Data preparation for using machine learning to detect attacks on networks

Student: Maria Eleni Kokkini 3170070

Professor: Yiorgos C. Stergiopoulos

Semester: winter semester 2020-2021

Contents

Machine learning:

Definition:

Due to the author Lisa Tagliaferri, machine learning is a brunch of artificial intelligence [1]. The aim of this field is to give to computers the ability to learn and make decisions without being explicitly programmed. Machine learning is closely related to mathematics. With the help of mathematics, machine learning constructs algorithms that enable their models to predict and make decisions. This field is used in a variety of applications and fields (e.g. email filtering, online customer support, etc). It is used especially when it is extremely difficult to employ conventional algorithms to perform the given task. The constructed algorithms allow the intelligent programs to operate tasks effectively. In conclusion, machine learning gives the computer “The ability to learn”. [2]

Types of machine learning algorithms:

There are many categories of machine learning algorithms, that they follow different approaches, support different types of input and output data and they also solve different types of problems. These are the different types of algorithms:

- **"Supervised learning"**: This type of algorithms is the most popular in machine learning. In this type of learning, we give input variables and the corresponding output variables and we employ an algorithm to learn the mapping between the values. The optimal result is to approximate the mapping function so well, in order to give new input data and predict the output. [3]
- **"Unsupervised learning"**: in this type of learning we only give input data without giving any corresponding examples for the output. The intention for this learning technique is to pattern the given form of data in order to gain more information about the given dataset. There are not false or right answers and the unsupervised model have to discover alone as many patterns as possible. [3]
- **"Semi-supervised learning"**: this type of learning falls between the above categories. Some of the training data have corresponding output values, whereas other do not. Many machine learning researchers have figured out that unlabelled data (input data without corresponding output), when they are used with a small quantity of labelled data (input data with corresponding output), can improve the accuracy of learning and counter the disadvantages of the before-mentioned categories. [4]
- **"Reinforcement learning"**: this is category of algorithms, that they train a learning model to make multiple decisions. These decisions, are taken under complex of difficult conditions. The trained machine applies "trial and error" techniques to figure out a possible fix to the problem. This is achieved by giving either penalties or rewards to the current action. The goal is to maximize the total reward. [5]
- **"Self-learning"**: The system supports pattern recognition and learning from unlabelled examples. Under these conditions, the system gradually becomes more and more intelligent, only with the help of AI techniques. [6]
- **"dictionary learning"**: This learning method focuses to the construction of a data frame. In this sparse dictionary, data vectors are modelled as linear combinations of features. This method is NP-hard. [7]
- **"Anomaly detection"**: this type of algorithms identifies unusual elements, that seems suspicious and that they have significant differences from the rest data. [8]
- **"Robot learning"**. Is the solution of machine learning to cope with robotic problems. The applications of this type of learning is mechanical design optimization, high-level decision making, etc… [9]
- **"Association rules"**: this type of learning discovers connections and associations between characteristics in datasets. This method is one of the commonest among machine learning techniques. [10]

Machine learning models:

First of all, we have to clarify the difference between a machine learning algorithm and a machine learning model. As an algorithm we define a process that runs on a set of data in order to establish a machine learning model. In contrast, a model is the result of a machine learning algorithm, that had processed a dataset. Usually, a machine learning model, acquires a variety of data in order to perform decently. [11]

