],"error":null}
{"id": 2, "method": "mining.authorize", "params": ["jzayu.worker1", "123"]}
{"id":null,"method":"mining.set_difficulty","params":[131072]}
{"id":null,"method":"mining.notify","params":
["ac6","3aa53ed9d61f8a4b660b5116000dc6e523d25b05e67c8f9a919d326d595add66","0100000001000000000000000000000000000
0000000000000000000000000000000000000ffffffff200326752704b429575c08","0d2f6e6f64655374726174756d2f000000000180f4
b865e90000001976a914f6c7f1c2cd06849dd836bb2f40244741dbc0c4fd88ac00000000",
["00210622d9f56f4bfb0afab7d79a4af6db0b435891aa36c1c25a21baa222afe9","5b3febd64e0cd7f15ba67ddf944f423b8941308a0df
d827fc25a60bb97678638","5f363bf8493986b2864d59ee2e6420e69709eabaec63157cba314bfe46afe4d7","f040195798b5f81172511
af1dc9d27efc0aeeca60dd2badf9f90387278fa1a41","c169190c031c5ba77613a11c97b0c1d2cc55ce05f63b46294537c8b2e37443a2"]
,"00620004","1a075155","5c5729b3",false]}
{"id":2,"result":true,"error":null}
{"id": 3, "method": "mining.extranonce.subscribe", "params": []}
{"id":null,"method":"mining.notify","params":
["ac7","d3067c8a2a7f8c0894eb198f3c4ce87a72ae43ad1f9bd41072afbe514b20c12e","0100000001000000000000000000000000000
0000000000000000000000000000000000000ffffffff200327752704be29575c08","0d2f6e6f64655374726174756d2f00000000018078
907de90000001976a914f6c7f1c2cd06849dd836bb2f40244741dbc0c4fd88ac00000000",
["d850b0b30b2262031f806019ca3d8427519adc764a17802c38f554c19f56d467","f464ea6ef5fe11547cec64d97634abc07878bb50c88
0830d72220a7135cebc2b","7deeec74e686c5b30c99147c267c2979c0a3a71c09d886ee008633772b8370cf","ba41f0201d16679d9070a
76a3f9ac333bc962bf21e8bc487ea59396af3536aaa","5b9ae91ee46a21c3a98c0b17caca3f99c18f76624a1e777d74979645b6a0e718"]
,"00620004","1a0695ff","5c5729bd",true]}
{"id":null,"method":"mining.notify","params":
["ac8","5735aa88cb236dc7a5b08ac592a12fea70691a082e9c6b011c56f598cfb05002","0100000001000000000000000000000000000
0000000000000000000000000000000000000ffffffff200328752704ce29575c08","0d2f6e6f64655374726174756d2f00000000010018
5404e90000001976a914f6c7f1c2cd06849dd836bb2f40244741dbc0c4fd88ac00000000",
["de227aae42c95daee32063f3f8a8073ec2d436c5a0de6a3ccf56dcf696bb6356","9c377d504992c0495f4335df92348c041f16d3b612a
de21ff9a192ecdbe2cc38","524d555cecab9accd563ac2da7c1f5e24a3616b8366848a7cb933bff6fd9f6e8"],"00620004","1a0695ff"
,"5c5729cd",true]}
{"id":null,"method":"mining.notify","params":
["ac9","9935b55ea69df06f1d56f22681d3bc48452808e48c6be99a0338d041cf5b6960","0100000001000000000000000000000000000
0000000000000000000000000000000000000ffffffff200329752704f229575c08","0d2f6e6f64655374726174756d2f0000000001003f
0d2ee90000001976a914f6c7f1c2cd06849dd836bb2f40244741dbc0c4fd88ac00000000",
["bc075235079b96c39f0b3df267b30caaf24a023b0101354cbf867337676fb618","b737a23782cae0a0914e2f59a9389cacb6e41c32999
059f578583b232ed0c4cb","c52d1269ae5a2fdf5e6db20e89911ae539a1944dfade01b18fcbc4400a858de7","fdd886c86aa910b1557c3
ef1427c4a23622e54124e980c989beb135f48bbe8c8"],"00620004","1a06097f","5c5729f1",true]}
{"id":null,"method":"mining.notify","params":
["aca","7cec9ddce077931685faa563688d39c49fd28a8750307b8ec1502d1482c0c22f","0100000001000000000000000000000000000
0000000000000000000000000000000000000ffffffff20032a7527041e2a575c08","0d2f6e6f64655374726174756d2f0000000001d156
b98ce90000001976a914f6c7f1c2cd06849dd836bb2f40244741dbc0c4fd88ac00000000",
["a5f88e7e25a702f5ca39639f59e3559d882e1faa23cf5768530a77543da8d020","c29c18b372d566c3a648ea5fabff417f5110e98a9d1
2c8e2eecb9338b5e62439","78b554dafe470d0a6ac90af9d9a7c6af5578b0dc2f47878b7100c443424ec882","89674345653 6df39c20f2
ce748ecb65239b97989e4cc35afb9687bc49e905202","bf68044adbd06469acf1ac14c208b561b78dd098eca7370b99f362120f76e7e5"]
,"00620004","1a05bc38","5c572a1d",true]}
{"method": "mining.submit", "params": ["jzayu.worker1", "aca", "00000000", "5c572a1d", "686e2701"], "id":10}
{"id":10,"result":true,"error":null}
{"id":null,"method":"mining.notify","params":
["acb","7cec9ddce077931685faa563688d39c49fd28a8750307b8ec1502d1482c0c22f","0100000001000000000000000000000000000
0000000000000000000000000000000000000ffffffff20032a752704552a575c08","0d2f6e6f64655374726174756d2f0000000001657d
1d5cea0000001976a914f6c7f1c2cd06849dd836bb2f40244741dbc0c4fd88ac00000000",
["d004c5a85f5cb1f5097f985a9f048224e9cdaa591bf00567a17bbedced3591ed","7e0420650df553e7153e4cfbe69b2505e678edffa0f
80fbd5a679d9b4940c40b","703400986a4d316a5ba72ff72500772bddb63b23e4a7dcdb7d565dd9b7cbc243","efb3420d70108ba4fd201
d386c5ac78423b195e37184a214a60e8dbdc702779c","232bea78d9da50037377c035c645058238c6a68850fb71613ba9bb250af1fb73",
"626658228ad4fc1da3ab1c852d661941adbaa50cda3bdab893dd7c092d92d71c"],"00620004","1a05bc38","5c572a54",false]}
{"id":null,"method":"mining.notify","params":
```

Figure 2– Stratum protocol data between the server (Blue) and the client (Red), Created using Wireshark.



