Karlsruhe Institute of Technology

Detection of Port Scans with Convolutional Neural Networks based on Aggregated Traffic

Master Thesis
of

Robin Miller

at the Institute of Telematics
on the Faculty of Informatics

First Reviewer:	Prof. Dr. M. Zitterbart
Second Reviewer:	Prof. Dr. H. Hartenstein
Supervisor:	Samuel Kopmann, M.Sc.

01.05.2022 – 31.10.2022

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe.

Karlsruhe, den 31.10.2022

Contents

1 Introduction **1**
- 1.1 Objectives . . . 2
- 1.2 Structure of this Thesis . . . 3

2 Background **5**
- 2.1 TCAM and High Bandwidth Networks . . . 5
- 2.2 Port Scan Types . . . 6
  - 2.2.1 TCP SYN Stealth Port Scan . . . 7
  - 2.2.2 TCP SYN Connect Port Scan . . . 7
  - 2.2.3 Specific TCP Port Scans . . . 8
  - 2.2.4 Further Port Scan Types . . . 8
- 2.3 Z-Score Normalization . . . 9
- 2.4 Machine Learning . . . 9
- 2.5 Machine Learning Approaches . . . 10
- 2.6 Metrics . . . 14
  - 2.6.1 Confusion Matrix . . . 14
    - 2.6.1.1 True Positive (TP) . . . 14
    - 2.6.1.2 True Negative (TN) . . . 14
    - 2.6.1.3 False Negative (FN) . . . 14
    - 2.6.1.4 False Positive (FP) . . . 15
  - 2.6.2 False Negative Rate (FNR) . . . 15
  - 2.6.3 False Positive Rate (FPR) . . . 15
  - 2.6.4 Conclusion on Confusion Matrix . . . 15
  - 2.6.5 Accuracy . . . 15
  - 2.6.6 Precision . . . 16
  - 2.6.7 Recall . . . 16

3 Analysis **17**
- 3.1 Problems with Port Scans . . . 17
- 3.2 Port Scan Attack Attributes . . . 17
  - 3.2.1 Probing Amount . . . 18
  - 3.2.2 Port Scan Timing . . . 18
  - 3.2.3 Destination Port Order . . . 18
  - 3.2.4 Attacker Node Source Ports . . . 18
  - 3.2.5 Port Scan Packet Header Fields . . . 18
  - 3.2.6 IP Address Hiding . . . 19
  - 3.2.7 Intrusiveness and Packet Amount . . . 19
  - 3.2.8 Port Range . . . 19
  - 3.2.9 Zombies . . . 19

3.3 General Port Scan Behavior . . . 20
3.4 Assumptions and Requirements . . . 20
3.4.1 Assumptions about Packet Traces . . . 20
3.4.2 Assumptions about Port Scan Types . . . 20
3.4.3 Assumptions Port Scan Attributes . . . 20
3.4.4 Assumptions of Network Structure . . . 20
3.5 Requirements . . . 21
3.5.1 Requirement Reaction Time . . . 21
3.5.2 Requirement High Recall . . . 21
3.5.3 Requirement Low False Positive Rate . . . 21
3.5.4 Requirement Small Port Range Attack Detection . . . 21
3.5.5 Requirement Computational Power . . . 21
3.6 Limits and Constraints . . . 21
3.6.1 Limit Timed Stealth Scan . . . 21
3.6.2 Constraint Host Discovery . . . 22
3.6.3 Machine Learning . . . 22
3.7 Summary . . . 22
3.8 Related Work . . . 23
3.8.1 Machine Learning Micro-Flow Port Scan Detection . . . 23
3.8.2 Signature-based Port Scan Detection . . . 24
3.8.3 Honeypot Ports and Systems . . . 24
3.8.4 Spice detection . . . 25
3.8.5 Threshold-based Detection . . . 25
3.8.6 Visualization Techniques of Port Scans . . . 25
3.9 Summary . . . 26

4 Design **27**
4.1 Aggregation Scheme Overview . . . 27
4.2 Aggregation Scheme . . . 28
4.3 TCAM and Aggregation . . . 28
4.4 Packet Attribute Extraction . . . 28
4.5 Aggregation Maps . . . 29
4.5.1 Dimensions of the Aggregation Map . . . 29
4.5.2 Buckets and Resolution . . . 30
4.5.3 Shape of the Aggregation Map . . . 31
4.5.4 Building an Aggregation Map . . . 32
4.6 Visualization of Aggregation Maps and Port Scans . . . 33
4.6.1 Aggregation Map Exposure Time . . . 34
4.6.2 Threshold and Labeling . . . 36
4.6.3 Normalization . . . 36
4.7 Aggregation Map Port Scan Properties . . . 37
4.8 TCAM Rules for Aggregation Maps . . . 37
4.9 Machine Learning Input with CNN . . . 39
4.10 Balancing Machine Learning Input . . . 39
4.11 Machine Learning Model (CNN) . . . 39
4.12 CNN Classification . . . 40
4.13 Optimizing Aggregation Map Parameters . . . 41
4.13.1 Exposure Time . . . 41
4.13.2 Resolution . . . 42

4.13.3 Threshold . . . 43
4.14 Timed Stealth Port Scans . . . 43
4.15 Summary . . . 44

5 Implementation **45**
5.1 TCAM in Experiment Setup . . . 45
5.2 Aggregation and Machine Learning Setup . . . 45
5.2.1 Dataset with Network Traffic . . . 45
5.2.2 Synthetic Attacks . . . 46
5.2.3 Port Scan Attack Injection . . . 46
5.2.4 Port Scan Attack Packet Labeling . . . 46
5.2.5 Aggregation Map . . . 48
5.2.6 Aggregation Map Labeling . . . 49
5.2.7 Aggregation Map Balancing . . . 50
5.2.8 Normalization . . . 50
5.2.9 CNN Framework Keras . . . 50
5.2.10 CNN Model . . . 50
5.2.11 Keras Classification . . . 51
5.3 Experiments . . . 52
5.3.1 Implemented Assumptions . . . 52
5.3.2 Parameter Optimization . . . 52
5.3.3 Exposure Time Optimization . . . 53
5.3.4 Resolution Optimization . . . 53
5.3.5 Threshold Optimization . . . 53
5.3.6 1000 Ports Port Scans . . . 54
5.3.7 Stealth Port Scans . . . 54
5.3.8 CIC Port Scans Balanced vs Unbalanced . . . 54
5.3.9 Implementation of Metrics . . . 56

6 Evaluation **57**
6.1 Parameter Optimization Experiments . . . 57
6.2 Experiment Exposure Time . . . 57
6.3 Experiment Resolution . . . 61
6.4 Experiment Threshold . . . 62
6.5 Experiment Complete Port Range Attacks . . . 63
6.6 Complete Port Range Attacks Training Performance . . . 64
6.7 Experiment 1000 Ports Port Scan Attacks . . . 65
6.8 Experiment Three Ports Port Scans . . . 67
6.9 Experiment CIC Port Scans Balanced vs Unbalanced . . . 69
6.10 Evaluating Requirements . . . 72
6.11 Summary . . . 74

7 Conclusion **77**
7.1 Future Work . . . 78

Bibliography **81**

1. Introduction

Port Scanning remains a threat in network security. The entrance door for many attacks is opened with a port scan. A port scan is done by sending single packets to a server on different ports. The ports that reply are expected to be open. Applying this method to a range of ports, the attacker learns which ports are open. It is also possible for the attacker to identify the services that are installed on the open ports. There are different port scan attacks. Since *nmap* [16] got introduced the most common scan is an un-intrusive SYN port scan. This thesis focuses on detecting this type of port scan. Other port scan types are discussed in the analysis chapter 3. The key problem is that developers rely on dependencies. These dependencies define how the servers and the network are set up. Over time dependencies get updated, to introduce new behavior or remove security issues. In recent times networks consist of systems with a high amount of dependencies. The inter-dependencies can result in vulnerabilities when the systems are updated. It is observable from websites like https://cve.mitre.org/ that every day new exploits are found. The approach of the cyber security community is to open-source exploit code, so that developers are able to test their own systems with the exploit code. With this approach an exploit is also verified. There is a vast amount of written exploits that are applicable just by knowing on which port a service is running. An attack scenario is simply executed, e. g., *metasploit*. In *metasploit* it is only necessary to configure IP address and port [10]. The rest of the exploit is automated. The exploits that *metasploit* is using can easily result in a complete takeover of the victim's system. In conclusion this results in easily executable attacks when a network node and port structure is known. These network structures are identified by network and port scans. As a countermeasure detection systems for port scans have been developed. Machine Learning (ML) rises in popularity. Approaches that observe micro-flow tuples with ML are commonly used. These tuples contain IP source, IP destination, ports, duration and potentially more features. These related ML approaches with supervised ML perform port scan detection on completed flows. This thesis adapts a novel DDoS detection approach [11] to determine if a port scan was executed within a monitored network. In contrast to known approaches, traffic is highly aggregated prior to classification. The aggregation scheme only relies on source IP address and destination port. One objective is fast detection of port scans. Ternary Content Addressable Memory (TCAM) is an option to monitor high bandwidth networks [27]. TCAM requires one clock cycle to match for

example a range of IPs. TCAM has a high energy consumption, therefore the proposed solution in this thesis requires only a limited amount of TCAM rules. The aggregation approach in this thesis needs to be optimized regarding parameters of the aggregation scheme. The resulting objective in this thesis is to identify optimal parameters and research if hidden port scans can be detected. With optimized parameters the aggregation scheme is expected to have a high success rate in detecting port scans, while also having short computation time. This approach promises the possibility to act before a follow-up attack on the services in the scanned network is executed.

1.1 Objectives

The overall goal of this thesis is to determine if port scans are detectable within a network with an aggregation scheme in conjunction with a machine learning model. This section introduces the optimization tasks to classify port scans in the most efficient way. The most efficient way includes the port scan detection possibilities when the attacker is trying to hide the scan activity.

Synthetic Data and Attack Injection

To test the aggregation scheme, datasets are necessary. The dataset needs to resemble real attack network traffic as closely as possible. Therefore the objective is to create synthetic datasets. The synthetic data is necessary to design different attack scenarios e.g. hidden port scans and straight forward un-intrusive port scans. The synthetic nature of the traffic has to allow for port scan attack injection at specified timestamps and further set the amount of scanned ports.

Detecting Port Scans

The next objective derived from the main objective is to detect a port scan using the synthesized data. With the synthetic attack injection stealth port scans are created. This objective focuses on the correct classification of aggregated traffic. This means that a given machine learning model is able to detect a port scan by classifying the aggregated traffic correctly.

Optimizing Detection

The next tasks are to determine optimal parameters for detecting the straightforward non-stealth port scan.
The aggregation scheme relies on the process of strongly aggregated traffic. Different parameters are researched that need to be optimized. It is measured how long traffic is aggregated. The duration of aggregated traffic determines how many time sequences of traffic are aggregated. It needs to be investigated how long the optimal time sequence of traffic is. Next it is investigated how many TCAM rules are necessary. This is done by measuring different granularities of how detailed aggregated traffic needs to be created. Identification of thresholds within the aggregated traffic is the next objective. These thresholds describe, if the aggregated traffic contained in one time sequence of traffic contains a port scan. These parameters are further discussed in the design chapter 4.

Detection of Hidden Port Scans

The next objective after establishing a consistent port scan detection is to determine if port scans are still detectable when the attacker is trying to hide the port scan behavior. The first stealth attack scenario focuses on the limit when the attacker scans small port ranges with longer time durations between each port scan. This results in smaller port scans, than the previous non-stealth port scan that scans the entire possible port range. The objective is to determine at which time durations of aggregation the port scan attacks are undetectable. An insight discussion on this topic is given in the design chapter 4.

1.2 Structure of this Thesis

The objectives discussed in the section above have to be solved. Therefore this section describes the measures taken.

The next chapter background (2) gives an overview of the necessary information and tools to understand the analysis and concept. In the background chapter it is discussed what a port scan is and which types are observed in this thesis. The machine learning approach is explained in detail. Backgrounds includes a section on the different metrics used to evaluate the results from the aggregation scheme. TCAM is explained as it is necessary to use in networks with high bandwidths.

The next chapter analysis is to describe the problem of detecting port scans. In this section requirements are determined. The limits of this research is highlighted in this section. Different attack port scan scenarios that require different detection responses are discussed. In the analysis chapter a comparison between the detection approach in this thesis and related work is done.

The design chapter describes the measures taken to identify the port scans described in the analysis chapter. In this chapter the details are shown on how the process of the aggregation scheme works. Further the important parameters for the aggregation scheme are discussed. The setup for the machine learning approach is explained in detail.

The implementation chapter is provided to ensure reproducibility. Here are implementation details, of which frameworks are used. Also this chapter describes how aggregation maps are created with a technical view. In the implementation the setup of the different experiments is described.

The evaluation chapter is provided to show the main results of this thesis in detail. In this chapter the performance metrics that are created with the evaluation scores from the backgrounds section are shown. In this chapter the introduced experiments from the introduction are evaluated. This contains the optimal parameters determined for the aggregation and the overall classification performance with a known dataset (CICIDS2017).

2. Background

The chapter provides background knowledge about technical aspects used in this thesis. First, TCAM is functionally explained. Secondly, to understand how a port scan is executed a port scan is defined and a selection of different port scan types are described. Normalization is explained since it is necessary for the aggregation used in the upcoming chapters. To evaluate the aggregation scheme performance different performance metrics for machine learning classification are explained.