- **"Artificial neural networks":** a neural network is a network of mathematical equations. It takes one or more input variables, proceeds them with the mathematical equations and results one or more output variables. This model consists of connected Units, the artificial neurons. Each unit receives a signal, processes it and then the signal is transferred to next neurons that they are connected with the previous one. [12]
- **"Decision trees"**: their employment, is to represent the decisions of the system and visualize them. This model maps observations about data to conclusions. The result is to guess the value of a target example according to the input examples. While learning a decision tree, the dataset is gradually split into multiple subsets according to the characteristic that is examined. This procedure is recursive. The recursion stops, when the subset of a node has the same value as the examined value. [1]
- **"Support vector machines (SVMs)":** The aim of an SVM in to define hyperplane in a space with as many dimensions as the total number of the features. There is more than one possible hyperplane that are able to classify the given examples. The number of the dimensions of the hyperplane is related with the number of the features. [13]
- **"Regression analysis"**: the goal of a regression model is to create a mathematical equation that defines the outcome variable as a function of one ore multiple predictor variables. This equation, is used to predict the outcome of the new predictor variables. [14]
- **"Bayesian networks"**: these networks are formed by a directed acyclic graph and a bunch of parameters. The graph represents the connections among the variables and the nodes. The parameters of the graph express the "conditional probability distribution" related with every single node. This kind of networks are extracted from the given dataset. [15]
- **"Genetic algorithms"**: these algorithms run on a population of potential solutions and they are characterized as stochastic searching algorithms. Each instance of the population is encoded as a gene. To produce new solutions, we simply mating two genes of the current population or just mutate a gene. From the existing population the most optimal genes and chosen to breed. The less optimal are dismissed. [16]
- **"K-nearest neighbours":** is a pattern recognition model that is employed to classify the given input. We choose a small integer (k) and we register the new-coming element to the class that is commoner among the k closest neighbours. [1]

Data sets used in machine learning:

When we want to use a given dataset, to provide it in a machine learning model we split it in three different categories; Training data, Validation data and Test data. There is much more need of training data instead of the other two categories. So, we characterize the majority of data as training data and we divide the rest data in the other two categories. There are many methods that they are used to operate this task.

- **“Training data”**: this dataset consists of a bunch of examples, that they are employed to train the machine learning model. The model learns from the mapping of the training data. The prementioned procedure is called fitting. [17]
- **“Validation data”**: they are also known as development set. This set is essential when we want to avoid overfitting. The examples in this dataset are used to tune the hyperparameters. [17]
- **“Test data”:** is a set of examples, independent of the training dataset, used only to come up with an unprejudiced evaluation. This set is not used for fitting at all, is used only after the completion of the training. [17]

Overfitting:

Overfitting describes the situation, where a model imprints the training examples extremely satisfactorily. An overfitted model fails to generate the obtained knowledge. Overfitting occurs when the details and the peculiarities of the specified data are memorized by the training model. As result, the noise of some examples is learned. This situation, prevents the model from having a decent performance in different data. [18]

Underfitting:

Underfitting describes the situation, where a model is inadequately trained and cannot model the given training set nor any other data. Such a model has always insufficient performance. Underfitting can be prevented by diminishing the used features and giving more data to the system. [18]

Main Topic: data preparation for using machine learning to detect attacks on networks

The main purpose was to analyse datasets and divide their packets by the IoT protocols they use. We focused on three different protocols and created a csv file for each of them. These protocols were mqtt, coap, amqp from application layer. I also extracted basic characteristics of each packet and transfer them to the corresponding file. The implementation is based on python programming language and scapy library.

What are these protocols? These protocols transfer data over internet and allow end users to extract information. More specific: [19]

- “Mqtt”: is the most commonly used IoT protocol, which stands for Message Queuing Telemetry Transport. Mqtt runs over Tcp/Ip and is really light weighted. It is based on client- broker architecture, where there is no communication among clients but each client publics/ subscribes messages to the broker. It is appropriate for limited resources.

- “Amqp”: stands for Advanced Message Queuing Protocol and is more usually used over Tcp/Ip. It is a quite heavy protocol and is commonly used for message exchanging in industry. It is not appropriate for limited resources.
- “Coap”: stands for Constrained Application Protocol and it runs over Tcp or Udp. Is used for web transactions and includes the basic functions of http protocol. Over Udp has the ability to broadcast and multicast data.