Time	Source	Src Port	Destination	Port	Protocol	Length
653.911361	192.168.1.42	62304	194.58.122.146	7615	TCP	66
654.021143	194.58.122.146	7615	192.168.1.42	62304	TCP	66
654.021203	192.168.1.42	62304	194.58.122.146	7615	TCP	54
654.021305	192.168.1.42	62304	194.58.122.146	7615	TCP	159
654.130732	194.58.122.146	7615	192.168.1.42	62304	TCP	60
654.132801	194.58.122.146	7615	192.168.1.42	62304	TCP	216
654.132956	192.168.1.42	62304	194.58.122.146	7615	TCP	130
654.243524	194.58.122.146	7615	192.168.1.42	62304	TCP	838
654.282790	192.168.1.42	62304	194.58.122.146	7615	TCP	54
654.392740	194.58.122.146	7615	192.168.1.42	62304	TCP	90
654.392886	192.168.1.42	62304	194.58.122.146	7615	TCP	119
654.548001	194.58.122.146	7615	192.168.1.42	62304	TCP	60
663.901822	194.58.122.146	7615	192.168.1.42	62304	TCP	774
663.942105	192.168.1.42	62304	194.58.122.146	7615	TCP	54
679.912703	194.58.122.146	7615	192.168.1.42	62304	TCP	640
679.953736	192.168.1.42	62304	194.58.122.146	7615	TCP	54
715.993604	194.58.122.146	7615	192.168.1.42	62304	TCP	707
716.034613	192.168.1.42	62304	194.58.122.146	7615	TCP	54
760.059720	194.58.122.146	7615	192.168.1.42	62304	TCP	774
760.098843	192.168.1.42	62304	194.58.122.146	7615	TCP	54
793.318450	192.168.1.42	62304	194.58.122.146	7615	TCP	163
793.428857	194.58.122.146	7615	192.168.1.42	62304	TCP	60
793.457632	194.58.122.146	7615	192.168.1.42	62304	TCP	91
793.498574	192.168.1.42	62304	194.58.122.146	7615	TCP	54
815.075817	194.58.122.146	7615	192.168.1.42	62304	TCP	842
815.115995	192.168.1.42	62304	194.58.122.146	7615	TCP	54
861.391658	194.58.122.146	7615	192.168.1.42	62304	TCP	774
861.430995	192.168.1.42	62304	194.58.122.146	7615	TCP	54
886.417378	194.58.122.146	7615	192.168.1.42	62304	TCP	640
886.458193	192.168.1.42	62304	194.58.122.146	7615	TCP	54
917.524242	194.58.122.146	7615	192.168.1.42	62304	TCP	640
917.564871	192.168.1.42	62304	194.58.122.146	7615	TCP	54
957.586793	194.58.122.146	7615	192.168.1.42	62304	TCP	774
957.628317	192.168.1.42	62304	194.58.122.146	7615	TCP	54
968.626569	194.58.122.146	7615	192.168.1.42	62304	TCP	506
968.667037	192.168.1.42	62304	194.58.122.146	7615	TCP	54
1018.626957	192.168.1.42	62304	194.58.122.146	7615	TCP	55
1018.736506	194.58.122.146	7615	192.168.1.42	62304	TCP	66
1023.635638	194.58.122.146	7615	192.168.1.42	62304	TCP	708
1023.675985	192.168.1.42	62304	194.58.122.146	7615	TCP	54
1057.907279	194.58.122.146	7615	192.168.1.42	62304	TCP	707
1057.947912	192.168.1.42	62304	194.58.122.146	7615	TCP	54
1107.910797	192.168.1.42	62304	194.58.122.146	7615	TCP	55
1108.021069	194.58.122.146	7615	192.168.1.42	62304	TCP	66
1112.921537	194.58.122.146	7615	192.168.1.42	62304	TCP	775
1112.961885	192.168.1.42	62304	194.58.122.146	7615	TCP	54
1162.920918	192.168.1.42	62304	194.58.122.146	7615	TCP	55

Figure 3 – TCP connection packets between Client (Red) and server (Blue) , Created using Wireshark.

There exist some other protocols used by some of the mining pools. Every other method found during the investigation also used Json messages, and very similar communication patters. The methods allowed by these protocols were sight variations of the original stratum protocol to allow different authentication methods and other small variations.

6.2 Netflow

Netflow is a CISCO router feature for analysing network traffic that aggregates all packets that share some values like source and destination IP and port, protocol and type of service, in a single flow, which stores general information about the connection.
It must be taken into account that a flow goes in a single direction. The client-server and the server-client packets of the same connection will form two flows.
Netflow v5, the version that will be used for this project has the following fields [25]:

- Source IP
- Destination IP
- Source Port
- Destination Port
- Next Hop
- Input SNMP interface
- Output SNMP interface
- Number of packets in the flow
- Number of bytes in the flow (layer 3 communications)
- Timestamp of the start of the flow
- Timestamp of the last ending of the flow
- TCP Flags
- Protocol
- Type of Service
- Source and destination autonomous system
- Source address mask
- Destination address mask

From all this information we have to extract all the data that could be relevant in order to identify stratum traffic.
Since we don't know every mining pool that exists, the IP address won't help for identifying stratum flows, since new pools may appear or a miner might use a private pool that we don't know about.
The ports won't be of use either, since the stratum protocol doesn't use any port by default.

Next hop, input and output SNMP interfaces, Type of service, autonomous system and the addresses mask won’t help either, since they mostly contain information about where the flows are being captured.
This leaves us with start and end time, number of packets, number of bytes, protocols and flags.
For starters, all traffic that doesn’t use TCP can be discarded.
As we’ve seen on the capture of stratum packets, the amounts of information that get transmitted is completely different depending on the direction, since the flows are not bidirectional, if we find the complementary flows (client-server and server-client) we will have much more information about the connection, increasing the chances of correctly classifying the flows as stratum or not.
The data we have left, as it is now, is not easy to analyse to find common characteristics that can help to classify them as stratum.
Since the number of bytes and the number of packets highly depend on the duration of the flow, flows coming from the same application that had a different duration would have completely different values. It can be solved creating values that do not depend on the duration of the flow. That’s why I decided to add the following fields to the data:

- Packets/second Inbound
- Bits/second Inbound
- Bits/Packet Inbound
- Packets/second Outbound
- Bits/second Outbound
- Bits/Packet Outbound
- Packets Inbound/Packets Outbound
- Bits Inbound/Bits Outbound

This new fields will make it easier to look at the data since it doesn’t depend on arbitrary conditions that depend on how long the flow lasted, and show data that actually can be compared between flows that lasted a different amount of time.

7 Data Capture

7.1 Stratum

After deciding which information about Netflow we will use, It’s time to create the stratum Netflow traffic.
There are many altcoins, each one has different characteristics, like the block time, how many transactions are recorded each block, the average block size, etc. That may introduce small deviations in the results, so this should be taken into account.
That’s why I decided to collect data from many different altcoins to get a broader look at the characteristics of the stratum protocol as a whole.
I recorded data from the following coins: Bitcoin, Bitcoin-CASH, DogeCoin, LiteCoin, Monero and Ethereum.
The pools used to mine Monero and Ethereum did not use the stratum protocol, but similar custom protocols that also worked over JSON. Since these protocols are designed for the same purpose as stratum, they share the same basic properties, and created similar traffic.
Monero is a cryptocurrency that is commonly mined by malware when it is installed on a machine, so it would be especially useful to detect that kind of traffic, that’s why I decided to add this altcoin traffic to the mix.