2.1 TCAM and High Bandwidth Networks

Ternary Content Addressable Memory (TCAM) is a form of physical memory. TCAM is different from Random Access Memory (RAM) memory in the sense that a value with three states (ternary) is matched to a RAM address. These three states are one, zero and "don't care ". TCAM matches on the first entry in its rule table. Therefore, the order of rules is important. TCAM that uses rules in ascending order are assumed. In this thesis TCAM is used to match header fields of a TCP layer and IP header field. These values are extracted bit wise. A short example is given for a port range in Figure 2.1. In this example it is assumed that the TCP packet header has a port which is given as 22 in decimal. A TCAM rule is necessary to match this value to a RAM address. 22 is 10110 in binary. The port header field has 16 bit values therefore in this example 0's are used to fill up the preceding bit fields. The TCAM rule then matches on 0000 0000 0001 0110 and returns a RAM value. To advance this rule it is possible to use "don't care "states. For example if a range of ports is needed it is possible to use 0000 001X XXXX XXXX. This matches everything that is between 1023 and 512. This form of bucket matching is important for the aggregation scheme and is further explained in the design 4 and implementation 5 chapters. A more technical overview of TCAM is given in [**?**] concerning next generation network devices.

TCAM Match Field	RAM Output Value
0000 0000 0001 0110	“&0x7fffffffeb58” RAM Address of Aggregation Map for Port 22
0000 001X XXXX XXXX	“&0x7fffffffeb62” RAM Address for Port Range
…	…

Figure 2.1: Example TCAM Table

TCAM is very expensive and has a high energy consumption [17]. However, TCAM has its application in SDN networks [25]. SDN networks contain switches which are already using TCAM for multiple header field classification [11]. TCAM can be used with high bandwidth networks [7], when trying to identify malicious traffic. In this thesis TCAM is used because it is fast. A match for a given sequence of zero, one and “don’t care “is matched in one processor clock cycle.

2.2 Port Scan Types

A port scan in this thesis is defined as an attacker node scanning a victim node within a network. The process of scanning is probing the victim node for their open ports with network packets. A structural overview is shown in Figure 2.2. The packet is sent with a protocol for example Transmission Control Protocol (TCP). An example of port scan output is shown in Figure 2.3. This port scan is done on the router within a local network. It shows the web interface on port 80 as well as on port 443. Also port 53 is open with a domain service running.

Port scans have different goals depending on the purpose of the scan. A possible scenario is that the port scan is used to identify a service and its version behind a port. One of the goals of a port scan in the scenario of this thesis is to be undetected. The protocols and packet headers differ depending on the port scan types. After defining port scan behavior it is needed to discuss the different port scan types. The port scan types differ in their visibility, intrusiveness and execution duration.

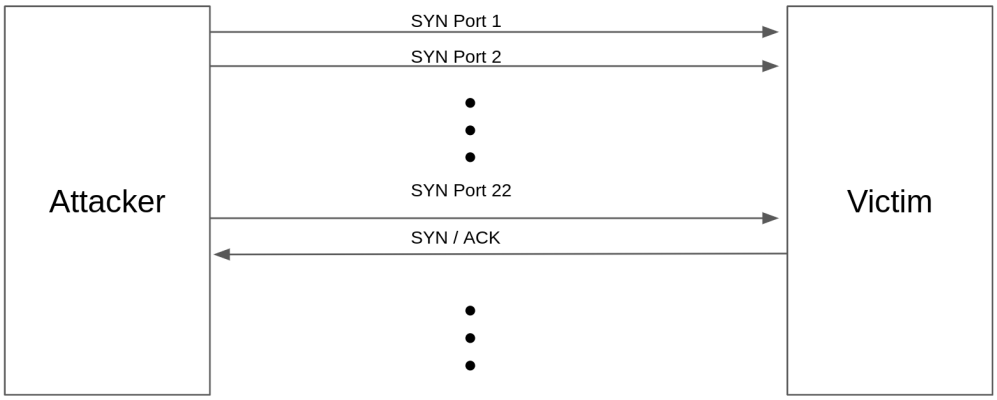


Figure 2.2: Example Port Scan Structure

```
~$ nmap 192.168.0.1
Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-07 08:25 CEST
Nmap scan report for 192.168.0.1
Host is up (0.057s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https
```

Figure 2.3: Example Port Scan Output

2.2.1 TCP SYN Stealth Port Scan

The first port scan presented is the SYN flag port scan. First a TCP packet is sent with the SYN Flag to the requested ports. There are two possible responses to this SYN packet. One is the SYN/ACK response which tells the attacker node that the port is open. The second response is RST (reset) packet. This gives the attacker node the information that the port is closed. An important characteristic of this port scan is that the attacker sends a RST packet in case of a SYN/ACK response from the system. This terminates the connection.

2.2.2 TCP SYN Connect Port Scan

The termination of the connection from the section before is the main difference in the connect port scan. In the scenario that the port is open, the connect port scan type does not instantly terminate the connection. The connection is replied with ACK. This helps

in identifying the service behind the port. The victim node port service will reply with data, to establish the connection. This data possibly includes the service. For example port 22 could send data containing the string “openssh “with the negotiation credentials. This tells the attacker node that port 22 contains the service secure shell protocol (SSH). Also since it is necessary for connection the version of the service can be transmitted. This port scan is more visible than the SYN Stealth Scan since there is double the amount of packets sent to the victim’s open ports. Also it is more intrusive since it detects more information than just a port being open or not.

2.2.3 Specific TCP Port Scans

This section is to give a short introduction to edge case TCP port scans. An introduction to *nmap* can be found in the paper [16]. As mentioned in the *nmap* documentation there are FIN, NULL and Xmas port scans. These port scans target specific requests for comments (RFC) weaknesses of TCP. A weakness for example can be that if there is no SYN Flag in the header of the packet a TCP port will respond with the RST flag if the port is closed. However if the port is open no response is sent. This information is used to determine port statuses which are not identified with the previous port scan types.

2.2.4 Further Port Scan Types

The previous sections explained different TCP Port Scans. There are more port scan types. Some of these port scans are highlighted in this section, however the main focus in this thesis lies on detecting TCP SYN ACK scans.

Another port scan method is to use different scanning header fields. For example a Null scan does not set the header field of the TCP packet for the flag. An open port does not reply to this TCP packet request, while a closed port will respond with a RST flag. This results in an inverse image of the services on a host, meaning that it is easy to identify which ports have services running. UDP port scans are necessary to figure out domain name system (DNS) hosts. UDP port scans require specific payload to determine if a port is open, since the UDP will not respond to empty packets, like the TCP versions provide.

Next there are TCP ACK port scans. The goal of these scans is to figure out firewall rules. An ACK scan will not determine if a port is open or closed, but will determine if the port is allowed to communicate.

The TCP window port scan exploits an error of the window size in the RST packet. If the port responds with a non zero window field the port is declared as open by the port scan. The scan at the same time determines firewall rules.

Another port scan technique is the idle port scan. In this scenario an IP is stolen from another host within the system. This ID then is used to hide the source of the attack.

The TCP FTP bounce scan is another method to hide the actual attacker. However it requires a vulnerable host within the network. Essentially FTP allows the attacker to send

requests from the vulnerable host. Therefore the original attacker node stays hidden.

The previous scans tried to hide the attacker. In this thesis it is researched if the aggregation scheme is able to classify such behavior correctly.

2.3 Z-Score Normalization

In this thesis z-score is used to normalize the samples of the dataset which includes synthetic port scan types. The normalization is done specifically on aggregation maps which are introduced in the design section 4. Z-score is applied to each data point in a dataset. The z-score expresses the deviation of the average of all values in the dataset [4].

$$\text{Normalizsed Data Point} = (\text{Datapoint} - \mu)/\sigma \tag{2.1}$$

The normalized data point in 2.1 is used on each data point in the dataset. The data point resembles the original value in the dataset. μ is the mean of the dataset. Finally σ denotes the standard deviation. The result of applying this formula to each value in the dataset is the normalized representation.

2.4 Machine Learning

The port scans need to be identified while being surrounded by background traffic. Therefore, in this thesis an machine learning approach is used with mechanics that are explained in this section. Machine Learning (ML) in the context of this thesis is training a model till weights are set and further data points can be requested and are classified by the model. A machine learning model gets traversed in two ways. First the forward pass is done by going through each layer and calculating the output. When labeled training data is available, the weights are recalculated with backward propagation. There are always two phases, the training phase and the testing phase. The ML model learns by experiencing more training and with more training improves detection of port scans. The identification is called classification and is done in the testing phase.

The model constraints the input dimensions, the behavior and the output. The output is the prediction the machine learning approach has for a given input. In context of this thesis this means that the machine learning model classifies in either port scan or benign traffic. Training with machine learning (ML) means using input data that is disjoint from testing data. The data can be provided labeled or unlabeled. If the training is done supervised, each data point that is fed into the machine learning model has a label. The label declares if the data point is benign or port scan. In the aggregation scheme approach supervised learning is used. Unsupervised learning would be done with data points that are not labeled.
Testing includes data points which are not part of the training and let the model classify these data points. In the context of this thesis data points represent aggregated traffic. These classifications are evaluated with the metrics mentioned in the upcoming section 2.6. The focus of this thesis is to detect port scans. Therefore, binary classification is necessary. This means the model will only output either the aggregated data point is a port scan or it is benign. With different models multi-class classification is possible. This means that for example another attack different from port scan can also be detected.

2.5 Machine Learning Approaches

The section above established that there are different ML approaches that differ in the sense of being supervised and unsupervised. In this thesis the focus is on convolutional neural networks (CNN). This model architecture is used because of its application in image classification [8]. CNNs are a form of deep neural networks. More specifically they are derived from multilayer perceptions (MLPs). MLPs for classification contain at minimum an input, output and hidden layer. These layers are iterated over in sequential order. The data points are processed by the input layer. Then a forward pass is applied. This means that all neurons within the layers are passed through. A CNN has a definable amount of hidden layers. CNN's contain an input and output layer. The layer types concerned for this thesis research are convolutional, pooling layers, filters, flatten and dropout. Also activation layers are used.

Layers

The upcoming paragraphs give a short overview of the Convolutional Neural Network (CNN) Layers. The first layer of a CNN is named an input layer. The input layer can be described as a feature map. A CNN can use a convolutional layer as its first layer. The input layer has to accept the shape of the input. The input shape in most frameworks is given as a connected array which contains all images. The format of the shape is given as: (number of inputs) × (input height) × (input width) × (input channels). In this thesis the input channel is represented as one, since there's no color specific detection. CNNs are designed for image recognition, their input layer is expected to have a two dimensional shape. The layers are connected in a sequential order in an CNN. This means that each output from a layer is introduced to the next layer. Therefore, each inner layer has a receptive field for the input that is translated to the layer output layer.

Convolutional Layer

The convolutional layer can be used as the first layer. This means that the input shape is evaluated with a learnable kernel. The kernel has a predefined size. Since this thesis focuses on image classification a two dimensional input and a two dimensional kernel are used. In Figure 2.4 the process is displayed.

The kernel and move over each fitting two dimensional shape of the input shape. The stepping size is called stride and can be done horizontally and or vertically. In each stride the kernel is used to calculate a new value to the input shape. If the stride step does not fit the input shape zero padding can be used to adapt the input shape. The process of the convolutional layer scales down the complexity of the input shape [20]. The stride and zero padding are called hyper parameters.

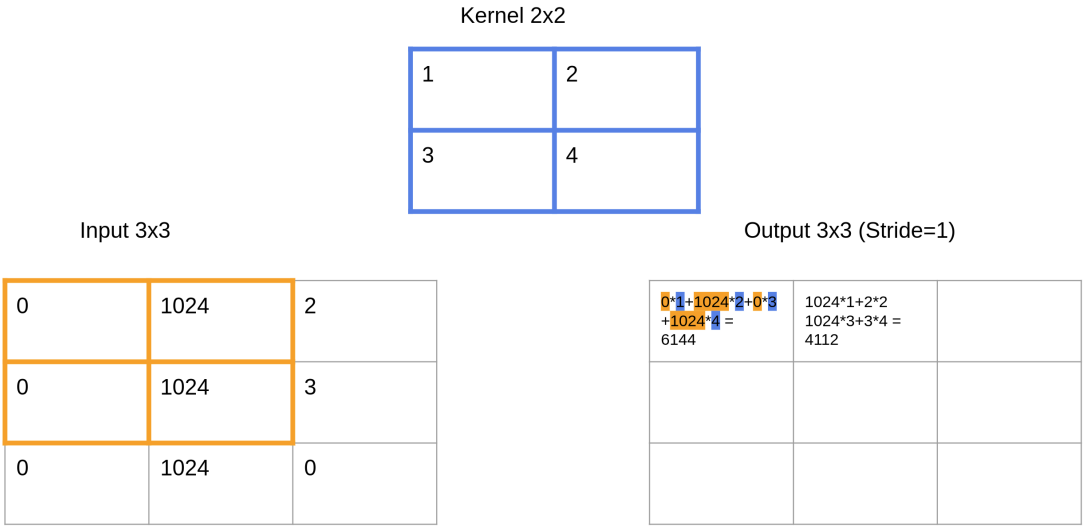


Figure 2.4: Convolutional Layer Processing

Max Pooling Layer

A max pooling layer reduces the dimensions of the shape it gets fed from the previous layer. A pooling layer uses a filter size to define how many values are evaluated for max. This means if the filter size is (2,2) that the max value within these four values is taken and used for the new reduced dimensional shape. Essentially reducing four fields into one. The pooling layer also steps through the given input shape with a stride. Depending on the stride and filter size the two dimensional shape is reduced. This can be viewed example wise in Figure 2.5 the input shape does not fit the stride and pooling size zero padding needs to be applied.