About the code:

The code is divided into multiple methods and each method has a specific functionality. The basic method that I created is called “pcap_pkt_reader” and it summons the majority of the other methods in the file. This method checks whether the given pcap file exists and if it does, it lists the given packets. After the packets are listed, I locate if the packet uses Tcp or Udp protocol and for both cases I follow the same process. Firstly, I check whether the packet runs over ethernet/cooked-linux or another protocol and I get the mac addresses of the packet, using my method “find_first_layer_protocol”. After that I detect the Ip addresses and the corresponding ports, using the methods “get_ip_addresses” and “get_ports”. Then, I capture tcp/udp payload in order to examine the existence of IoT protocols. As we know, the higher level a protocol is, the inner its header in the packet is. I converted the payload into string format because the previous type was a Scapy type and it was quite uncomfortable to work with. I make sure that the payload is not empty, otherwise there is no use of examining this packet. If the payload is empty, it means that there is no header for a higher layer. Two characteristics that I embedded in my code later was the date and the time the source sent each packet, using “get_date_and_time” method. These characteristics seem to be vital for any process of the packets. Both date and time needed a reformation in order to be converted in human readable form.

Code sample:

```
{
  """get the physical layer"""
   if "Ether" in pkt:
      mac_src_in_bytes = str(pkt["Ether"].src)
      mac_src = prettify(mac_src_in_bytes)
      mac_dst_in_bytes = str(pkt["Ether"].dst)
      mac_dst = prettify(mac_dst_in_bytes)
      layer = "Ethernet"
   elif "CookedLinux" in pkt:
      mac_src_in_bytes = str(pkt["CookedLinux"].src)
      mac_src = prettify(mac_src_in_bytes)
```

```
        mac_dst = None
        layer = "Cooked Linux"
    else:
        mac_src = None
        mac_dst = None
        layer = "other"

    """get src and dst  ip"""
    src_ip = pkt["IP"].src
    dst_ip = pkt["IP"].dst

    """get ports: dst and src"""
    src_port = pkt[type_protocol].sport
    dst_port = pkt[type_protocol].dport

 """get date and time"""
 pkt_time = pkt.time
 p_time = datetime.fromtimestamp(pkt_time).strftime('%Y-%m-%d ')  # format: 2020-10-08
 p_hour = datetime.fromtimestamp(pkt_time).strftime('%H:%M:%S')  # format: 22:40:41

}
```

I first investigate the existence of Amqp protocol. This is checked by my method “is_amqp” [20], with the instance of the packet, its payload and its transfer protocol as parameters. To construct this method, I first found out the architecture of an amqp packet. I figured out that the first packet that establish the connection will conclude the word “AMPQ” inside the payload. In addition, each amqp packet has a fixed header (8 bytes) and always ends in the same way. The frame always ends with this specific byte: %xce. Due to Iana organisation, amqp packets always use 5671 or 5272 port. This information was the criteria for me to classify a packet as amqp or not.

Code sample:

```
{
    if "AMQP" in payload:  # initiates the connection (first amqp packet)      answer = "yes"
     if len(payload) >= 8:
```

```
        end_code = "\\xce"
        index = payload.find(end_code)
        right_index = len(payload) - 5
        if index == right_index:            answer = "yes"
    if (src_port == 5671) or (dst_port == 5671) or (src_port == 5672) or (dst_port == 5672):
     answer = "yes"
}
```

If the packet does not use Amqp, then I check if it is an mqtt packet. I found again the architecture of the packet an created an algorithm based on that. I examine as before if the word mqtt is in the payload. After that I search whether the first byte of the packet is one of the possible mqtt header combinations. I have listed all the possible combinations in another method, called "create_mqtt_first_byte" [21]. This method is based on the information that each mqtt packet has a fixed protocol type and a fixed combination of flags. So, I created a list with all possible combinations of types and flags and I use the list to match the first byte of the given packet with one of these codes. If the packet matches with one of these codes then we mark it as mqtt. Otherwise we examine ports. If the packet uses 1883 or 8883 port, then due to Iana organisation the packet runs over mqtt.