In the following table contains some information about each cryptocurrency that may be relevant: [26]

Coin	Average Block time	Average Block size	Average number of Transactions per block.	Hashing algorithm
Bitcoin	10 min	787kb	2270	SHA-256
Bitcoin-Cash	10 min	50kb	115	SHA-256
DogeCoin	1 min	11kb	20	Scrypt
LiteCoin	2.5 min	20kb	42	Scrypt
Monero	2 min	16kb	6	Cryptonight
Ethereum	13 sec	20kb	96	Keccak256

Table 7 – Cryptocurrencies information [26].

To collect the traffic, I captured pcap files during various hours while mining with various cryptocurrencies. Then, I used Softflowd as exporter and nfcapd as collector to create nfcapd files. Then I extracted the data using nfdump.

After starting capturing the flows, I realized that they are extremely long, being between 10 and 30 minutes each. Therefore, they produce a very little number of flows in comparison with other applications. This will reduce the amount of stratum data that I will be able to create, and may have an impact later when trying to find stratum flows that lay hidden among millions of other flows.
For each coin, I captured data for 24-72 Hours, achieving the achieving the following results:

Coin	**Flows Captured**
Bitcoin	303
Bitcoin-Cash	77
LiteCoin	77
DogeCoin	85
Monero	58
Ethereum	91
Total:	691

Table 8 - Mining data collected.

Even though ideally I would have aimed for a larger amount of data, since I only have one computer that can be used for mining, and of time limitations, this is all the data I have been able to collect.

7.2 Data from other applications

To test whether we can differentiate between stratum and other flows or not, and to find if there is some information in flows created by stratum connections that allows us to discern them from other flows, flows from applications that use other protocols will be needed.

For an initial test I decided to record flows from various applications commonly used. This includes http and https traffic from multiple different websites (audio streaming, video, static sites, forums, etc.), data from p2p applications (torrents with different max speeds and online games), and other traffic from desktop applications that connect to their servers.

The tests were made with a windows computer, so, since windows is constantly connecting to Microsoft servers, those connections also created flows.

After one hour of data capture I managed to collect approximately 1.000.000 flows, way more than what I have been able to capture from mining traffic. This shows that stratum barely creates any amount of flows in comparison with other applications.

This might be a problem with identifying stratum traffic later on, since a small number of false positives could be enough to hide the true positives.

8 Data analysis

8.1 First analysis

After creating all the data, I decided to create some graphs to see if there is an obvious difference between flows created by mining software and flows created by other applications.

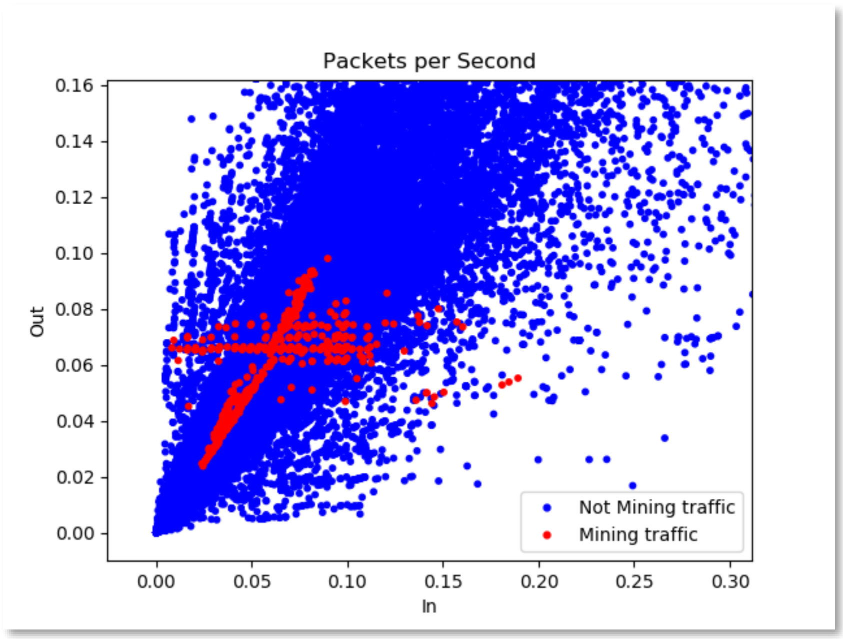


Figure 4 - This plot shows the amount of packets per second sent and received each second. Created with matplotlib. [31]

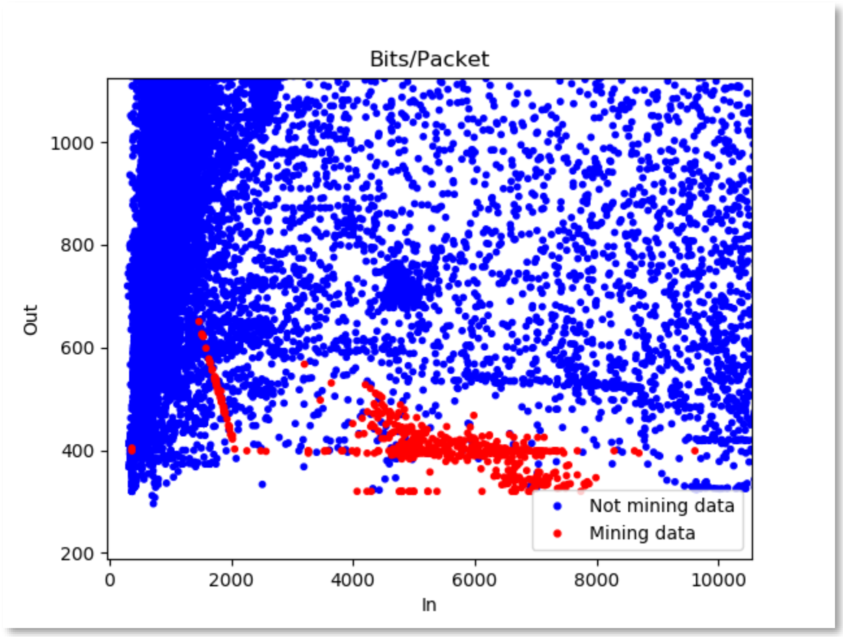


Figure 5 - This plot shows the average number of bits each packets holds in each flow, sent and received. Created with matplotlib. [31]

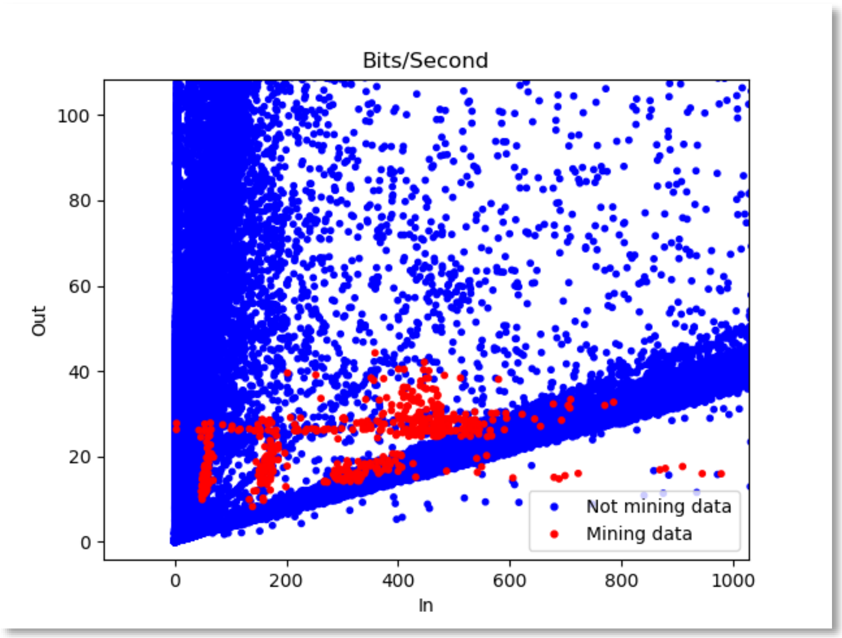


Figure 6 - This plot shows the amount of packets per second sent and received each second. Created with matplotlib. [31]

It should be noted that the choice of data fields used to create the plots is arbitrary, with the sole purpose of showing the differences between the flows created mining traffic and the rest of the flows

As It can be seen in the plots in Figures 4, 5 and 6, there is a vast difference between most flows, which tend to have bigger bitrates and send more packets per second, and stratum, which concentrates its results near the lower parts of the graph.