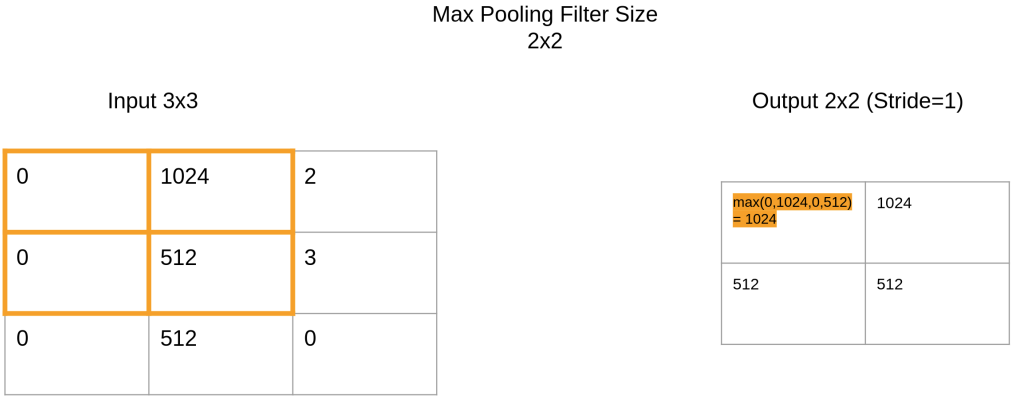


Figure 2.5: Max Pooling Layer Processing

Activation Layer

Whenever a convolutional or pooling layer is used an activation function is needed within the layer. This means that the output from the previous layer is adapted with a function (activation function) for every neuron. In this thesis rectified linear unit (relu) and sigmoid activation are used. In case of relu this is the formulae $f(x) = max(x, 0)$. This means that all negative values are removed from the input shape. The sigmoid activation works similar to the relu activation with the only difference being the formula applied to the input. The formulae for sigmoid is: $sigmoid(x) = 1/(1 + exp(-x))$.

Flatten Layer

The flatten layer is used to transform a feature map shape into a vector. E. g. a 32x32 matrix is transformed into a vector with the length of 1024. This is shown in Figure 2.6.

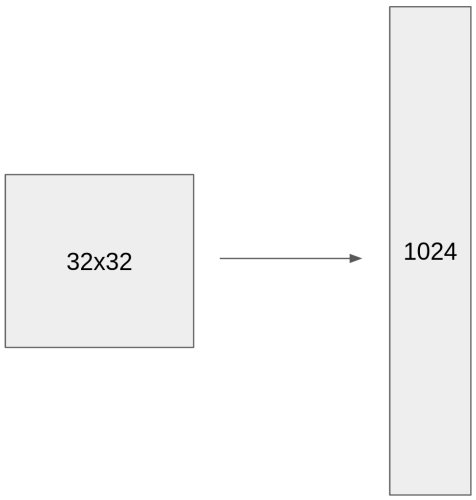


Figure 2.6: Flatten Layer Processing

Dense Layer

A dense layer is a fully connected layer. This means that each neuron from the previous layer is connected to the next layer. This is realized with the function 2.2. The bias resembles an optimization value of the machine learning model. The dense layer used in this thesis uses a predefined output space value. Meaning that if the dense layer is used with output value 64, the output format of the dense layer is 64. The kernel is adjusted accordingly to accommodate this.

$$\text{output} = \text{dot}(\text{input}, \text{kernel}) + \text{bias} \tag{2.2}$$

The application of the formula of a matrix can be displayed as shown in Figure 2.7.

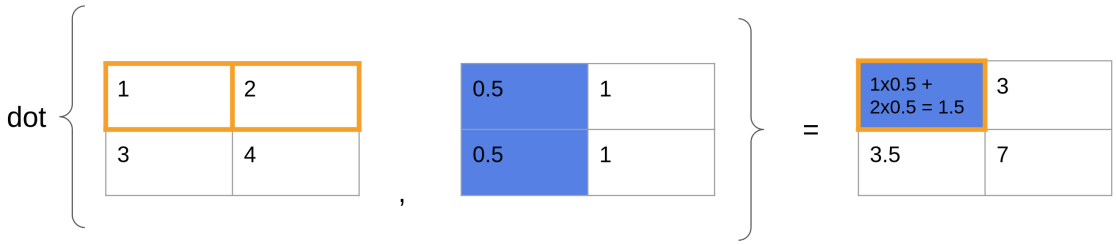


Figure 2.7: Dense Layer Processing

Dropout Layer

A dropout layer is used against overfitting. Overfitting means that the output determination from the previous layers has too many unnecessary connections that hinder classification. Dropout layers can be applied on fully connected layers to remove unnecessary connections between neurons. These unnecessary neuron connections are created by overfitting. Overfitting can occur with high amounts of learning data.
Dropout layers apply their reduction by randomly selecting input values and setting them to zero. A rate is predefined at which the inputs get reduced. The inputs that are not set to zero are scaled up, so the overall sum is not changed.
The Figure 2.8 shows an example where the value three is set to zero. The other values get scaled up with the formulae 2.3. The resulting scaling factor is 1.25 in this example. The Figure 2.8 shows an example where a fifth of all values gets removed and the other values get scaled up. The sum remains unchanged, both vectors sum up to 15.

$$\text{Scaling Factor} = \frac{1}{1 - \text{Rate}} \tag{2.3}$$

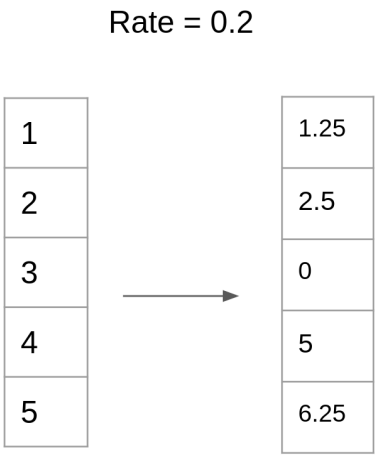


Figure 2.8: Dropout Layer Processing

2.6 Metrics

The port scan detection needs to be evaluated. Therefore, in this section metrics that define the quality of classification performance from the above mentioned machine learning approach is given.

2.6.1 Confusion Matrix

The confusion matrix has four different fields. Each value of the four fields identifies if a single data point got classified correctly. The different fields are calculated with the labels which are defined as positive (P) and negative (N). A table is shown in Figure 2.9 that describes the different values of the confusion matrix.

Confusion Matrix	Predicted Port Scan	Predicted Benign Traffic
Ground Truth Port Scan	Correctly Classified Port Scan (TP)	Falsely Classified Port Scan as Benign Traffic (FN)
Ground Truth Benign Traffic	Falsely Classified Benign Traffic as Port Scan (FP)	Correctly Classified Benign Traffic (TN)

Figure 2.9: Confusion Matrix with Port Scan Detection

2.6.1.1 True Positive (TP)

True Positive is the case when a data point is labeled port scan and the classification of the machine learning approach also declares the data point as port scan. This is a correct classification from the machine learning approach for a port scan. In a network this results in the fact that a port scan took place, and the machine learning approach classified the traffic as containing a port scan.

2.6.1.2 True Negative (TN)

The next metric is True Negative. A True Negative means that the label of the data point is declared as benign and the machine learning algorithm classifies the data point as benign. This is the second correct classification metric. The machine learning algorithm in this case classified benign traffic correctly.

2.6.1.3 False Negative (FN)

False Negative is the case if the machine learning approach classifies a data point wrongly as benign. The label is port scan, but the classification indicates it is benign. This is an error of the machine learning classification. In a network this means that a port scan was not classified as a port scan and therefore is un-detected.

2.6.1.4 False Positive (FP)

False Positive applies when the label is benign but the machine learning approach classifies the traffic as containing a port scan attack. This is an error of the machine learning approach. In the context of port scan classification this means the machine learning approach classified normal traffic as a port scan. This is especially problematic since normal traffic in this case can result in alerts within the network.

2.6.2 False Negative Rate (FNR)

The False Negative Rate sets the amount of False Negatives in relation to how many samples could have been classified positive. This means in the context of port scans that if the False Negative Rate is low the amount of undetected port scans is small in comparison to how many could have been detected. A value near zero is desired for this metric.

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{2.4}$$

2.6.3 False Positive Rate (FPR)

The False Positive Rate identifies the rate of false positives occurring compared to how many benign data points are in the dataset. The False Positive Rate is low when only a few normal background traffic samples are classified as port scan traffic compared to how many could have been falsely classified. This metric should have a value near zero.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.5}$$

2.6.4 Conclusion on Confusion Matrix

The four different fields of the confusion matrix can be used as in relative and absolute. To define the overall performance of the machine learning classifications. However, the fields of the matrix are used to define further metrics which are explained in the upcoming sections.

2.6.5 Accuracy

Accuracy is a metric that can be used to measure the quality of true positivity. True positives in this thesis are detected port scans. The formula is shown in 2.6. Technically $(P + N)$ is the amount of labeled data. Accuracy counts up how many times the classification was correct and gives a percentage to the entire labels. This means in context of port scans, that if the accuracy is high a high percentage of correct classification of benign traffic and port scan took place. A low value of accuracy means that either a high amount of port scans were undetected or a high amount of background traffic was falsely classified as port scan. Both cases can happen in conjunction.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}} \tag{2.6}$$

2.6.6 Precision

The precision uses fields from the confusion matrix. Formular 2.7 shows how many of the total amount of positive predictions that the machine learning model made, are correct. This means precision describes how many false positives got created while classifying port scans correctly. A high precision of near 1.0 means almost no false positives got created in comparison to how many port scan predictions have been made.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.7}$$

2.6.7 Recall

Recall is a metric given with the formulae 2.8. This metric divides the correctly classified labels by the amount of total labels that should have been classified as port scan. False Negative are data points which are labeled as port scan but the classification defines them as benign. The recall identifies the ratio of port scans that got identified compared to how many can be identified. Therefore the recall identifies how many port scans got detected, in comparison to how many could have been identified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.8}$$

3. Analysis

The analysis chapter is to explain in detail the complications and problems that are introduced by port scan attacks. It is discussed why port scanning is problematic. Different attributes are defined in succession. Also related work approaches to detect port scans are introduced.

3.1 Problems with Port Scans

In [18] it is denoted that 50 percent of network attacks are not executed after a port scan took place. However, this means that 50 percent of network attacks are either done blindly or a port scan is used. Port scans are essential to identify services running on different servers within the network. If an attacker has no information about the network or the hosts in the network the attacker needs to identify hosts and services. This is done by port scans. Deriving from this it is important to identify port scans as quickly as possible. A system administrator should be informed of the scanning behavior. System administrators and network hosts are victims of port scanning. Also a small fraction of victims are people who use their own router to surface their server on the internet. Honey pots have confirmed many port scans take place on wide ranges of IP addresses as described by Vasilomanolakis [24]. Vasilomanolakis used multiple honeypot servers over the globe to identify this scanning behavior. In conclusion the problem of openly cast port scans resides worldwide and many approaches have been taken to combat this.

3.2 Port Scan Attack Attributes

Sophisticated Port scans are designed to be stealth meaning that it is hard to detect that a port scan took place. The attacker is assumed to hide his scanning activities, because the attacker's IP can potentially be blocked by a firewall or a system administrator. There are different methods for hiding a port scan. A port scan can be described with different attributes. A selection of attributes is explained in the upcoming sections.

3.2.1 Probing Amount

In this thesis the first attribute that is observed is the amount of packets the attacking node sends to the victim system. The fewer packets which are sent, the harder it is to detect the port scan. However, the attacker needs to identify different services therefore it is not possible to identify all open ports with only a few packets. It is possible to hide scanning activities over a longer duration of time, which is explained in the next section.

3.2.2 Port Scan Timing

An attacker possibly is trying to hide his scanning behavior by not scanning a lot of ports at the same time. This means that there will be time gaps between scanning different ports. This significantly complicates the detection of a port scan. Patient attackers can wait days between scanning different ports. This is known as spice port scans in [23]. This will result in only a few packets that are port scan traffic per day. A system administrator is unlikely to identify this behavior with normal human observation. Also aggregated traffic approaches like presented in this thesis, have the problem of either observing small ranges, or long aggregation time windows. Longer aggregation windows can identify the hidden port scan (because all the necessary packets are in the aggregation window), however this would be problematic when it comes to the reaction time. Also the port scan is less visible, since the attacker likely did not scan a high amount of ports at the same time. This results in a delayed or not executed alert to the system admin.

3.2.3 Destination Port Order

The order of which ports are scanned is an attribute of port scans. The scanning tool provided by *nmap* for example does not scan ports 1 to 1000 in that order unless specifically specified. The port scan order is randomized. Depending on the port scan timings this can lead to confusion in detection. Since in some instances the random ports which are scanned over longer time durations do not happen in a predictable order.

3.2.4 Attacker Node Source Ports

The selected ports used by the port scanning software, the software usually randomizes different ports on their attacker network node. It can be observed that *nmap* uses random source ports. In [6] the authors describe the range of chosen ports as between 33000 and (256-65536) as the default from *nmap*. *Nmap* also allows for specifying the range of source ports. The option for different source ports can be explained that in possible attack scenarios only a few ports like 54 (DNS) are allowed to access.

3.2.5 Port Scan Packet Header Fields

Different port scans use different packet header fields. As described in the background subsection 2.2.4 there are port scans like the TCP Null Scan. This means no header Flag is set. This is harder to identify if the port scan detection is using for example fixed signatures for detection. Technically, this no header port scan behaves differently to a normal port scan. The detection needs to be adjusted to handle both cases.

3.2.6 IP Address Hiding

An Attacker tries to hide his port scan activities. Therefore if the attacker is able to get multiple IP addresses within a subnet this is used as an advantage. The attack would change in the sense that different systems in the network request different ports on the target machine. This lowers the visibility and can even be abused in other ways. For example if an attacker takes over a system that he does not want within the network, the attacker could create obvious attacks, which would lead to marking or blocking the system. Another option is to just forge port scans by IP spoofing. This appears as if another system did a port scan. Botnets are another way of creating multiple different source IPs. Depending on the size of the botnet the port scan is also harder to detect. The main goal of the attacker in this case is that his port scans either are not detectable or the systems which provided the attacker with the port information are not his only access to the network. Already in 1999 it was noticed that multiple attackers work together to achieve the goal of scanning different networks [5]. This also results in distributed behavior of the port scans.