Code sample:

```
{
   if "MQTT" in payload:          answer = "yes"
   for mqtt_code in list_with_codes:  # e.g xe0
      exact_code = "\\" + mqtt_code + "\\"  # e.g \xe0\
      if exact_code in payload[:7]:            answer = "yes"
    if (src_port == 1883) or (dst_port == 1883) or (src_port == 8883) or (dst_port == 8883):
      answer = "yes"
}
```

The last check is about coap protocol. Coap protocol has always a fixed header of 4 bytes. The first two bits are the version, which must always be 01. Then there are four different coap packet types, two bits each. Then there is a random token length, which is a number between 0-8 and needs 4 bits. I combine those cases and then I check if the first byte of the packet match with one of those combinations. If there is a match then I mark this packet as coap otherwise, I scan the ports. Iana has recorded that coap protocol uses the port 5683.

Code sample:

```
{
```

```
if len(payload) >= 4:
    for byte in first_byte:
        if byte in payload[:7]:            answer = "yes"
    if (src_port == 5683) or (dst_port == 5683):        answer = "yes"
}
```

For each case I store the vital information in a corresponding list, so that I can pass it in a csv file. Those characteristics are extracted with the help of "extract_characteristics_from_packet" method. This method returns the size of the given packet, whether the packet is encrypted by checking the existence of ssl/tls, the size of the payload, the ratio of the payload, the ratio of the previous packet, the size of the previous packet and the payload in a hexadecimal format.

Code sample:

```
{
   pkt_size = len(pkt)
   if 'SSL/TLS' in layer.name:            is_encrypted = 1
   payload_size = len(pkt["Raw"])
   payload_ratio = (payload_size / pkt_size) * 100
   previous_ratio = (pkt_size / previous_packet_size) * 100
   packet_time_diff = Decimal(pkt.time) - Decimal(previous_packet_time)
   payload = pkt[type_protocol].payload
   payload_fix_format_type = binascii.hexlify(bytes(payload))
}
```

Lastly, I produce three different csv files. I create a coap file using "write_coap_file", an mqtt file using "write_mqtt_file" and an amqp file using "write_amqp_file". These methods are using the same algorithm. I firstly construct the header and then I zip the lists with the information of each protocol, in order to create lines with tuples of the characteristics mentioned above. Then I pass these lines into the file.

Code sample:

```
{
protocol_file = 'protocol_information.csv'
   with open(protocol_file, 'w', newline='') as csvfile:  # automatically close the file
      csvfile = csv.writer(csvfile, delimiter=',')
```

```
        csvfile.writerow(
            ['application_protocol', 'transport_protocol', 'layer', 'mac_src', 'mac_dst', 'src_ip', 'dst_ip',
             'src_port', 'dst_port', 'pkt_size', 'is_encryptd', 'payload size', 'payload_ratio', 'previous_ratio',
             'packet_time_diff', 'payload', 'packet_date', 'packet_time'])
        for (variables) in zip(lists):
            csvfile.writerow([variables])
}
```

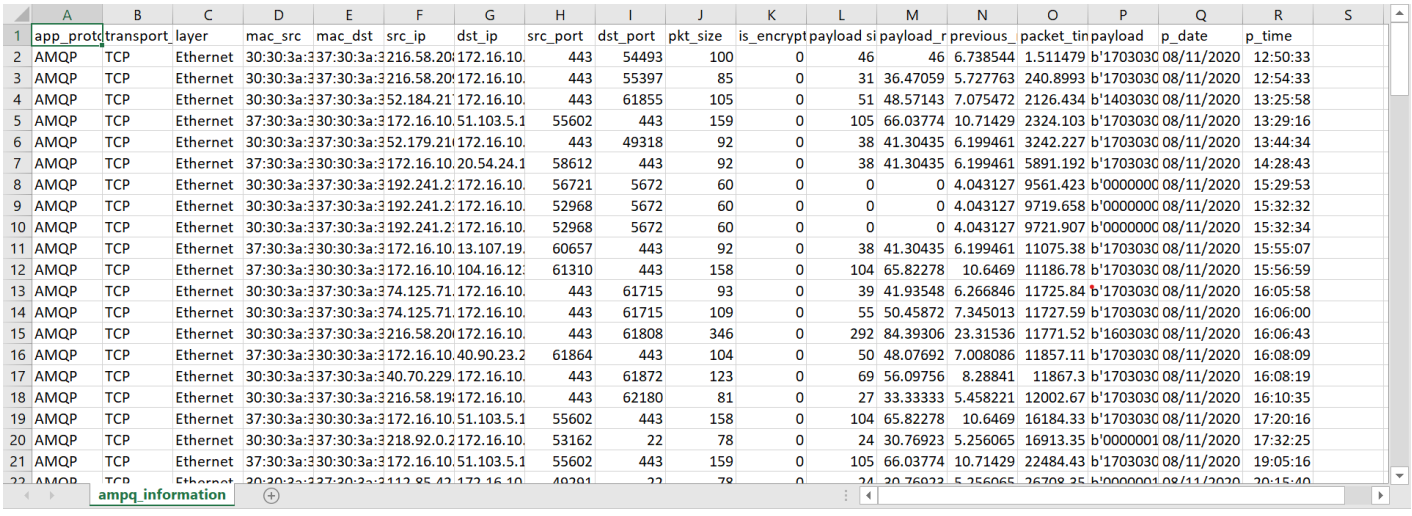
Methods that I used and they are not mentioned above:

1) create_hex_from_bits: the purpose of this method is to convert series of bits into heximal form. This is useful in order to match sequences in packets.
2) create_coap_first_byte: I created this method in order to create all the possible combinations of the first coap byte. This method is used in the function is_coap, to identify coap packets.
3) Prettify [22]: this method converts byte to hex format. This is useful about the mac addresses.
4) fill_coap_info: with this method I pass in the coap list the information about each coap packet, in order to use the content of the list to create the coap file.
5) fill_mqtt_info: in correspondence with the above method.
6) fill_amqp_info: in correspondence with the former one.

Capturing examples:

In this section I will show a part of the output of my code and I will also give some explanations. For these results I have used a file of 70,286 KB, which includes 130506 packets.

- My algorithm captured 125 packets, that they use the AMQP protocol.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	app_protc	transport_	layer	mac_src	mac_dst	src_ip	dst_ip	src_port	dst_port	pkt_size	is_encrypt	payload si	payload_r	previous_	packet_tin	payload	p_date	p_time	
2	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	216.58.20	172.16.10.	443	54493	100	0	46	46	6.738544	1.511479	b'170303(	08/11/2020	12:50:33	
3	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	216.58.20	172.16.10.	443	55397	85	0	31	36.47059	5.727763	240.8993	b'170303(	08/11/2020	12:54:33	
4	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	52.184.21	172.16.10.	443	61855	105	0	51	48.57143	7.075472	2126.434	b'140303(	08/11/2020	13:25:58	
5	AMQP	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	51.103.5.1	55602	443	159	0	105	66.03774	10.71429	2324.103	b'170303(	08/11/2020	13:29:16	
6	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	52.179.21	172.16.10.	443	49318	92	0	38	41.30435	6.199461	3242.227	b'170303(	08/11/2020	13:44:34	
7	AMQP	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	20.54.24.1	58612	443	92	0	38	41.30435	6.199461	5891.192	b'170303(	08/11/2020	14:28:43	
8	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	192.241.2	172.16.10.	56721	5672	60	0	0	0	4.043127	9561.423	b'000000(	08/11/2020	15:29:53	
9	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	192.241.2	172.16.10.	52968	5672	60	0	0	0	4.043127	9719.658	b'000000(	08/11/2020	15:32:32	
10	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	192.241.2	172.16.10.	52968	5672	60	0	0	0	4.043127	9721.907	b'000000(	08/11/2020	15:32:34	
11	AMQP	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	13.107.19.	60657	443	92	0	38	41.30435	6.199461	11075.38	b'170303(	08/11/2020	15:55:07	
12	AMQP	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	104.16.12	61310	443	158	0	104	65.82278	10.6469	11186.78	b'170303(	08/11/2020	15:56:59	
13	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	74.125.71	172.16.10.	443	61715	93	0	39	41.93548	6.266846	11725.84	b'170303(	08/11/2020	16:05:58	
14	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	74.125.71	172.16.10.	443	61715	109	0	55	50.45872	7.345013	11727.59	b'170303(	08/11/2020	16:06:00	
15	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	216.58.20	172.16.10.	443	61808	346	0	292	84.39306	23.31536	11771.52	b'160303(	08/11/2020	16:06:43	
16	AMQP	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	40.90.23.2	61864	443	104	0	50	48.07692	7.008086	11857.11	b'170303(	08/11/2020	16:08:09	
17	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	40.70.229.	172.16.10.	443	61872	123	0	69	56.09756	8.28841	11867.3	b'170303(	08/11/2020	16:08:19	
18	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	216.58.19	172.16.10.	443	62180	81	0	27	33.33333	5.458221	12002.67	b'170303(	08/11/2020	16:10:35	
19	AMQP	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	51.103.5.1	55602	443	158	0	104	65.82278	10.6469	16184.33	b'170303(	08/11/2020	17:20:16	
20	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	218.92.0.2	172.16.10.	53162	22	78	0	24	30.76923	5.256065	16913.35	b'000000(	08/11/2020	17:32:25	
21	AMQP	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	51.103.5.1	55602	443	159	0	105	66.03774	10.71429	22484.43	b'170303(	08/11/2020	19:05:16	
22	AMQP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	113.85.42	172.16.10.	49291	22	78	0	24	30.76923	5.256065	26708.35	b'000000(	08/11/2020	20:15:40	