We were able to find some interesting information from these plots, but they still don't have enough information to extract an easy rule that will allow us to determine if a flow comes from a mining operation or not. The fact that we can't find a straightforward answer doesn't mean that it's not possible to find a way to detect stratum traffic from this data. Since there appears to be some differences between mining and other kinds of traffic and there are more fields that haven't been plotted, there could be a hidden way to extract the mining flows using an unseen relation between the data fields. That is why I decided to run some tests with machine learning to see if we can find a model that allows us to find this stratum traffic.

As a side note, another thing I noticed on these plots is that there are different patterns in the mining data, which may be created by the different coins, this is why I created another group of graphs focused on mining data only, to see if there is a relation to each coin and the data they create.

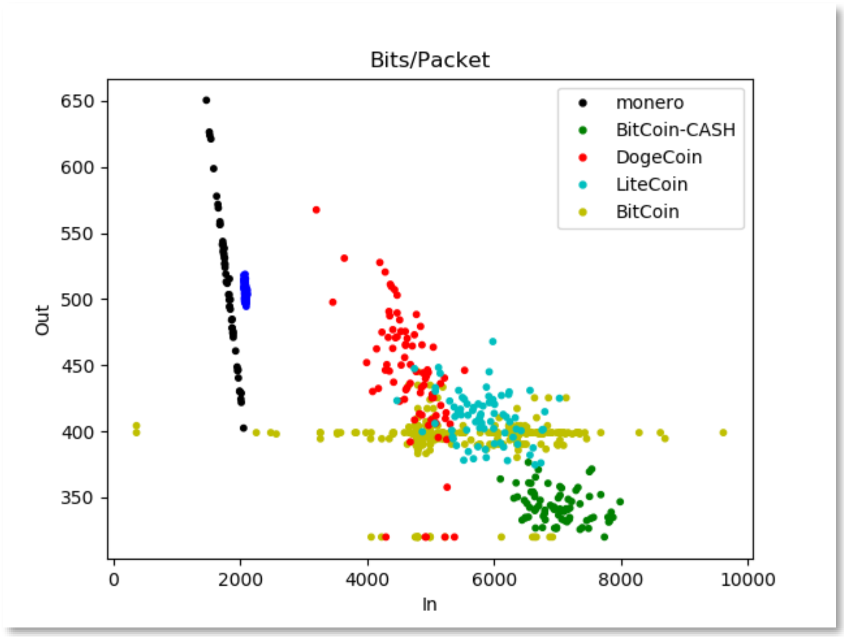


Figure 7 - This plot shows the amount of packets per second sent and received each second for each coin. Created with matplotlib. [31]

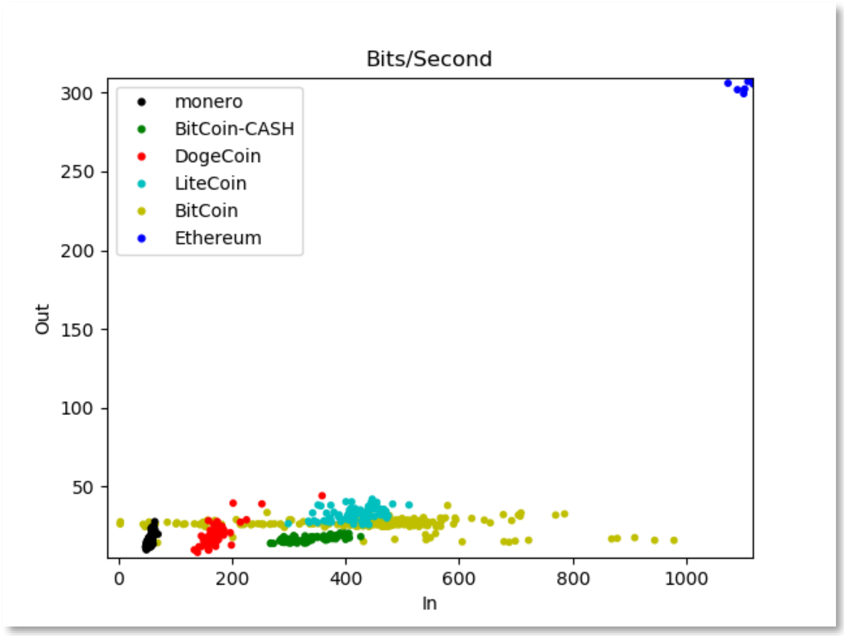


Figure 8 - This plot shows the average number of bits each packets holds in each flow, sent and received for each coin. Created with matplotlib. [31]

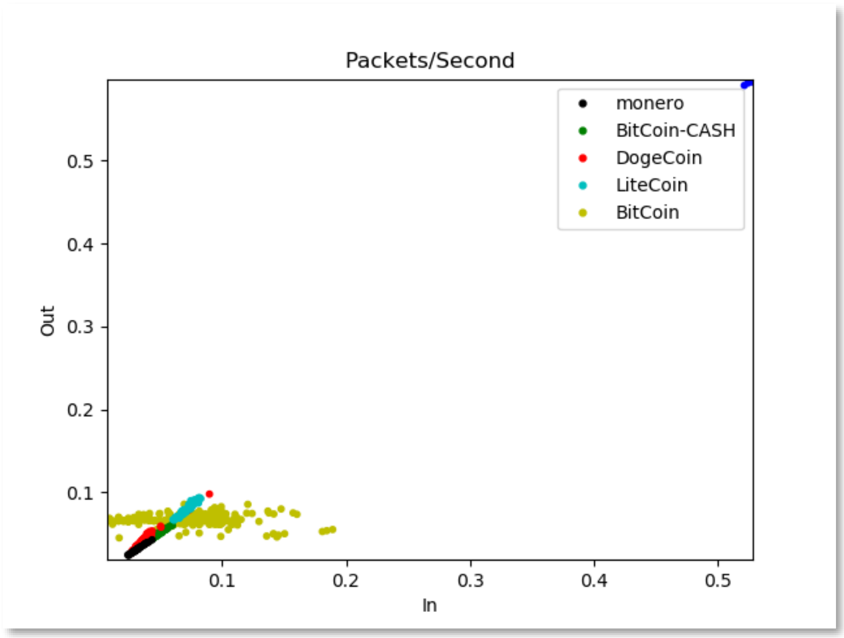


Figure 9 - This plot shows the amount of packets per second sent and received each second. Created with matplotlib. [31]

As it can be seen in the plots in figures 7, 8 and 9, it's easy to differentiate the data from the different coins, each one creates a separate cluster. Note that Ethereum data for packets/second and bits/second is way higher than the rest of cryptocurrencies. This is probably related to the fact that its block time is really small while it maintains a similar amount of transactions for each block and block size than the rest of cryptocurrencies. (Table 7).