3.2.7 Intrusiveness and Packet Amount

Also the intrusiveness of the port scan can be observed. As declared in the background chapter there are SYN connect scans which identify the service behind the port. Technically this means there is at least one more packet sent to the open port. In comparison to a simple SYN Flag scan, which only identifies if the port is open. This is more intrusive because the attacker gets more information on the systems within the network. Higher amounts of packet per port results in higher likelihood of identifying the port scan.

3.2.8 Port Range

Attackers are not likely to scan the entire port range on a system. This is given to the fact that from the 2^{16} ports the most common services are run in the lower port ranges. Also attackers can have meta information about the network and therefore only scan the ranges where the attacker expects a certain service to be run. The meta information can be leaked via open source git hubs, or the attacker having inside information from working in the victim network.

3.2.9 Zombies

An approach to hide the attacker is to use a node within the network to identify the open ports. This is also used in context with firewalls as can be read in the paper from Zhang and colleges [28]. The core idea here is that a vulnerable node within the network is used to probe other services. This node is called the zombie. The system detection might identify the node and block it, which in itself introduces more problems. If the detection system is easily convinced a node is doing scanning behavior, it can potentially be used to shut down services that are needed. If the detection system is hard to convince a system is doing port scans, it will result in the fact that it wont react to normal port scans.

3.3 General Port Scan Behavior

Finally, the behavior of an attacker that is new within a subnet is discussed. In this subnet the attacker has essentially no information about end systems. This results in the fact that different hosts and services need to be identified. After observing the given attributes of the port scan attacks it is needed to identify the port scanning behavior in fast succession so the scanned system can either be protected or the attacker identified and locked out of the subnet. This thesis is aimed at exploring a novel approach on identifying port scans with an aggregated scheme.

3.4 Assumptions and Requirements

For comparability in this section it is defined what the requirements for port scan detection are. Since there is a vast amount of port scans possible, in this thesis it is focused on the following assumptions. The amount of port scans types that are established in the chapter background 2.2 are reduced. The objective is to identify straight forward and specific time delayed port scans as explained in the following.

3.4.1 Assumptions about Packet Traces

The packet traces that are investigated in this thesis are partly synthetic packet traces. However, background packet sequences are derived from real traffic. The dataset from CICIDS2017 [21] is used for background traffic. The first assumption that is established is that the packet sequence if it contains a port scan, the transmission duration of each packet is similar to other background traffic packets. The tool *IDT2* [3] is used which uses statistical approaches to ensure this transmission behavior. Further explanation on *IDT2* is given in the implementation chapter.

3.4.2 Assumptions about Port Scan Types

Since there is a high amount of port scan types it is necessary to fixate which port scan types are targeted to detect. In this thesis non-stealth SYN port scans are assumed. The port scans use the TCP SYN flag and try to identify if a port is open.

3.4.3 Assumptions Port Scan Attributes

First a non-stealth port scan is assumed to have a high amount of packets that are sent to an IP address over a short amount of time. Secondly, the timing of the port scans can be delayed, for different port ranges. It is assumed that not always 2^{16} ports are scanned, it is likely that only 1000 or less ports are scanned. These port scans do not have special features like no TCP header flag fields. Also UDP port scans, since they are having different timings and work with specificity, crafted packets are not used in the aggregation scheme of this thesis.

3.4.4 Assumptions of Network Structure

The dataset from CICIDS2017 [21] is used as mentioned before. Since this dataset with packet traces is used, that observes all node to node traffic, it is established that the entire traffic within a given network is observed. This is not always the case. On the edge of a network different traffic could occur. Also a single node within a network does not have the same capabilities. A single node can only observe traffic which it interacts with.

3.5 Requirements

After establishing the different assumptions about the port scan attacks, and network structure it is necessary to declare the requirements for state of the art port scan detection method.

3.5.1 Requirement Reaction Time

Detecting a port scan as fast as possible is the main objective. The system administrator or SDN controller should be able to react, before damage can be done. This means that the attack following up the port scan should be predictable.

3.5.2 Requirement High Recall

A requirement for reliable port scan detection is that the actual executed port scans are detected when they occur. Therefore, a high recall as described in the background chapter means that port scans that took place are detected. Recall is chosen since it does not rely on the entire network traffic. Port scans are not expected to take place in a large quantity. Recall describes how many possible port scans got detected.

3.5.3 Requirement Low False Positive Rate

The false positive rate describes which relative amount of normal background traffic got classified as port scan. It is important to keep the false positive rate small, because the consequence of a large false positive rate would mean that the alert of the detection system can not be trusted.

3.5.4 Requirement Small Port Range Attack Detection

A port scan attack that targets only a few thousand ports should be detectable. Port Scans which scan the entire port range should also be detectable.

3.5.5 Requirement Computational Power

The detection node within the network has to be computational doable for a standard architecture. This means that neither RAM, CPU or GPU can be used to an extent that would result in denying the end systems normal tasks.

3.6 Limits and Constraints

With given requirements it is identified which limits and constraints are necessary.

3.6.1 Limit Timed Stealth Scan

The approach in this thesis does not try to detect port scans that are done on small port ranges over a long amount of time (ranges of less than ten ports). Since aggregated traffic is used, these port scans are indistinguishable from background traffic. Experiments are done in the evaluation chapter to show this behavior. A port scan that is done every day for example with ten ports scanned is not expected to be detected by the approach in this thesis.

3.6.2 Constraint Host Discovery

In this thesis it is focused on port scans which try to identify services installed on a host. A port scan with this behavior sends crafted TCP packets. Host discovery scans, are done with protocols like Internet Control Message Protocol (ICMP) and do not use TCP packets. The reason being that ICMP works with a different protocol.

3.6.3 Machine Learning

The adapted approach in this thesis uses machine learning to identify port scans. This results in the fact that wrong classifications can occur. This means it is possible that if background traffic symbolizes a port scan, the machine learning approach can give a False Positive. Therefore a constraint for the port scan detection is that false positive alarms are reduced to an unnoticeable amount. Also the detection rate concerning the false negatives should be low enough to ensure that the normal non-stealth port scans are detected.

3.7 Summary

The assumptions about port scans are that the background traffic that a port scan appears in, is normal and derived from real traffic. The port scan types are reduced to non-stealth SYN port scans. A port scan usually scans around 1000 ports. Port scans happen in randomized order of ports scanned. The requirements are that the reaction time for a system administrator is low, so that countermeasures can be applied before the exploit of the system. Also a low false positive rate and a high recall are requirements. The computational power needs to be acceptable for SDN networks. Therefore, the TCAM rule amount can not be exceeded beyond 32x32. A limits is that port scans done over an extended period of time are not expected to be detectable with the detection approach used in this thesis. Machine Learning can lead to false positives, and the metrics have to be analysed accordingly.

3.8 Related Work

This section introduces port scan detection methods. It is discussed what the differences compared to the approach used in this thesis are.

3.8.1 Machine Learning Micro-Flow Port Scan Detection

First it is necessary to distinguish between a macro and micro-flow approach. A macro-flow is observing a limited amount of features within traffic. A micro-flow observes any amount of features of a given packet sequence. The approach introduced in this thesis can be categorized as macro-flow based. In micro-flow based detection data is first reworked into identifying features, and in succession this data gets used with a machine algorithm approach. The focus is to identify features with help in identifying port scans. This is done with the lasso algorithm as explained in [21]. In the paper with CICIDS2017 [21] flows are created. A flow based detection results in the fact that before detection can take place, flows have to be calculated. The next step would need to be to calculate from this flow data averages and other important identifying features. This does not work well with reaction time as defined within requirements. The port scan can take place within the flow. If more flows are first identified and then features are calculated the attacker possibility has enough time to already exploit the scanned system.
Another difference in the approach of [21] is that flows are classified. The accuracy on flow based classification by the machine learning algorithm *Decision tree* was measured as 99.84 percent. Precision and Recall are similarly high. In the approach of this thesis the aggregation is done packet wise over time sequences this is different from classifying flows, because the flows can have varying durations. The aggregation scheme uses the same time duration for each aggregated traffic chunk.
It is to distinguish between training and testing phases of the machine learning approach. It is given that the learning time for the machine learning approach is negligible (because the learning phase can be done prior to deployment). This is different for the detection or testing phase. If the machine learning algorithm is working with a flow of for example five different tuple values, all these values have to be extracted. It is observable from the dataset CICIDS2017 [21] that there is a high amount of features calculated in succession. The conclusion drawn from the above is that detection with focus on reaction time is not achievable with micro-flow based port scan detection. The main criteria to differentiate between micro-flow port scan detection, and the macro-flow oriented aggregation scheme approach is that, the approach in this thesis is to only work with network traffic packets that are aggregated in time slots as training and testing data. This aggregated data is calculated in a low amount of time as described in the design section 4.

3.8.2 Signature-based Port Scan Detection

In the field of intrusion detection SNORT [19] is commonly used. SNORT itself is a configurable multi component intrusion detection framework.
The component detection engine can be configured to detect according to a rule set. An example for a rule set based intrusion detection is signature-based intrusion detection.
Signature-based intrusion detection is a known way of filtering out packets according to a rule set. In the article of Vinod Kumar [12] it is described how signatures can be formed. An example is given that includes protocol, source address, source port, direction, destination address and destination port. Therefore, all these fields in every packet have to be checked before they are handed over to the intrusion detection system. In more detail this is done with a rule set. This rule set potentially has less values. If a packet is received that matches a rule an alert is given. With an intrusion detection system it is possible to process the packet differently, for example dropping it.
A difference to the approach in this thesis is the tuple size difference. The aggregation scheme uses only two header fields of the packet. The destination port and source IP address. Therefore, the matching rule set is significantly smaller. Signature-based algorithms do not rely on machine learning approaches.

3.8.3 Honeypot Ports and Systems

Another method for detecting port scans is to use ports on end systems that are not expected to be used given the deployment of the network. These ports however listen for incoming packets and notify the system admin or controlling instance. Another approach can be that an entire server is used as a honeypot. If an attacker node scans this server, it is marked and the system admin is notified.
Honeypot intrusion detection is explained in more detail in [26]. In the work from Chunmei YIN et al. they describe the usage of *Honeyd*. This software can set up multiple virtual hosts within a network. These virtual hosts can be configured to mimic different server types (honeypots). This approach has shortcomings. First, it is security by obscurity. The attacker could technically get lucky and does not scan the honeypot ports. Also it is likely that, if the attacker has access to a system, he knows how to get access again. This means that each time he is marked, he can step by step identify the honeypot ports. Also possible leaks of the defined honeypot ports are possible.
The main difference of this approach to detection approach in this thesis is that the checked packets differ. The honeypot systems only communicate with possible attackers directly. The honeypot system does not have access to other connections the attacker might use to apply port scans.

3.8.4 Spice detection

In [23] an detection mechanism for slow and randomized port scans is described. The main idea is that it is focused on packets sent by a scanner that are crafted and sent to unusual ports. These kinds of packets are kept longer. Using this mechanism an anomaly score is built for every network event. If the score surpasses a threshold a network admin is notified.
The difference between the spice detection approach from [23] is that they use conditional keeping of packets. Aggregating traffic conditionally is not used in this thesis. However, with using TCAM this is not practical for the requirement of low reaction time. TCAM in this thesis is used to compete with high-bandwidth networks. If the aggregation scheme relies on special conditions they have to be built in within the aggregation.

3.8.5 Threshold-based Detection

Threshold-based detection means that the amount of packets is measured and after a certain amount the source IP address is marked as malicious. This approach can be implemented in different ways. Algorithms like Threshold Random Walk (TRW) are presented in the paper of Jung [9]. These algorithms seem to work with very few port scan packets. The authors of the paper state that it only takes four or five connection attempts. The threshold-based detection is done on exact connection attempts. This is computationally exhausting for most systems. As described in the design section the approach used in this thesis uses a threshold for the labeling process. However, this threshold is applied to an aggregated amount of traffic and not per connection. Another difference is that the approach in this thesis uses machine learning approaches instead of tools like SNORT to create signatures from the results.

3.8.6 Visualization Techniques of Port Scans

In the paper from Muelder [15] an article is presented that uses grid and scatter plots for displaying attacks. The approach is similar to the aggregation scheme used in this thesis, in the sense that traffic is monitored over time with these plots. However while Muelder [15] uses different plots at once, there is no approach for the problem of high-bandwidth and there is no machine learning to automatically detect port scans. The idea from the author Muelder et al. is that a human would observe the traffic. The overall goal is to get better insights on new behaviors that can be used in an intrusion detection system. It is to mention that in the future work section of the paper from Muelder [15] the approach of using machine learning is described. This is provided in this thesis.