ampq_information

In the above screenshot we see that each row describes each packet and the rows describe the extracted characteristics. For example, some information about the first amqp packet is that it runs over TCP, uses the Ethernet, its source port is 443 and its destination port is 54493, its size is 100 bytes, it is not encrypted, its payload is 46 bytes and it was delivered at 12:50:33.

- My algorithm captured 11 packets, that they use the COAP protocol.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	app_protc	transport_	layer	mac_src	mac_dst	src_ip	dst_ip	src_port	dst_port	pkt_size	is_encrypt	payload si	payload_r	previous_	packet_tin	payload	p_date	p_time
2	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	167.248.1	172.16.10.	8545	5683	60	0	0	0	4.043127	65885.74	b'0000'	09/11/2020	07:08:38
3	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	167.248.1	172.16.10.	8545	5683	60	0	0	0	4.043127	65885.88	b'0000000	09/11/2020	07:08:38
4	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	167.248.1	172.16.10.	41034	5683	60	0	0	0	4.043127	65886.05	b'0000000	09/11/2020	07:08:38
5	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	167.248.1	172.16.10.	41034	5683	62	0	8	12.90323	4.177898	65886.05	b'06e1636	09/11/2020	07:08:38
6	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	167.248.1	172.16.10.	41034	5683	60	0	0	0	4.043127	65886.2	b'0000000	09/11/2020	07:08:38
7	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	162.142.1	172.16.10.	19575	5683	60	0	0	0	4.043127	70487.33	b'0000'	09/11/2020	08:25:19
8	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	162.142.1	172.16.10.	19575	5683	60	0	0	0	4.043127	70487.46	b'0000000	09/11/2020	08:25:19
9	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	162.142.1	172.16.10.	53986	5683	60	0	0	0	4.043127	70487.69	b'0000000	09/11/2020	08:25:20
10	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	162.142.1	172.16.10.	53986	5683	62	0	8	12.90323	4.177898	70487.69	b'06e1636	09/11/2020	08:25:20
11	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	162.142.1	172.16.10.	53986	5683	60	0	0	0	4.043127	70487.82	b'0000000	09/11/2020	08:25:20
12	COAP	TCP	Ethernet	30:30:3a:3	37:30:3a:3	67.27.165.	172.16.10.	80	64590	1514	0	1460	96.43329	102.0216	80272.51	b'3178373	09/11/2020	11:08:24
13																		

As before, in the above screenshot we see that each row describes each packet and the rows describe the extracted characteristics. For example, some information about the second coap packet is that it runs over TCP, it runs over Ethernet, its source Ip address is 167.248.133.27, its destination port is 5683 and it was delivered on 09/11/2020.