Another thing that is shown in the plots is how Bitcoin has results all over the place.
This might be because Bitcoin is the one with the largest number of transactions, with on average ten times more transactions than the other coins [26], which can affect the number of mining.notify messages that the pool server sends to the miners so that they can verify the new transactions while mining. Since the amount of transactions is not constant, as you can observe in the Figure 10, this creates variations on the bitrate and the number of packets that are received when mining.

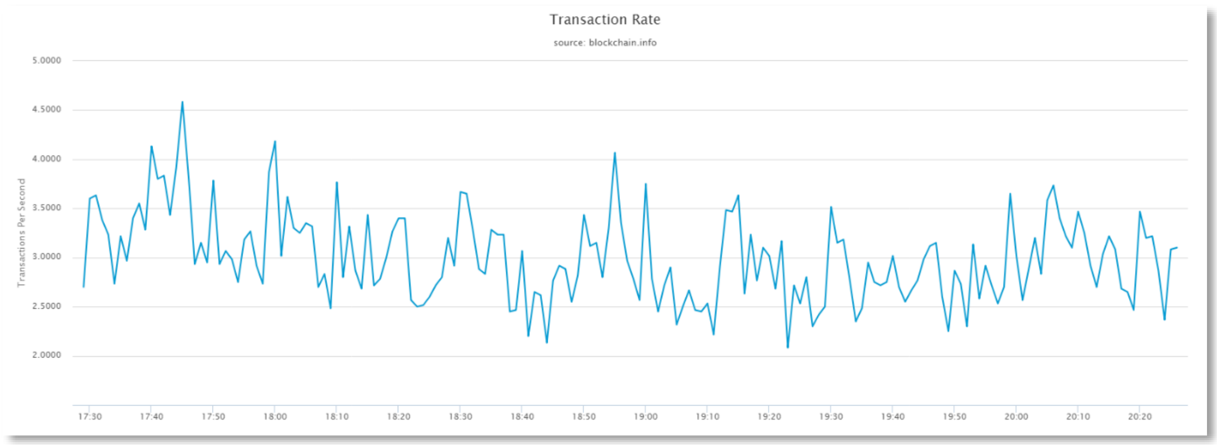


Figure 10 – Number of transactions per second in Bitcoin. Extracted from blockchain.com. [27]

Even though we can't see an obvious way to differentiate between bitcoin and the other coins, I believe that with the help of machine learning we will be able to find a model that correctly classifies mining data in the coins they are mining.

8.2 **Learning Models**

As I stated before, the amount of mining traffic data I managed to collect is rather small. If I divided the data in training, testing and validation tests I would end up too little mining data in each set. Therefore, I decided to use cross-validation to create the models.
Cross-validation consists on doing multiple tests when creating the model, and dividing the data in different subsets that are used to train and validate the models. For each test one random subset is used to validate the data while the others are used to train the model. This way you get one model for each test. Then, the resulting model that achieved the best results is used. This allows to create a model with a small amount of data without getting an overfitted model created by an insufficient validation data or lose some information in the process that could have helped build a better model. [28]
I decided to test the following models:

- Support Vector machine
- CART
- C4.5
- Naïve Bayes

8.2.1 Support vector machines

This method creates models by dividing the classes geometrically, creating a hyperplane between the two classes that then is used to classify new additions to the model. Depending on which side of the plane the new data point falls in, it will be classified on one plane or the other. [29]
The implementation used will be SMO, a solution that trains support vector machines.

8.2.2 CART

Cart, or Classification and Regression Trees, is a model to create binary decision trees. This model does not use Information gain as reference to create the splits in the tree, but uses Gini impurity, which is faster to compute. [29]

8.2.3 C4.5

This model creates trees by recursively adding leaves if only one class is left (or only a small number of nodes remain), or branches if not, using either information gain or gain ratio, depending on the algorithm used. [29]
The algorithm that I will use is J48, an implementation of C4.5 that uses gain ratio when deciding how to create the branches.

8.2.4 Naïve Bayes

A model that works by creating a set of scores that rate each of the input variables and classify it depending on the resulting score, this is a simple classification model that is easy to implement and that, in the right conditions, can work really well. [29]

8.3 Weka Tests

8.3.1 Mining classification

To test the different models, I decided to use Weka. Weka is a tool for data analysis that contains multiple machine learning algorithms and allows the user to apply them to his data and compare the results to obtain the best one for the task at hand. [30]
To test this algorithms, I used the 691 stratum collected flows and 23670 flows from the flows collected by other applications.

After testing the five algorithms with Weka using the dataset with the traffic collected at home, I obtained the results shown in table 9.

Algorithm	TP	FP	TN	FN	Accuracy	Precision	Recall
SMO	0	0	23670	600	0.9753	0.0000	0.0000
CART	586	8	23662	14	0.9991	0.9865	0.9767
J48	590	5	23665	10	0.9994	0.9916	0.9833
Naïve Bayes	597	10371	13299	3	0.5726	0.0544	0.9950

Table 9 – Results of the model tests with Weka using created data.

The first thing that can be noticed from the table is that both Naïve Bayes and SMO results were far from optimal.
SMO classified all the data as not coming from stratum applications. This may be caused because the data coming from other applications was distributed all over the place and the algorithm didn't manage to create a hyperplane that could separate the stratum flows from the others without creating lots of false positives. Naïve Bayes actually got the bests results in terms of Recall, since it only failed to classify 3 stratum flows, but was incapable of classify data as not Stratum correctly, which ended up with very bad results overall, since most of the data that will be classified will be from other applications.
The models based on trees worked pretty well, and they got similar results. The best resulting tree is the one created by J48, shown in figure 11.