3.9 Summary

This section is given for an overview of the analysis. First it is assumed that port scans are often predecessors to an attack. Therefore to identify port scans attributes which characterize port scans are necessary.
The attributes are timing, packet amount, port ranges and possibly infected systems that are used as zombies to scan the network. From these attributes a general port scan behavior is defined which is needed to be detected. For this port scan behavior it is necessary to create certain assumptions about the port scans attack and the background traffic. This thesis only researches port scans which are non-stealth and use the SYN flag of the TCP protocol. Given the assumptions requirements are established that need to be achieved in order to detect port scans successfully. The main requirement being reaction time. Another requirement being feasible computational power. It is focused on creating a restricted amount of TCAM rules. The constraints describe the limits of an aggregation based detection approach. The two main constraints are that long duration (slow) stealth scans are not expected to be detected and machine learning can reply false positives in case of background traffic disruptions.
In related work first it is established that no micro-flow detection is used in this thesis. It is only focused on two specific header fields. The process can be described as macro-flow based. The next method of detecting port scans in related work, is signature based. The main difference to the aggregation approach in this thesis is the amount of header fields observed for detection. Honeypot systems can be used to trap attacker nodes. These systems contain ports that are under no circumstances used with the normal traffic designed in the network. Therefore if the attacker scans one of these ports he gets identified. This approach differs heavily to aggregation of traffic in the sense that honeypot systems are not observing the entire network. Threshold based detection is also discussed, threshold based means that if an attacker reaches a certain amount of request within a fixed amount of time, the attacker node gets identified. Another related work paper discussed is a paper from 2005 which uses a similar approach to the aggregation scheme in this thesis. It uses grid and scatter plots similar to aggregation maps used in this thesis(introduced in the design chapter 4). However, the core difference being, that no TCAM is used and therefore the advantage of this thesis's approach, is that it works in high-bandwidth networks. Another difference is that machine learning is used for automatic attack detection.

4. Design

The design chapter introduces the general idea of detecting port scans with the two step detection approach in this thesis. It is explained how the first step with aggregation scheme is used to aggregate network traffic. The second step is the detection with machine learning classification. This chapter also discusses how to tackle the problem of port scan detection within high-bandwidth networks. The Aggregation Scheme Overview section describes the used abbreviations in the upcoming sections and introduces a short overview of the aggregation scheme.

4.1 Aggregation Scheme Overview

Network traffic is monitored and aggregated in the following way: The packets are collected with certain packet attributes. These packet attributes are destination port and source IP address. Next the network traffic is chunked into equally sized exposure times. Exposure time resembles a time window that contains part of the network traffic (4.9). One exposure time contains a network traffic dependent amount of packets. The next step is the creation of a two dimensional representation (aggregation map) of the packet's attributes. The aggregation map has the packet attributes on the x-axis and y-axis. The aggregation is done by iterating over each packet within each exposure time. While iterating over the packets within the exposure time, the packet attributes are matched to buckets in the aggregation map. The aggregation map contains only buckets of the same size. The aggregation occurs by incrementing the coordinate tuple (defined by two buckets) within the aggregation map. The resulting aggregation map contains all packets within one exposure time. A port scan behavior can be observed in the aggregation map by a vertical line, since a port scan has a high amount of packets within a short time. The packets have different destination ports, therefore the y-axis with the port range describes the port scan attack. The source IP address space bucket shows which subnet the port scan attack originated from.

4.2 Aggregation Scheme

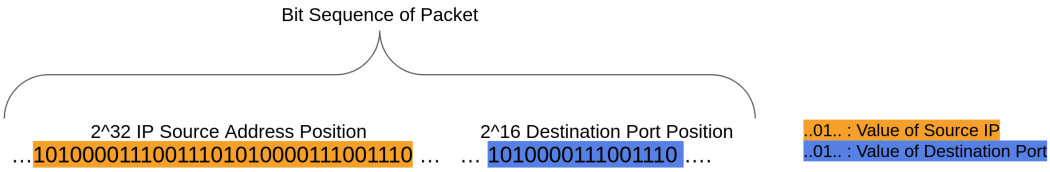
This following section introduces a sequential workflow of how the aggregation scheme is designed. The aggregation scheme is adapted from [11]. However, in this thesis different dimensions are used. The dimensions define which packet attributes of the network traffic are used. The dimensions are then used as axes for the aggregation map. In this thesis IP source address and destination port ranges are the selected dimensions. In [11] the objective is to detect Distributed Denial of Service (DDoS) attacks. Therefore, different dimensions are used.

4.3 TCAM and Aggregation

For high bandwidth networks it is necessary to adapt to the high amount of calculations done. This is necessary since for each packet attributes need to be extracted and then matched to the position in the aggregation map. If pre or post-processing of the destination port or source IP address would occur the amount of calculations would increase. This means that the matching of the packet attributes to the aggregation maps needs to be done without further calculation. TCAM (section 2.1) is used since it enables multi-field packet matching within one clock cycle. With TCAM it is possible to evaluate bit values of TCP fields for network traffic aggregation. The bit packet values are needed to match the destination port and the source IP address (target position in the aggregation map). The returned RAM address of the TCAM module then represents the position in the aggregation map that needs to be increased. This is faster than the calculation process which first needs to write to an address in memory and then evaluate the value in the memory to match into the aggregation. This ensures that short reaction time can be achieved in high-bandwidth networks.

4.4 Packet Attribute Extraction

The defined attributes (destination port and source IP address) of each packet need to be extracted from each packet in the monitored network traffic. The extraction of the attribute is done on bit level, with TCAM. An example is that from each packet, source IP address and destination port are extracted and matched in the upcoming process.
Figure 4.1 gives an example of the extraction process. It can be observed that the bit sequence of each packet is evaluated. The position of the IP header field for the IP address is determined and the position of the destination port in the bit sequence is determined. The TCAM rules use “don’t care “states for the other bit values of the packet. The extracted values for IP source address and destination port resemble tuples. These tuples have semantic value for the aggregation. As described in the example there are “don’t care “states within the evaluation TCAM rule. This is for the reason that a TCAM rule should not only contain one specific IP address but rather a subnet of IP addresses. This is for the reason that the amount of TCAM rules is limited. As described in the background chapter 2.1 the cost and energy consumption for TCAM are high. The aggregation is done towards collecting tuples from a range of IP addresses and a range of port scans. The resulting match needs to be written to memory. A more in depth explanation for TCAM rule creation is given in section 4.8, when the whole process of aggregation maps is explained.



Rule Number	RAM Output
XX..X10100001110011101010000 1XXXXXXXX..XX... X10100001XXXXXXXX..XX	(Source IP Subnet “161.206.161.XXX” and Destination Ports “41216-41471”) **to RAM AddressY**
XX..X10100001110011101010000 2XXXXXXXX..XX.. X10100001XXXXXXXX..XX	(Source IP Subnet “161.206.162.XXX” and Destination Ports “41216-41471”) **to RAM Address Z**

Figure 4.1: TCAM Packet Multi-Header-Field Matching

4.5 Aggregation Maps

An aggregation map is a two dimensional representation of a network traffic within a defined time window (exposure time). This section introduces the process of the network traffic aggregation. It is explained how an aggregation map is created and how it can be used with the previous explained packet attribute extraction.

4.5.1 Dimensions of the Aggregation Map

First the dimensions of the two dimensional representation have to be defined. In this thesis IP source address space and destination port are used as the dimensions.
Other dimensions than IP source address and destination port are possible. However, an advantage of the destination port axis is that if an attacker scans multiple targets, this is resulting in a stronger display of the port scanning activity. Since the attributes extracted from the packets are without the destination address space, it does not have an effect on the detectable port scanning behavior, if the attacker scans different network nodes. The ports scanned on different victim nodes are each included into the aggregation map, and mark the port scanning behavior more intensely. The dimensional view of an aggregation map is shown in Figure 4.2.

The x-axis is chosen as the source IP address space. This decision is inspired by the idea to identify the attacker. However, this is only a secondary goal, after the main goal of identifying that a port scan took place. The approach in this thesis is only able to identify the source IP address range the attacker started his attack from. In conclusion the aggregation maps are created with the dimensions IP source address space and destination port.

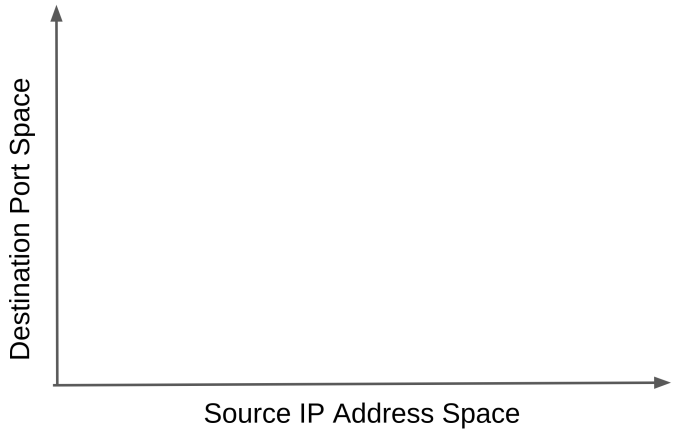


Figure 4.2: Aggregation Map Dimensions

4.5.2 Buckets and Resolution

The axis are divided into fixed amounts of equally sized buckets, e. g., 32 buckets for the y-axis and 32 buckets for the x-axis. The next step is to take the chosen attributes of the packets and divide their entire range by the amount of buckets. This provides the range for each bucket. Source IP address and destination ports are used. The horizontal axis is displaying source IP address buckets. All source IP address buckets are of equal size. The vertical axis represents a port range. With the same condition, that all buckets are of the same size. In Figures 4.3 and 4.4 an example bucket tuple is shown. It shows the limits between min and max values for one example bucket. The bucket borders define where the extracted packet attributes get fit into.
The amount of buckets of the aggregation map define the resolution of the aggregation map. This thesis focuses on different resolutions since the granularity is impacted by the predefined resolution size.

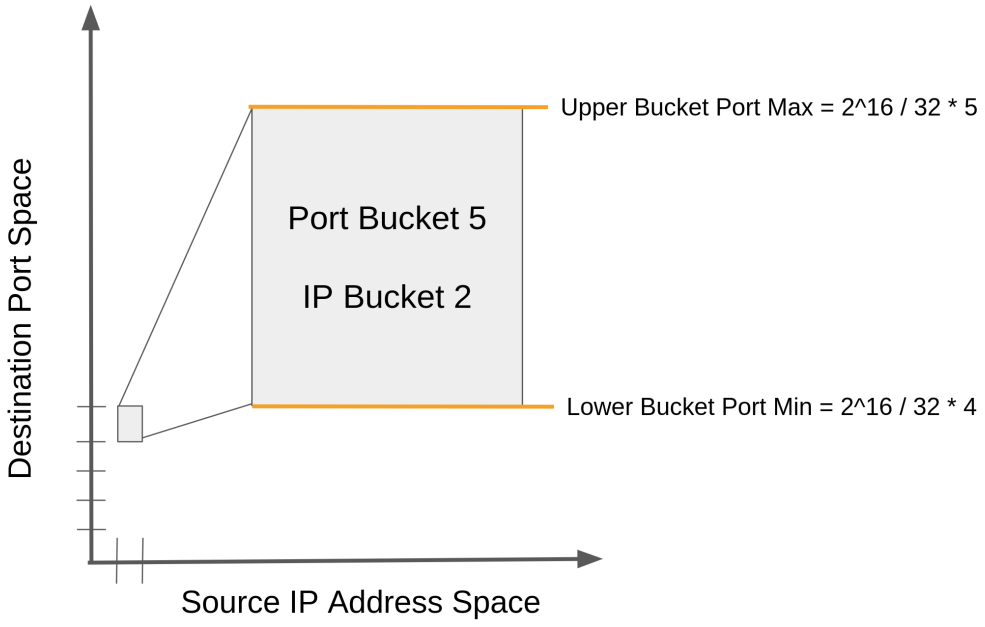


Figure 4.3: Aggregation Map Port Bucket Example

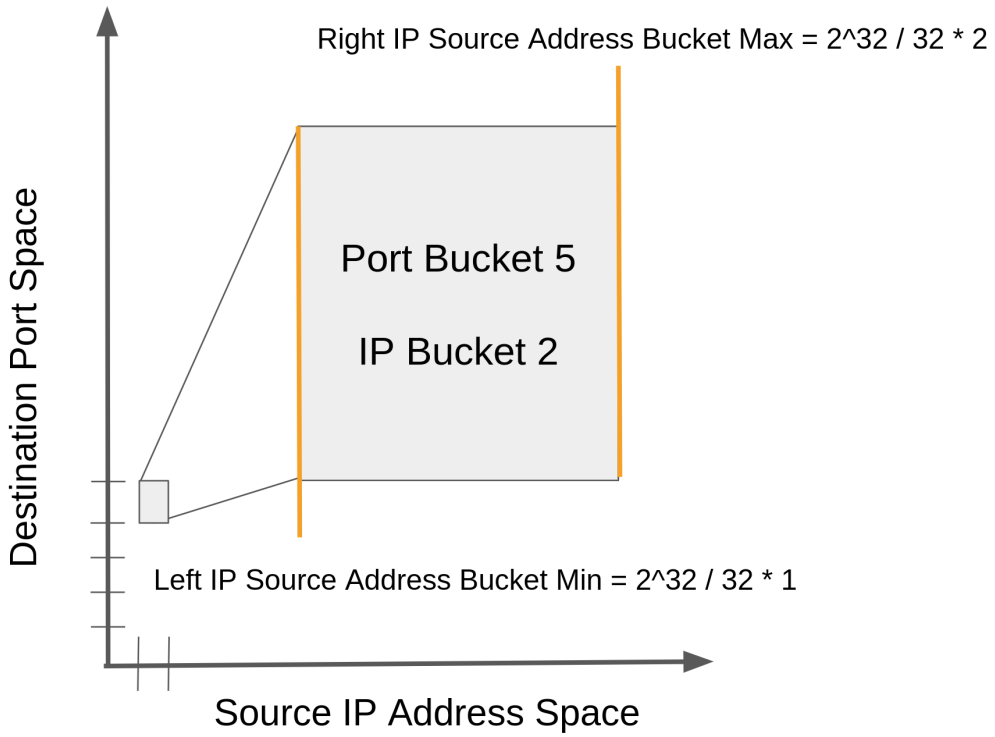


Figure 4.4: Aggregation Map Source IP Address Bucket Example

The aggregation map will show a more defining image to the machine learning algorithm if the resolution is higher. However, the amount of buckets can be adapted. This means resolutions of smaller format are researched as well. 16x16 for example is another possible resolution. This results in the fact that the bucket size grows in the sense that more ports are in one bucket. In this thesis it is focused on quadratic resolutions. This means the bucket amount of the x-axis is the same as the y-axis bucket amount.