- My algorithm captured 661 packets, that they use the MQTT protocol.

app_protocol	transport_	layer	mac_src	mac_dst	src_ip	dst_ip	src_port	dst_port	pkt_size	is_encrypt	payload si	payload_r	previous_	packet_tin	payload	p_date	p_time
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	216.58.19	172.16.10.	443	54484	1484	0	1430	96.36119	1	0	b'b0182ec	08/11/2020	12:50:32
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	216.58.19	172.16.10.	443	54484	1484	0	1430	96.36119	100	0.001134	b'a29cf8c5	08/11/2020	12:50:32
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	178.73.21	172.16.10.	48178	1883	60	0	0	0	4.043127	958.6125	b'0000'	08/11/2020	13:06:31
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	178.73.21	172.16.10.	48178	1883	60	0	0	0	4.043127	958.6797	b'0000000	08/11/2020	13:06:31
MQTT	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	216.58.20	59473	443	165	0	111	67.27273	11.1186	1434.232	b'10b9727	08/11/2020	13:14:26
MQTT	TCP	Ethernet	37:30:3a:3	30:30:3a:3	172.16.10.	216.58.20	62903	443	1419	0	1365	96.1945	95.61995	2432.226	b'b0a04d8	08/11/2020	13:31:04
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	20.54.110.	172.16.10.	443	64337	1514	0	1460	96.43329	102.0216	2847.251	b'c0ada87	08/11/2020	13:37:59
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	20.188.78.	172.16.10.	443	55316	1514	0	1460	96.43329	102.0216	4979.555	b'b000e05	08/11/2020	14:13:32
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	93.184.22	172.16.10.	80	55345	288	0	234	81.25	19.40701	4980.103	b'a2b44f9	08/11/2020	14:13:32
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	20.188.78.	172.16.10.	443	55316	445	0	391	87.86517	29.98652	4980.841	b'10021f2	08/11/2020	14:13:33
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	20.188.78.	172.16.10.	443	55316	1514	0	1460	96.43329	102.0216	4981.882	b'a287ba7	08/11/2020	14:13:34
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	20.188.78.	172.16.10.	443	55316	1514	0	1460	96.43329	102.0216	4983.022	b'b0e3cf5	08/11/2020	14:13:35
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	104.103.9	172.16.10.	443	55385	275	0	221	80.36364	18.531	4989.388	b'e0cb935	08/11/2020	14:13:41
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	194.177.2	172.16.10.	80	55395	1514	0	1460	96.43329	102.0216	4989.883	b'1000007	08/11/2020	14:13:42
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	194.177.2	172.16.10.	80	55395	1514	0	1460	96.43329	102.0216	4989.883	b'd0ff159	08/11/2020	14:13:42
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	194.177.2	172.16.10.	80	55395	1514	0	1460	96.43329	102.0216	4989.902	b'b006010	08/11/2020	14:13:42
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	194.177.2	172.16.10.	80	55395	1514	0	1460	96.43329	102.0216	4989.902	b'90c2020	08/11/2020	14:13:42
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	194.177.2	172.16.10.	80	55395	1514	0	1460	96.43329	102.0216	4989.907	b'c01e344	08/11/2020	14:13:42
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	51.104.13	172.16.10.	443	56349	904	0	850	94.02655	60.91644	5264.979	b'900c8b3	08/11/2020	14:18:17
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	51.104.13	172.16.10.	443	56470	89	0	35	39.32584	5.997305	5297.998	b'd0b92a6	08/11/2020	14:18:50
MQTT	TCP	Ethernet	30:30:3a:3	37:30:3a:3	204.79.19	172.16.10.	443	58299	94	0	40	42.55319	6.334232	5804.616	b'c0e89ad	08/11/2020	14:27:17

mqtt_information

Like the previous screenshots we see that each row describes each packet and the rows describe the extracted characteristics. For example, some information about the third mqtt packet is that it runs over

TCP, it runs over Ethernet, its destination Ip address is 172.16.10.141, its destination Mac is 37:30:3a:32:30:3a:38:34:3a:30:65:3a:32:61:3a:34:37 and it is not encrypted.