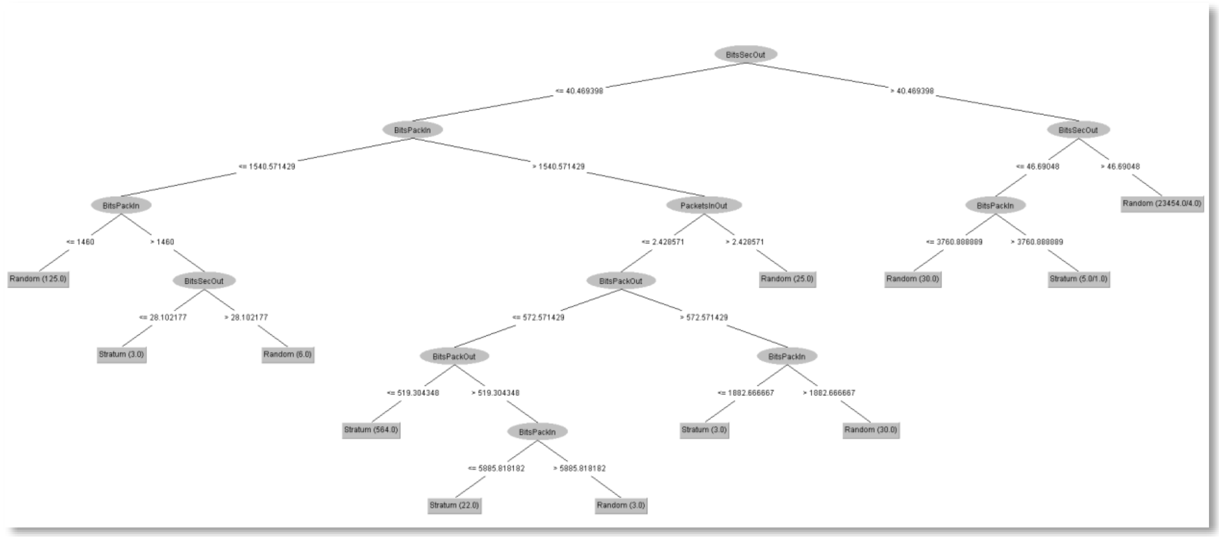


Figure 11 – Decision tree created by the J48 algorithm.

With this we learned that it is possible to classify stratum flows from stratum data, but this model only used traffic created artificially, and does not accurately represents traffic from an actual network. Because of this, this model may not perform well on a real environment.

To test this, I acquired 2 hours of traffic capture from a 10 Gbps access link of a large university network, which connects around 25 faculties and 40 departments (geographically distributed in 10 campuses) to the Internet. The flows from this traffic were tagged to show if they were from mining applications or not using deep packet inspection to identify miners.
This was done by searching for two regular expressions in the traffic, one to detect mining methods used by the clients and one used to identify methods used by the server. This way we can identify who is the client and who the server in the connection.
The regular expression used to detect mining connections client-server and server-client are, respectively:

- *\"method\" ?: ?\"(mining\.authorize|mining\.get_transactions|mining\.subscribe |mining\.submit|getblocktemplate|submitblock)\"*
- *\"method\" ?: ?\"(mining\.notify|mining\.set_difficulty)\"*

Since we used dpi to detect the mining flows, we don't know if we missed some flows that were actually mining traffic but we couldn't detect because they used encrypted protocols or other protocols that were unknown to us. This means that some flows might be wrongly tagged as other traffic.
As shown in table 10, even though it correctly classified 10 out of the 14 flows coming from stratum, it was unable to correctly classify a 7379 flows coming from other traffic, leading to an accuracy of 0.99589, a recall of 0.71429, and a precision of 0.00013. This amount of false positives would make using this method to detect miners unusable.

Classified as:

Stratum	Other		
10	4	Stratum	Tagged
7379	1788015	Other	as:

Table 10 - Confusion matrix of the tests with the first model.

Even the bad results, this doesn't mean that this method can't be used to identify miners using Netflow data. Looking at the wrongly classified flows and comparing them with the flows used to create the data we can notice that while the wrongly classified mining flows have similar properties to those used to create the model, most of the false positives are really different from the mining flows that we collected earlier. This is probably caused by a model that is not complex enough, and a more complex model created with more data would be able to correctly classify this data. The discrepancies between the flows from can be seen in the plots in Figures 12, 13 and 14.

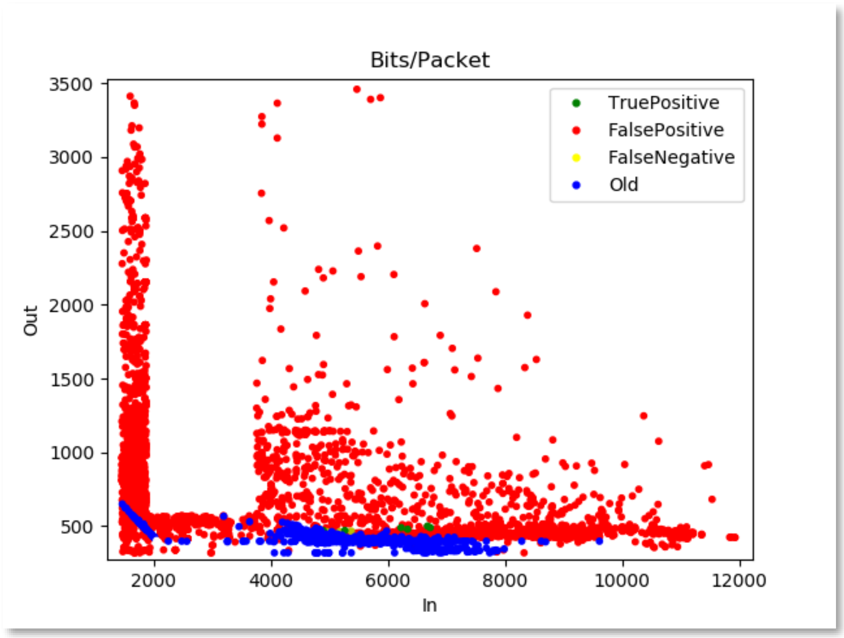


Figure 12 - Plot showing the bits/packet of the mining flows used to create the model (blue), the true positives (green), the false negatives (yellow) and the false positives (red). The vertical axis represents the outbound data (client-server), and the horizontal the inbound data of the connection(server-client).

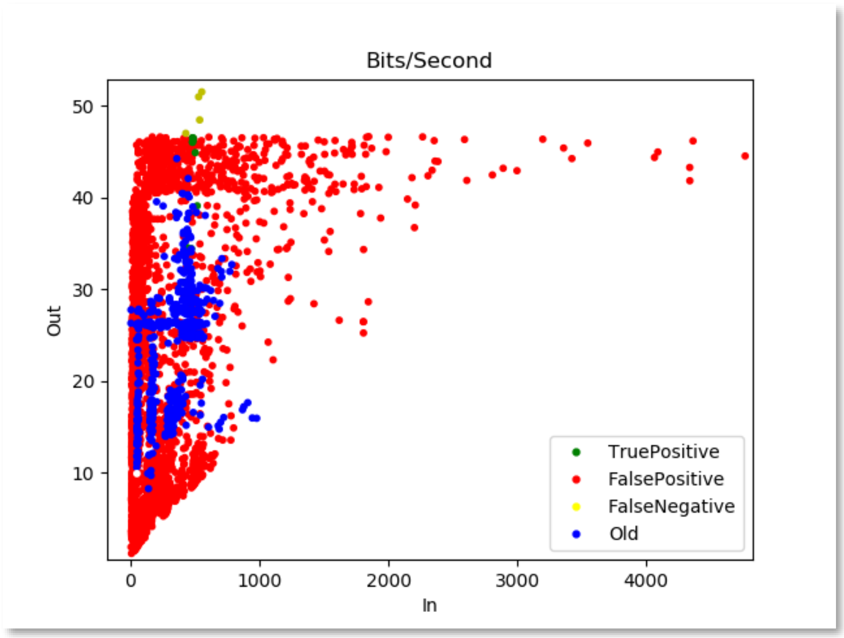


Figure 13 - Plot showing the bits/second of the mining flows used to create the model (blue), the true positives (green), the false negatives (yellow) and the false positives (red). The vertical axis represents the outbound data (client-server), and the horizontal the inbound data of the connection(server-client).