4.5.3 Shape of the Aggregation Map

A two-dimensional representation is used. On the x-axis the source IP address space is divided into buckets. This results in the fact that the source IP buckets are divided into subnets. For example the first subnet could be 0.255.255.255 to 7.255.255.255. Each coordinate-tuple in the aggregation map therefore resembles a mapping of a source IP address range (bucket x-axis) to a destination port range (bucket y-axis). Each bucket is initialized with zero. An empty representation is shown in Figure 4.5. The integer is incremented with matches from the network traffic packets. The y-axis is determined from the range of ports. Therefore 2^{16} values are divided into buckets. TCAM has high energy cost, and is expensive. Therefore the bucket ranges are limited. Each coordinate-tuple resembles exactly one TCAM rule.

Figure 4.5: Initialised Aggregation Map

4.5.4 Building an Aggregation Map

An example of how an aggregation map is built is given:
A packet from the network traffic contains source IP address 10.10.10.3 and destination port 22. Since the transmission is done bit-wise, a bit value for each attribute is extracted as described in the packet attribute extraction section 4.4. These bit values then are calculated into which bucket they belong. In the case of IP address 10.10.10.3 the value in decimal is 168430083. The decimal value of the port is 22. These values are set relative by the total number of IP addresses and ports. The maximum number of IP addresses is 4294967295. Therefore to identify the bucket first the fixed amount of buckets is divided by the entire IP address space (4294967295). In this example a resolution of 32x32 is used. Therefore 4294967295 is divided by 32. Fixed integer values are used which means the result is rounded to 134217727. This resembles the size of each bucket. Finally the index in the aggregation map is calculated, by dividing the IP address value 168430083 by the size of each bucket. The result is bucket one (rounded). The same process is done for the port. The given example destination port and source IP address therefore will increment the value in the array at index one for the IP address and zero for port. It is important to note that the y-axis is inverted. Therefore this incremented coordinate tuple in the aggregation map is in the lower left corner of the graph.

A further schematic example with TCAM is shown in Figure 4.6. In this example the bit sequence gets analyzed with TCAM. Semantically this means the coordinates within the aggregation map are known. The TCAM rules match the values of the destination port and source IP address to the correct bucket in the aggregation map.

This example process is done for each packet within the given fixated exposure time. The exposure time is explained in the next subsection.

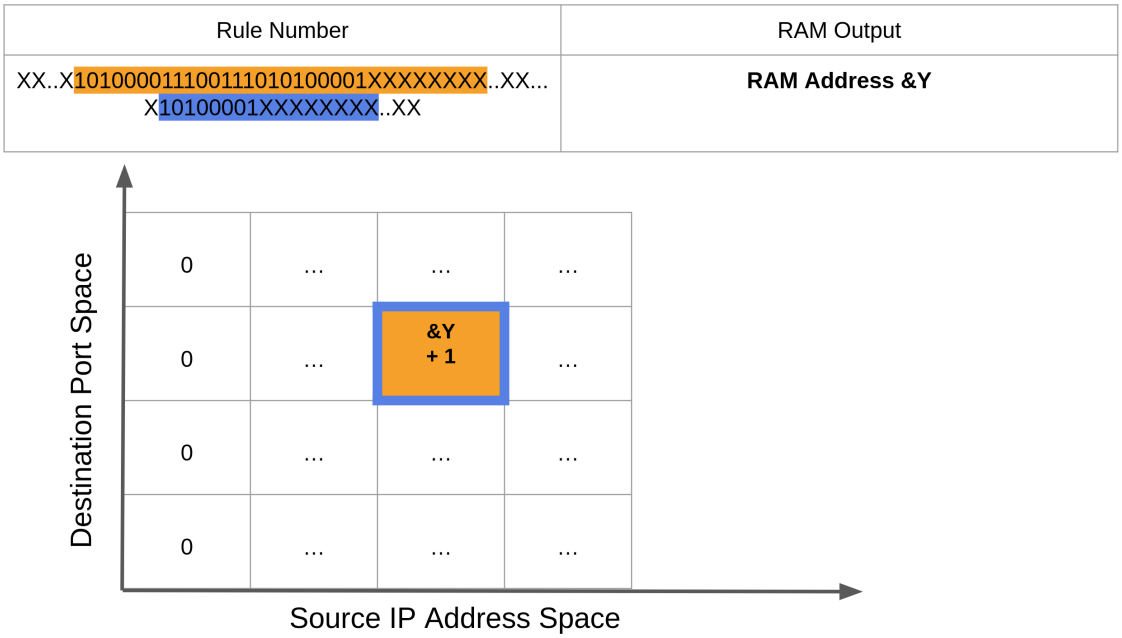


Figure 4.6: Aggregation Map Matching

4.6 Visualization of Aggregation Maps and Port Scans

For explaining key mechanics in the aggregation scheme aggregation maps are visualized. To facilitate human readability gray scaling is used. However, the machine learning model input is normalized before using it as input for the machine learning model. This means that the actual representation differs from a grayscale image, because the grayscale image is created by using absolute values.
With a port scan attack it is assumed that a high traffic occurrence towards the same IP address takes place. The traffic occurrence created by the port scan targets the same IP address but different ports. Since the aggregation map is monitoring different destination ports, the aggregation map is supposed to contain a vertical line per port scan. This is symbolized in grayscale in Figure 4.7. The position on the x-axis resembles the subnet bucket the port scan is coming from. A vertical line is noticeable. This is the behavior a machine learning model is supposed to detect. The Figure shows a clear port scan with low background traffic, however more network traffic can occur and is usually present. The port scan in this case is created in source IP bucket two. And reaches the complete range of buckets for ports. The assumption is made that all ports are scanned. It can be observed that the port scan is gray and not white. This can be explained with the random order of port scans. If the attacker used the random port order, this means the port scanner scans, for example port one and then port 30000. This means the overall distribution in the buckets can vary depending on exposure time of the aggregation map.

Machine learning should identify the difference to normal background traffic as shown in Figure 4.8. As shown in the Figure there is no vertical line. The coordinate tuples in this aggregation map show normal services within the network that communicate with each other.

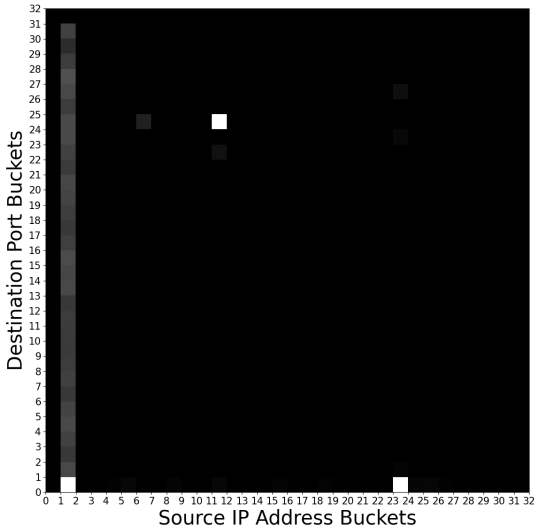


Figure 4.7: Example Aggregation Map during a Port Scan Attack Greyscale Image

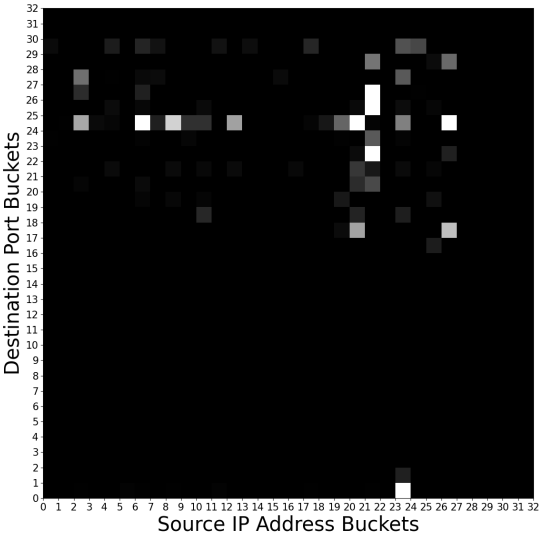


Figure 4.8: Example Aggregation Map with Background Traffic Greyscale Image

4.6.1 Aggregation Map Exposure Time

The idea of the aggregation scheme is to collect the information of the attributes over a certain duration. The duration in this thesis is kept consistently the same for one detection process. This can be observed in Figure 4.9, the exposure time windows have exactly the same fixed duration. This is ensured with a timer. Every time the timer resets the current aggregation map is finished and the next aggregation map begins. It is observable that different traffic intensity can occur at different exposure times. There are more packets within the first exposure time.

The exposure time is the amount of time that the packets are counted into one aggregation map. In figure 4.9 that means that the first *packet* and the second *packet* are within the first aggregation map and the third *packet* is used in the next aggregation map.
This means that packets from the arriving network traffic are separated in time windows.

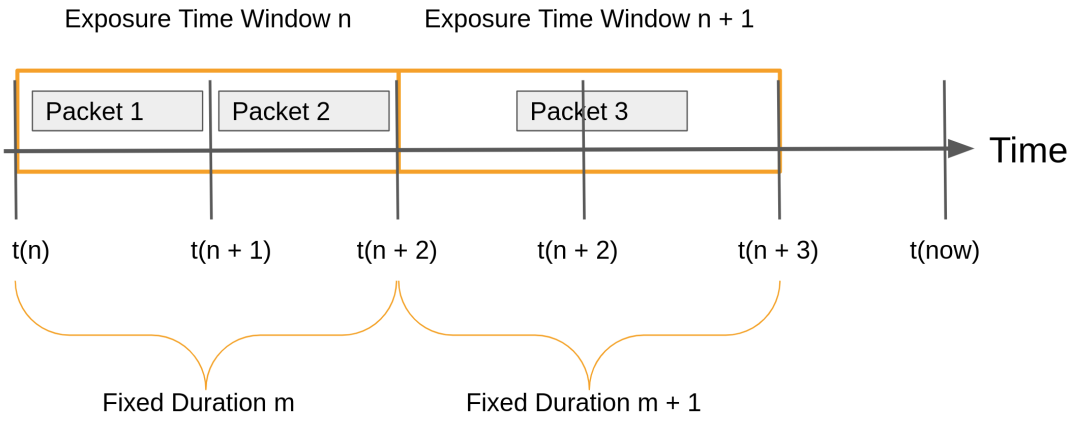


Figure 4.9: Exposure Time Windows

All packets that are monitored within this time window are used to increment the aggregation map. The exposure time is a parameter that can be adjusted for optimization. A short exposure time duration is resulting in not having the entire port scan within one aggregation map. Also in small networks short exposure times result in empty aggregation maps. Long exposure times are problematic with reaction time, as well as the detection can get more difficult. The reason being that more background traffic is aggregated into one aggregation map. As shown in the grayscale Figure 4.8 the background traffic can get more intense. With a longer time period a high amount of packets are within one aggregation map. If the scanned target has a lot of other connections in this exposure time, this shows a harder to classify aggregation map. Another aspect is higher exposure times might include multiple port scans. Figure 4.10 shows that if for example every 30 seconds a port scan takes place, there can be overlap of port scans when the exposure time is longer than one minute. With lower exposure times, these port scans are not within the same exposure time. The machine learning model should be trained with aggregation maps that contain multiple port scans in short succession.

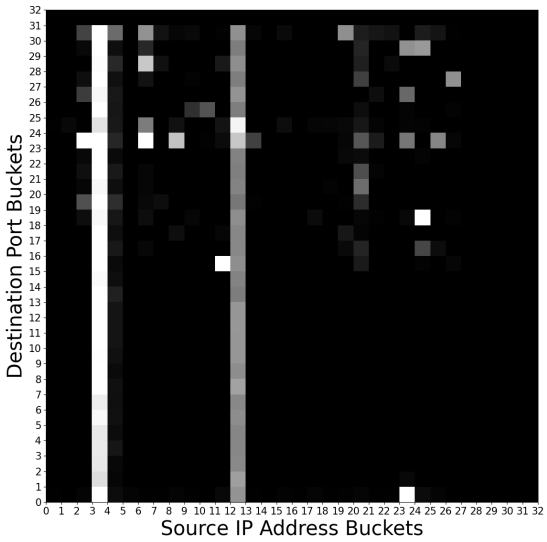


Figure 4.10: Example Aggregation Map with Multiple Port Scans

4.6.2 Threshold and Labeling

An aggregation map needs to have a label for binary classification. The label should represent if the aggregation map contains a port scan or not. For this reason a fixed absolute threshold is used. Absolute meaning that not the overall amount of packets in the exposure time are evaluated, but rather the amount of port scan packets. This means that a fixed amount of packets is defined to determine if an aggregation map contains a port scan or is benign. This means e. g. (threshold is ten) that if in the exposure time of the aggregation map more than ten packets belong to a port scan, the aggregation map is labeled as port scan.
For this reason its labels are required on packet granularity. In the packet labeling process the labeled packets are counted for each exposure time window. If the port scan packet amount is higher than the threshold the entire map is labeled as containing a port scan.
With shorter exposure times the threshold becomes more important. When there is network traffic with a port scan over three minutes and the aggregation map exposure time is around half a second, the absolute threshold has to be adjusted to smaller ranges. However, if the exposure time is for example one minute, the threshold should be adjusted to be higher. The conclusion of the threshold experiments is described in the evaluation chapter 6.4.