References

[1] L. Tagliaferri, "An Introduction to Machine Learning," 2017. [Online]. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning#:~:text=Machine%20learning%20is%20a%20subfield%20of%20artificial%20intelligence. [Accessed Wednesday October 2020].

[2] "Machine Learning," [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/machine-learning . [Accessed Wednesday October 2020].

[3] J. Brownlee, "Supervised and Unsupervised Machine Learning," 2016. [Online]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/ . [Accessed Wednesday Oktober 2020].

[4] GeeksforGeeks, "ML | Semi-Supervised Learning," [Online]. Available: https://www.geeksforgeeks.org/ml-semi-supervised-learning/. [Accessed Wednesday October 2020].

[5] " What is reinforcement learning? The complete guide," 2018. [Online]. Available: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/ . [Accessed Wednesday October 2020].

[6] "AI, Self-Learning & Deep Learning Technologies," [Online]. Available: https://orbograph.com/ai-self-learning-and-deep-learning/ . [Accessed Wednesday October 2020].

[7] Y.-J. Zhang, "Machine Learning for Image Classification," 2015. [Online]. Available: https://www.igi-global.com/chapter/machine-learning-for-image-classification/112330 .

[8] V. Flovik, "How to use machine learning for anomaly detection and condition monitoring," 2018. [Online]. Available: https://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection-and-condition-monitoring-6742f82900d7. [Accessed Wednesday Otober 2020].

[9] J. Kober, "Robot learning," [Online]. Available: https://link.springer.com/referenceworkentry/10.1007%2F978-1-4471-5102-9_100027-1. [Accessed Thursday October 2020].

[10] A. R. a. t. A. A. A. Tutorial., 2016. [Online]. Available: https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html . [Accessed Thursday October 2020].

[11] J. Brownlee, "Difference Between Algorithm and Model in Machine Learning.," 2020. [Online]. Available: https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/ . [Accessed Thursday October 2020].

[12] "All machine learning Models Explained in 6 minutes.," 2020. [Online]. Available: https://towardsdatascience.com/all-machine-learning-models-explained-in-6-minutes-9fe30ff6776a . [Accessed Thursday October 2020].

[13] R. Gandhi, "Support Vector Machine- Introduction to Machine Learning Algorithms," 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 . [Accessed Tuesday October 2020].

[14] "Regression Analysis Essentials For Machine Learning," [Online]. Available: http://www.sthda.com/english/wiki/regression-analysis-essentials-for-machine-learning .

[15] U. o. Bergen, "Bayesian networks," [Online]. Available: https://www.uib.no/en/rg/ml/119695/bayesian-networks .

[16] J. Shapiro, " Genetic Algorithms in Machine Learning," 2001. [Online]. Available: https://link.springer.com/chapter/10.1007%2F3-540-44673-7_7 . [Accessed Tuesday October 2020].

[17] T. Shah, " About Train, validation and test sets in Machine Learning.," 2017. [Online]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7. [Accessed Tuesday October 2020].

[18] J. Brownlee, "Overfitting and underfitting with machine learning algorithms.," 2016. [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ . [Accessed Tuesday October 2020].

[19] "INTERNET OF THINGS," Ansystem, March 2020. [Online]. Available: https://www.avsystem.com/blog/iot-protocols-and-standards/. [Accessed Wednesday November 2020].

[20] "AMQP Advanced Message Queing Protocol," [Online]. Available: https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf. [Accessed November 2020].

[21] "MQTT Control Packet format," Solace Corporation, November 2020. [Online]. Available: https://docs.solace.com/MQTT-311-Prtl-Conformance-Spec/MQTT%20Control%20Packet%20format.htm.

[22] "Python. Print mac address out of 6 byte string," [Online]. Available: https://stackoverflow.com/questions/4959741/python-print-mac-address-out-of-6-byte-string.

[23] [Online]. Available: 17) Vegard Flovik (2018). How to use machinehttps://towardsdatascience.com/how-to-use-machine-learning-for-anomaly-detection-and-condition-monitoring-6742f82900d7 .