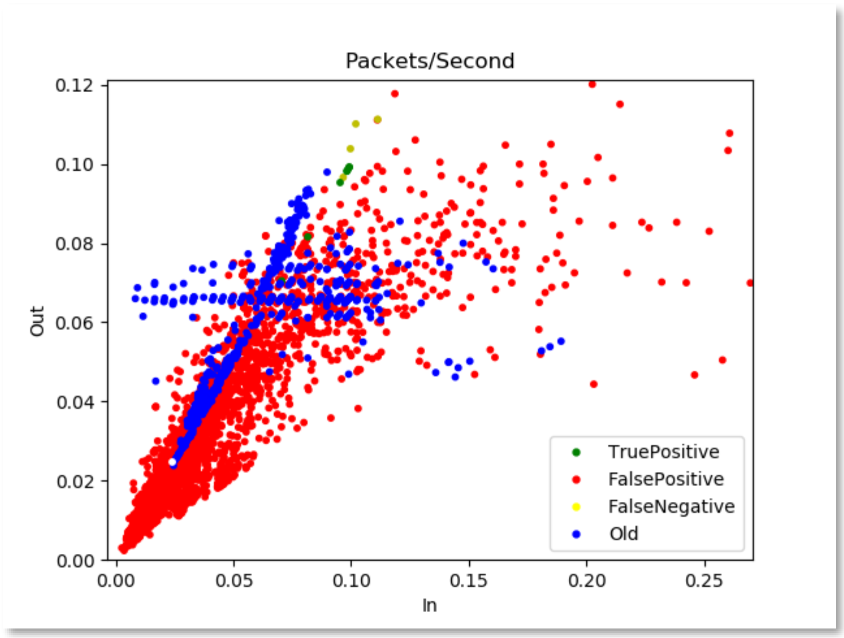


Figure 14 - Plot showing the packets/second of the mining flows used to create the model (blue), the true positives (green), the false negatives (yellow) and the false positives (red). The vertical axis represents the outbound data (client-server), and the horizontal the inbound data of the connection(server-client).

For this reason, I decided to create another model using the data from this dataset. Since there are only 14 stratum flows, which is not enough to correctly train a model, therefore I will add the data from stratum flows that was created earlier to this dataset. The resulting dataset contains 705 flows from mining traffic and 1795394 flows from unidentified applications. After testing with Weka using this new dataset we get the results shown on table 11.

Algorithm	TP	FP	TN	FN	Accuracy	Precision	Recall
SMO	0	0	1795394	705	0.9996	0.0000	0.0000
CART	657	13	1795381	48	0.9999	0.9810	0.9320
J48	694	15	1795379	11	0.9999	0.9790	0.9840
Naïve Bayes	702	182775	1612619	3	0.8982	0.004	0.996

Table 11 – Results of the model tests with Weka using the university campus traffic.

The results are similar from the ones in the first test. The Support Vector Machine and the Naïve Bayes failed the same way, while both tree models got decent results. The best model overall happened to be the one created by J48, the implementation of C4.5. It should be noted that while the number false positives and false negatives increased slightly in comparison with the first model results, the total amount of flows from other applications increased by more than 70 times.
Figure 15 shows the resulting tree created by C4.5. This tree is much more complicated than the one created by the first dataset shown in Figure 11.

8.3.2 Coin Classification

The same methodology was used to create a model to classify the flows between the different cryptocurrencies, using the 691 mining flows. The results are shown in table 12, showing the accuracy and the average F Score of each model.

Algorithm	Accuracy	Average F Score
SMO	0.844	0,815
CART	0.973	0.972
J48	0.978	0.978
Naïve Bayes	0.981	0.981

Table 12 - Results of the model tests for the coin classificator with Weka.

All models achieved decent results when classifying the different cryptocurrencies. In this case, Naïve Bayes managed to correctly classify

98% of the flows, showing that using the information from Netflow we are able to extract the cryptocurrency being mined with high confidence.

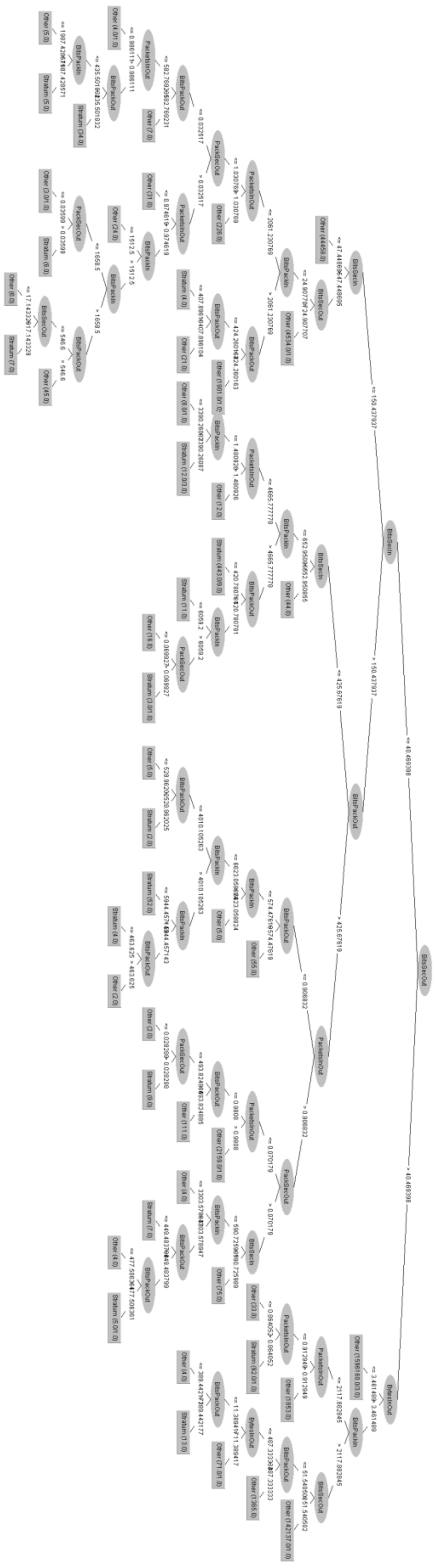


Figure 15 - Tree created with C4.5 using the dataset coming from the university campus.

9 Results

During the previous chapter we created a model that was able to detect mining traffic using only Netflow. The results showed that using this new method to detect miners is possible.
Using the new model against the data coming from the university traffic again shows a great improvement from the results of the first model.