4.6.3 Normalization

Background traffic can fluctuate, therefore the representation of the aggregation map can also change. Background traffic is sensitive to new connections made by the servers within the network. This is one example for traffic fluctuation in the sense that there is going to be more incremented values within one aggregation map. For example in some windows there are only a couple hundred packets, where in others, there are ten thousand or more packets. The background traffic can therefore manipulate the visibility of a port scan. To improve model learning z-scoring is used, as explained in the background section 2.3. It is important to note that the standard deviation of the population as described in 2.3 is calculated within a time window. This means that each aggregation map is normalized on its own. Further research on using the entire network traffic is done in related work.

In Figure 4.11 the process of normalization is shown. The aggregation map contains a port scan from bucket one. It can be seen that the values representing the port scan are way less extreme after the normalization.

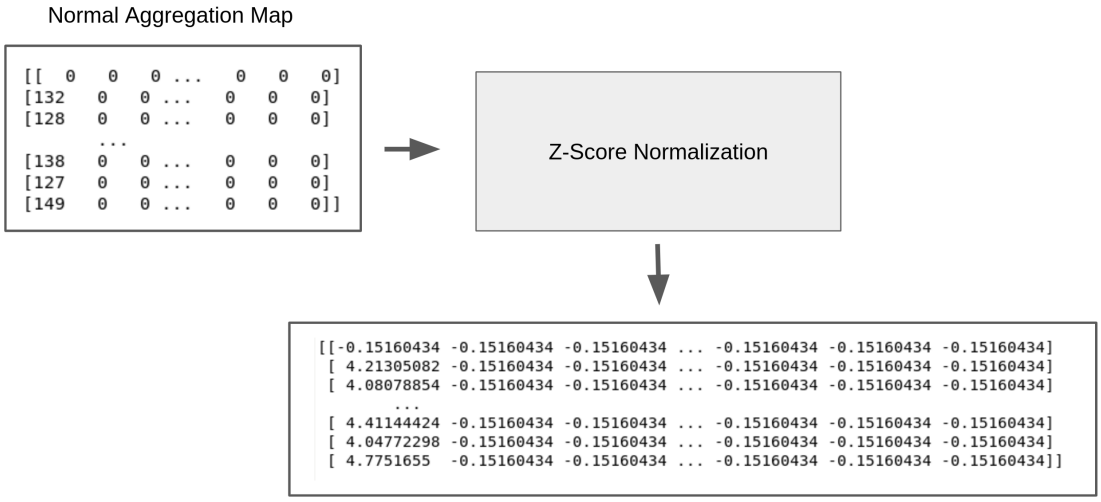


Figure 4.11: Aggregation Map Normalization

4.7 Aggregation Map Port Scan Properties

This section discusses what properties an aggregation map which was created during a port scan contains. The expected behavior is that the attacker node scans a high amount of ports of the victim node. Since the port range is on the y-axis the resolution has an impact on how the port scan is displayed. A small resolution results in a longer vertical line. This is for the reason that smaller resolution has a larger amount of ports within one bucket. A normal port scan however contains only about a 1000 ports scanned. With a resolution of 32x32 this means that the entire port scan is within one bucket. This one bucket has an extreme value compared to its adjacent buckets. With normalization this bucket still has a strong identification property.
The next property discussed is introduced by port scanning software scanning from different source ports. The aggregation map also contains the reply packets for the port scan packets. This means that the source port that was used by the port scanning software is used by the victim node. The victim node that got scanned will reply with the destination port as the source port from the attacker. The ports used by the port scanner can use the complete range of ports. This creates a second line within the aggregation map. The victim node sends all the packets back to the attacker. Since the aggregation map contains the IP source address this highlights the victim node source IP bucket.

4.8 TCAM Rules for Aggregation Maps

TCAM is employed for supporting high-bandwith network traffic. For this reason a TCAM rule structure must be created. The bucket sizes (resolution) define the amount of matching rules needed. Each coordinate-tuple resembles one matching rule. The amount of matching rules therefore is NxN. It is to note that the resolution can be adapted. An example scheme for a resolution of 16x16 is given in tables 4.12. In the tables the individual necessary rules are displayed. They get matched from each packet as described in Figure 4.1. 16 buckets matching rules exist for each the IP source address and destination port. The TCAM rules only differ in the four highest bits respectively. However, the

IP source address has a larger header field than the ports. This means that the TCAM rule length from the ports only has a length of 16 bits and the IP source address has 32 bits. The decimal representation range shows the total amount of ports and IP source addresses that need to be matched. The amount of different ports is $2^{16} = 65536$. Ranging from zero to 65535. The IP source addresses have $2^{32} = 4294967296$ different values. They range from zero to 4294967295. The next step is to combine the two rule sets. Each source IP address needs to create a bucket for each possible port. Therefore, in Figure 4.13 the combination is shown. The first bucket column starting from the left lower corner is shown. In combination this results in $16^2 = 256$ TCAM rules for a resolution of 16x16. It is to note that when TCAM is used, not all resolutions are achievable. A resolution of 8x8 would need bucket sizes of 8096 each. Using the scheme from above this is not possible with the four preceding bits.

Port Match field (16 Bit)	Decimal Representation	Source IP Address Match field (32 Bit)	Decimal Representation
1111XXXXXXXXXXXX	61440-65535	**1111XXXXXXXXXXXXXXXXXXXXXXXXXXXX**	4026531840-4294967295
..	..	..	..
0001XXXXXXXXXXXX	4096-8191	**0001XXXXXXXXXXXXXXXXXXXXXXXXXXXX**	268435456-536870911
0000XXXXXXXXXXXX	0-4095	**0000XXXXXXXXXXXXXXXXXXXXXXXXXXXX**	0-268435455

Figure 4.12: TCAM Rules for Ports and IP Source Addresses

Match field (Packet Bits)	Decimal Representation	Name and (row,column) Grid Position
.. 1111XXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXX .. 1111XXXXXXXXXXXX ..	4026531840-4294967295 (IP Source Address) 61440-65535 (Ports)	Bucket 256 (16,16)
..	..	..
.. 0000XXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXX .. 0001XXXXXXXXXXXX ..	0-268435455 (IP Source Address) 4096-8191 (Ports)	Bucket 2 (0,1)
.. 0000XXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXX .. 0000XXXXXXXXXXXX ..	0-268435455 (IP Source Address) 0-4095 (Ports)	Bucket 1 (0,0)

Figure 4.13: Combined TCAM Rules

4.9 Machine Learning Input with CNN

The next step is to use the normalized aggregation maps as input for the machine learning model that is trained supervised. In this thesis Convolutional Neural Networks (CNN) are used. The CNN applies image classification. The machine learning model is trained to identify the traffic patterns shown in Figure 4.7 of a port scan in the aggregation map. Since the aggregation maps are used as images, the color depth is declared as one. This defines that there are no RGB values in the aggregation map image. The value in each pixel of the image represents the normalized traffic intensity.

4.10 Balancing Machine Learning Input

For the machine learning model, balanced input data can be used. This means the same amount of port scan aggregation maps and benign aggregation maps need to be fed to the machine learning model. However, datasets are limited by their total duration. This means that with shorter exposure times the total amount of aggregation maps decreases significantly. A graph displaying this behavior can be viewed in the evaluation chapter 6.2. This means that for the detection training of the machine learning model significantly less aggregation maps are provided with higher exposure times. On the other hand, an aspect is that port scans are not supposed to happen 50 percent of the time. This means that the machine learning model could potentially profit from an unbalanced input. Also an anomaly detection approach could be used. This means that the larger amount of aggregation maps containing only benign traffic could help the machine learning model in classifying background traffic with higher precision (meaning there are less false positive classifications). For these reasons, experiments where balanced aggregation maps and unbalanced aggregation maps are used are researched.

4.11 Machine Learning Model (CNN)

In this thesis a convolutional neural network is used. As shown in Figure 4.14 the input layer receives an image defined by the bucket size (resolution NxN). In this thesis a repetitive pattern is used. The Figure 4.14 shows that when the first convolution is done, the kernel size is (2,2). With back propagation the kernel is trained. The next step is to remove negative values from the input with the activation function Relu. Next the image size is reduced with a two dimensional max pooling layer. The reduction helps identify important characteristics. Since the maximum is taken the port scans should be highlighted more in this step. This pattern is applied three times. Then since the result has to be a binary classification which describes if either a port scan took place, or the aggregation map image shows benign traffic, the input is further reduced.

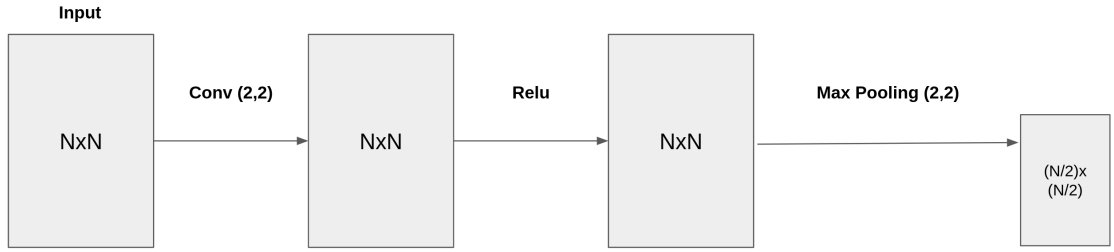


Figure 4.14: CNN Model Pattern

4.12 CNN Classification

The last step is to use the binary classification of the machine learning model to classify given validation data. For this reason in Figure 4.15 the dimensions of the previous input need to be reduced further. Therefore, first a flatten layer is used to transform the two dimensional shape into a vector. Next, a dense layer is used and an output shape of 64 is fixed. Next activation relu is used to remove negative values. Further the vector is reduced by removing half of the neurons with a dropout layer. Next from the resulting 32 values with a dense layer a value between zero and one is extracted. The value then is evaluated with a sigmoid activation function to determine if it is closer to zero or one. The result is a binary classification that describes if the input contains a port scan or benign traffic.

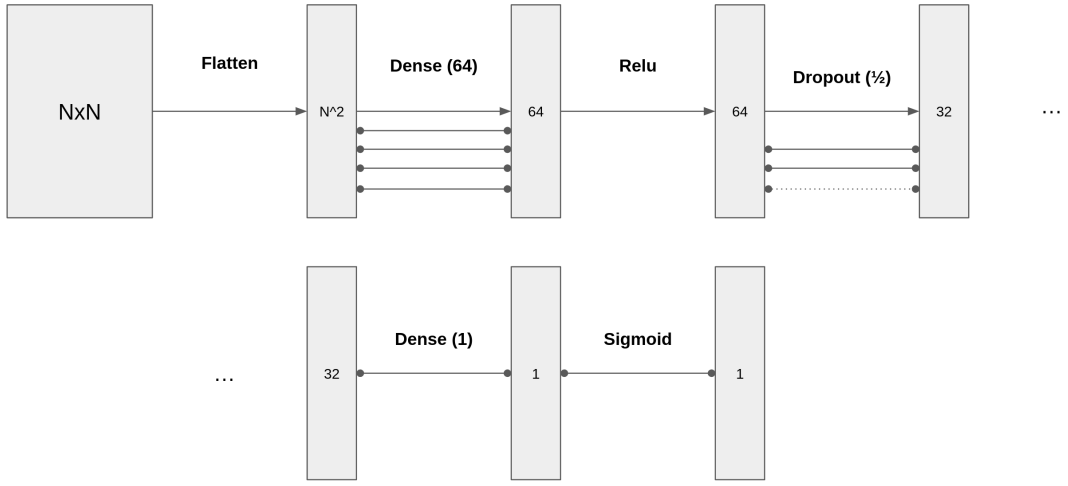


Figure 4.15: CNN Model Classification

For the evaluation the introduced metrics from background chapter are used on the classification results.

4.13 Optimizing Aggregation Map Parameters

In this section it is described which parameters of the aggregation scheme are focused on to optimize. As mentioned before, the dimensions of the aggregation map are the IP source address on the x-axis and the port range on the y-axis.

4.13.1 Exposure Time

The exposure time is the first parameter to optimize. The exposure time has a strong influence on each aggregation map. With longer exposure time for one aggregation map more packets are included in one aggregation map. With less exposure time a port scan is more often split and not within one aggregation map.
In Figure 4.16 traffic patterns are described. Orange shows the exposure time windows, the first exposure time is double as long as the second. In the first exposure time port scans take place. It can be observed that the first port scan is within one exposure time in both cases. However, in the first exposure time Figure the port scan is accompanied with benign traffic while in the second exposure time variation the port scan is almost alone in one exposure time window. Another scenario is displayed in Figure 4.16 where with shorter exposure time windows the port scan is not entirely within one exposure time window. This is more likely with shorter exposure time for each aggregation map. The aggregation maps can change because of this behavior, because the port scan order is random. Meaning that the port scan potentially is way less visible within the exposure time. Also the port scan in the Figure is distributed equally within each exposure time, however also it is possible that only a few packets are in one exposure time, while the other is a clear port scan. These edge cases, should create hard to classify aggregation maps for the CNN.

In this thesis it is researched at which exposure times, the machine learning model performs best. The performance is measured with the metrics given in the background section 2.6. The main objective of optimizing this parameter is to identify at which exposure time detecting port scans is optimal.