Shown in table 13 are the results of the model using this dataset.

Classified as:

Stratum	Other		
13	1	Stratum	Tagged
15	1795379	Other	as:

Table 13 - Confusion matrix of the tests with the definitive model.

In comparison with the first results, this new model results have improved greatly from the results coming from applying first model to the same dataset. This results lead to an accuracy of 0.99999, a recall of 0.92857 and a precision of 0.46429. The precision is still low, but if it is compared with the 0.00013 of the previous model, it is a good improvement.

10 Conclusions

The aim of this research project was to create a way to find miners using Netflow data and investigate what information could be extracted from this data. After testing models with two datasets and four different machine learning algorithms I created two machine learning models that are able to find mining traffic and detect the currency being mined respectively, using only the data given by Netflow.

The results have shown that both models had a decent performance. I believe it could be added to a security solution, like a SIEM, which could correlate the events created by the model and generate alerts of miners in the network with high confidence.

The model used to classify between cryptocurrencies has shown the ability to classify between the different types of coins we had data on. Sadly, because of time limitations the number of cryptocurrencies used in the dataset is small. This has been caused by the slow rate at which flows from these applications are created.

10.1 Future work

The next step in this investigation will be to collect more data from more different cryptocurrencies and pools and use it to create new models.

The addition to this new data will give a wider look at the mining data as a whole, and make the binary classifier more accurate. This would also benefit the cryptocurrency classification model, which would be able to detect more cryptocurrencies.

In addition, data from cryptojacking malware will be analysed and compared with the known data. Afterwards, in case this data is too different from the original mining data to be detected, this should be added to the dataset to create a new model that can detect this traffic. Furthermore, tests could be made to see if it is possible to discern between normal mining traffic and cryptojackers using the Netflow data.

11 References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” bitcoin.org, pp. 1-9, October 2008.

[2] A. Berentsen and F. Schär, “A Short Introduction to the World of Cryptocurrencies,” Federal Reserve Bank of St. Louis , 2018.

[3] “What is a Decentralized Application?,” in Decentralized Applications: Harnessing Bitcoin's Blockchain Technology, O'Reilly Media, Inc, July 2016, pp. 1-8.

[4] L. Wang and Y. Liu, “Exploring Miner Evolution in Bitcoin Network,” NYU Polytechnic School of Engineering, 2015.

[5] A. Mizrahi, “Cryptojacking Rises as Ransomware Declines, Cyber Security Researchers Find,” Bitcoin.com, 1 July 2018.

[6] M. Rosenfeld, “Analysis of Bitcoin Pooled Mining Reward,” Cornell University, 2011.

[7] J. D'Herdt, “Detecting Crypto Currency Mining in Corporate Environments,” SANS Institute, 2015.

[8] K. Coffman and A. M. Odlyzko, Internet Growth: Is There a 'Moore's Law' for Data Traffic?, SSRN Electronic Journal, 2000.

[9] Cisco public, “Cisco Visual Networking: Index: Forecast and Trends, 2017–2022,” Cisco, 2018.

[10] “NetFlow Export Datagram Format,” 14 September 2007. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html. [Accessed 21 November 2018].

[11] “Bitcoin Miner May Indicate Malicious Activity,” 9 October 2013. [Online]. Available: https://tools.cisco.com/security/center/viewAlert.x?alertId=31186. [Accessed 22 November 2018].

[12] “Bitcoin Miner Client Activity,” 30 November 2015. [Online]. Available: https://tools.cisco.com/security/center/viewIpsSignature.x?signatureId=2737&signatureSubId=0. [Accessed 22 November 2018].

[13] “Port 8333 Details,” 16 June 2017. [Online]. Available: https://www.speedguide.net/port.php?port=8333. [Accessed 22 Septebmer 2018].

[14] “Startum Bitcoin Mining Protocol,” 13 May 2014. [Online]. Available: https://tools.cisco.com/security/center/viewIpsSignature.x?signatureId=4238&signatureSubId=0. [Accessed 22 November 2018].
[15] D. Delaney, “How to Detect Cryptocurrency Mining Activity on Your Network,” 3 May 2018. [Online]. Available: https://www.netfort.com/blog/detect-cryptocurrency-mining-activity/. [Accessed 21 September 2018].
[16] Ars Technica, “How to keep your ISP’s nose out of your browser history with encrypted DNS,” 4 August 2018. [Online]. Available: https://arstechnica.com/information-technology/2018/04/how-to-keep-your-isps-nose-out-of-your-browser-history-with-encrypted-dns/. [Accessed 29 November 2018].
[17] P. O. S. J. B. Domhnall Carlin, “Detecting Cryptomining Using Dynamic Analysis,” 2018 16th Annual Conference on Privacy, Security and Trust (PST), 28-30 Aug 2018.
[18] E. V. V. M. M. L. C. K. H. B. G. V. Radhesh Krishnan Konoth, “MineSweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense,” Conference on Computer & Communications Security, Oct 2018.
[19] A. d. Vries, “Bitcoin's Growing Energy Problem,” Joule, 2018.
[20] Cisco Systems, “CISCO IOS NETFLOW,” Cisco Systems, 2004.
[21] D. Medina, “An IDS Using NetFlow Data”.
[22] W. W. Royce, “Managing the development of large software systems,” 1970.
[23] D. Wells, “Extreme Programming,” Don Wells, 8 October 2013. [Online]. Available: http://www.extremeprogramming.org/. [Accessed 23 November 2018].
[24] SlushPool, “Stratum Protocol,” 2012. [Online]. Available: https://slushpool.com/help/topic/stratum-protocol/. [Accessed 20 1 2019].
[25] Plixer, “NetFlow Version 5,” 15 12 2016. [Online]. Available: https://www.plixer.com/support/netflow-v5/. [Accessed 10 11 2018].
[26] bitinfocharts, “Block Time historical chart,” [Online]. Available: https://bitinfocharts.com/comparison/confirmationtime-btc-ltc-bch-xmr-doge.html. [Accessed 3 February 2019].
[27] blockchain.com, “Transaction Rate - Blockchain,” blockchain, 10 03 2019. [Online]. Available:

https://www.blockchain.com/charts/transactions-per-second?timespan=3h. [Accessed 10 03 2019].
[28] S. Marsland, Machine Learning, An Algorithmic Perspective, Second Edition, Boca Ratón: Chapman & Hall/CRC, 2015.
[29] Q. Yang, H. Motoda, A. Ng, B. Liu, Z.-H. Zhou, P. S. Yu, G. J. McLachan, M. Steinbach, D. J. Hand, D. Steinberg, X. Wu, V. Kumar, R. J. Quinlan and J. Ghosh, "Top 10 algorithms in data mining," Knowl Inf Syst, vol. 13, no. 1, pp. 1-37, 2008.
[30] M. A. H. a. I. H. W. Eibe Frank, The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, 2016.
[31] Project, Matplotlib, "Matplotlib," [Online]. Available: https://matplotlib.org/. [Accessed 27 02 2019].
[32] T. K. Ho, "Random Decision Forests," 3rd International Conference on Document Analysis and Recognition, 15 August 1995.