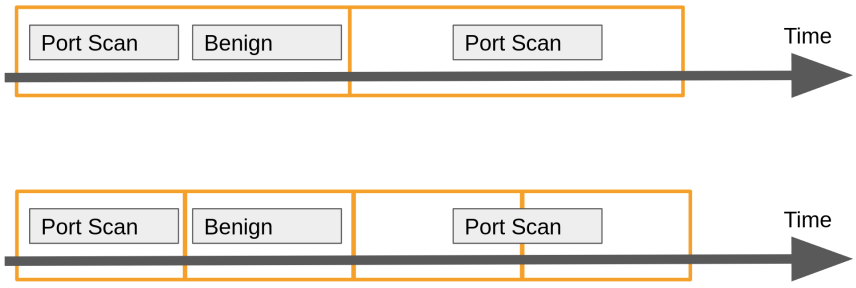


Figure 4.16: Different Exposure Times

4.13.2 Resolution

The resolution technically refers to the amount of buckets used. On the one hand this strongly correlates with how many TCAM rules are necessary, and on the other hand it is important to notice when an aggregation map is too clustered to distinguish a port scan from background traffic.
Also with less TCAM rules the amount of ports and IP source addresses grows significantly in each bucket. This is shown in 4.17, while with a resolution of 2x2 the ports that are contained within one bucket are 32768, they already reduce to 16384 with a resolution of 4x4. The port clustering is reduced with higher resolution.
It is to mention that with smaller resolution, the CNN model needs to use zero padding. This is for the reason that convolutional layers with 2x2 kernels need to have space to calculate in each stride and max pooling layers reduce the size. Since zero padding is used, the lower bound of 2x2 for the resolution can be used.

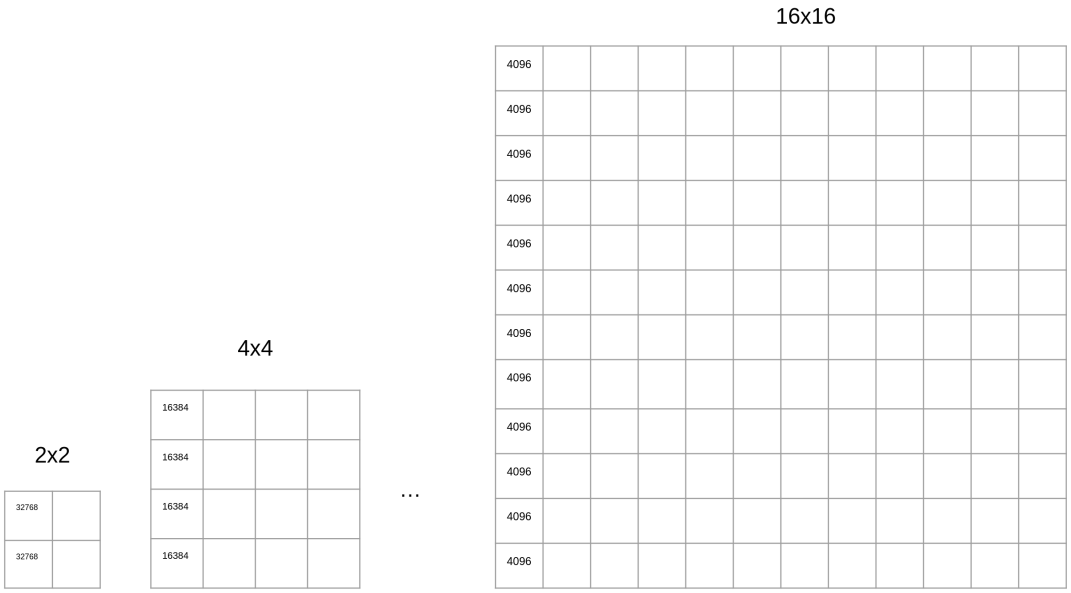


Figure 4.17: Different Resolution Port Bucket Sizes

The goal for the resolution parameter is to identify which resolution works best with the given machine learning model. The maximum resolution researched is 32x32. This is the upper bound since more TCAM rules would result in an undesirable amount of TCAM rules.

4.13.3 Threshold

Finally, the labeling process of the aggregation maps needs to be optimized. The threshold for each aggregation map is used with the packets that are within one exposure time. Meaning that if the aggregation map's exposure time overlaps a network traffic with a port scan, the packets that are port scan are counted. If they surpass the threshold the entire map is labeled as containing a port scan.
This part is tricky since the goal should be to identify port scans that occur over a large amount of time, with only very few ports scanned. A high threshold might lead to label these port scans as benign traffic.
As mentioned in the analysis however the goal is to identify non-stealth scans prior to the highly stealth ones. Which results in the fact that small thresholds e. g. ten can be used for long exposure times.
To identify the best thresholds different thresholds are tested as shown in the implementation section 5.3.4. It is assumed that if the exposure time is reduced strongly also the threshold should be reduced. This means if the exposure time only represents a fracture of a second it is advised to use a threshold of one.

4.14 Timed Stealth Port Scans

Port scans which are distributed over long time periods with a low amount of ports scanned, are harder to detect. Especially in the aggregation scheme, they appear as normal background traffic. The Figure 4.18 shows the process of a timed stealth port scan. Between each port scan multiple days go by. Meaning that the port scan never exceeds more than a few ports per day. In a normal network this is nearly undetectable, since other services that try to connect to certain ports could be responsible for these packets. It is to note that these port scans are not necessarily from port one to ten. The port scan might use a list like ports 22, 80, 443 to just check for commonly used service ports.

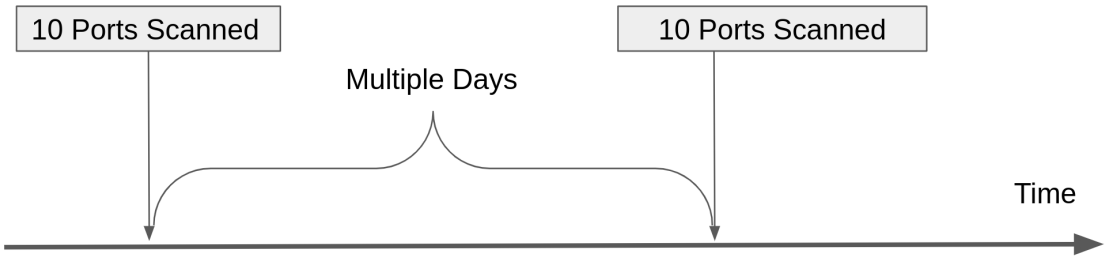


Figure 4.18: Timed Stealth Port Scans

4.15 Summary

The aggregation scheme translates network traffic into aggregation maps. This process is done by selecting specific attributes from each packet within the network traffic. These attributes are source IP address, and destination port. The next step is to create a two dimensional graph that represents the attributes. The graph has an x-axis which is used to divide the IP address space into buckets. The y-axis is used to divide the port range into buckets. The bucket amount is referred to as resolution. Next the packet attributes are extracted for each packet within every exposure time. These attributes then are sorted into their buckets, and the value in the bucket is incremented.

After the process of the aggregation scheme, the machine learning approach is used. First the aggregation maps are normalized. This is done by z-scoring each aggregation map individually. Next each aggregation map is labeled with a fixed threshold, and then given to an input layer for a machine learning model. The fixed threshold describes how many port scan packets are within the aggregation map. If the amount of port scan packets is higher than the threshold, the aggregation map is labeled as containing a port scan.

Next the labeled, normalized aggregation maps are fed to a machine learning model (CNN). The CNN binary classifies each aggregation map. The results of machine learning are evaluated with different metrics, as explained in the background section. For aggregation scheme optimization the following parameters are researched: exposure time, resolution and threshold. Further harder to detect port scans are researched.

5. Implementation

First the implementation chapter introduces the setup of the design section. This means it is explained how the aggregation scheme and the machine learning model are set up to classify network traffic datasets. It is explained in detail which parameters are used for the creation of aggregation maps. After the setup, it is described how the experiments on exposure time, resolution and threshold are created. The different experiments where the attacker is synthetically simulated to hide his scanning behavior are explained. Finally the CICIDS2017 comparison experiment is described.

5.1 TCAM in Experiment Setup

In the experiments in this thesis, TCAM is not used. This is for the reason that network traffic datasets are used for training the machine learning model. TCAM can be used to create new aggregation maps, which can be predicted with the trained model from the work in this thesis. However, to evaluate the performance of the machine learning model TCAM is not necessary, because it is only used during system deployment.

5.2 Aggregation and Machine Learning Setup

The upcoming section introduces how the design got implemented. It is explained how attacks get injected, how attack traffic is injected into background training and how the classification with the machine learning model is done.

5.2.1 Dataset with Network Traffic

To test the aggregation scheme a dataset with network traffic is necessary. The dataset needs to have the possibility to inject port scan attacks into the network traffic. Therefore, the two main criteria are: (1) realistic background traffic and (2) configurable port scans. To achieve realistic background traffic the dataset from CICIDS2017 [21] is used. The dataset contains one working week of network traffic. This means there are five days with eight hours of network traffic. For the injectable background traffic the Monday traffic without attacks is used. The PCAP file from Monday contains eight hours of benign traffic, which is best for attack injection since no other attacks need to be filtered

out. Both labeled flows and a packet trace are contained within the CICIDS2017 dataset. The aggregation scheme uses packet by packet analysis therefore the PCAP file from CICIDS2017 is used.

5.2.2 Synthetic Attacks

As described in the objectives, specific port scan attacks need to be injected into the background traffic. The CICIDS2017 dataset contains port scans already. However, it does not have distributed port scans over different time windows. For the training of aggregation maps the port scans from different buckets is necessary. Otherwise the machine learning model classifying the aggregation maps could learn that the attacker node only uses one IP source address. For this reason the port scan attacks are injected into the Monday PCAP file as explained in the next section.

5.2.3 Port Scan Attack Injection

After the background traffic PCAP file is chosen, the next step is to inject attacks. The attack injection has to be carefully evaluated since the conditions of the network still apply. For this reason a tool is used which uses statistical databases to assert that the attacks are matching the conditions of the network traffic PCAP file. The tool is *IDT2* [2]. *IDT2* can be used to inject port scans at a given timestamp. The port scan attack is based on a statistical database that *IDT2* creates before it injects the port scan attack.
IDT2 supports different parameters to change the port scan attack. The timestamps are chosen beforehand and have to be configured to be within the duration of the background traffic. This means that the start timestamp from the CICIDS2017 PCAP on Friday needs to be configured. Different port scans can be injected by defining IP source address and timestamps by reusing the already injected PCAP file. *IDT2* uses the background traffic from CICIDS2017 to create statistical databases which define the packet transmission times. With these transmission times port scan packets are injected. It is to note that IDT2 has more parameters, however in this thesis timestamp, ports (amount and specific range), IP source address (attacker), IP destination address (victim) and the generic parameter port scan attack are the parameters used. The result of *IDT2* is a PCAP file containing preconfigured port scan attacks.

5.2.4 Port Scan Attack Packet Labeling

The port scan attacks which are injected into the PCAP file need to be labeled. Since the research in this thesis focuses on packet by packet analysis each packet needs to be labeled benign or port scan. First a CSV file format is created with *tshark*. *Tshark* is the command line interface of wireshark [13]. With *tshark* the PCAP file is converted to CSV. From the above section it is known that the IP source address and IP destination address are used for the attack injection. In the benign traffic on Monday a unique victim IP is used. This makes the labeling process straightforward, in the sense that each packet with destination IP address matching the unique victim IP address is labeled as port scan. After the labeling process, a CSV file is created where each row (representing a packet) has a label.

The Figure 5.1 shows the process. The destination address¸ therefore needs to be kept in the CSV file till the labeling process is done. It is a network unique address that is not in the benign traffic PCAP file. For the further aggregation only the other columns are necessary. A normal CSV packet trace contains significantly more columns with packet information. These can be removed after the labeling process to reduce computational work.

IP Source Address	Destination Port	Destination Address	Timestamp	Label
192.168.0.50	22	9.8.7.6	1507963405	1
192.168.0.50	23	9.8.7.6	1507963406	1
192.168.0.51	80	192.168.0.5	1507963410	0

Figure 5.1: CSV Trace Label Process

The labeling process has to be adjusted for the experiments concerning the Friday port scans of CICIDS2017. In the experiment setup 5.3.8 the labeling is described. When the CICIDS2017 dataset is labeled, not only the destination address is evaluated, but also the IP source address. This is because the victim node has traffic not connected to the attacker.

5.2.5 Aggregation Map

The next step is to build the aggregation maps. As described in the design section, only specific fields of the CSV file are necessary. In this case source IP address, destination port and timestamp are extracted. With the timestamps the CSV file is divided into chunks. The chunks timestamp borders are defined prior to building the aggregation maps from the packet labeled CSV file. They resemble the in the design section explained exposure time. Since in this thesis the Monday traffic is used, the exposure time is calculated relative to the amount of aggregation maps. This is done to keep comparability so that the entire duration of eight hours is evaluated. Using predefined exposure times can result in overlap over the start and end time of the packet trace. The traffic on Monday from CICIDS2017 contains roughly 28800 seconds. If 28800 aggregation maps are specified each aggregation map has a time window of nearly one second. Dividing the dataset, results in chunks that each need to be formed to an aggregation map.
Therefore, an array with zeros is initialized for each chunk. The next step is to iterate over each packet in the CSV data and increment the corresponding position in the aggregation map. The x-axis as defined in design resembles the IP source address space. The y-axis resembles the destination port. First the both axis need to be divided into buckets. These buckets all have the same size. For the x-axis the IPv4 range is used. Next it depends on the resolution how many buckets are created. If the resolution is given as 32x32 then the source IP addresses (2^{32}) are divided by 32 and the destination ports (2^{16}) as well. This results in the actual bucket size.
Since the aggregation scheme starts from zero for the port's axis, and grows to a larger port's upward, the y-axis is inverted. This is done because of the structure of an array and the aggregation maps that are provided to the machine learning model in the next steps. As explained in design the port scan attack should resemble a vertical line in the aggregation map. This means if ports 1-65536 are scanned. On a gray-scale intensity encoded image of the aggregation map a white line should be viewable in the corresponding source IP address bucket on the x-axis. A two dimensional array starts with indexes zero and zero. This resembles the top left corner of the aggregation map. Therefore the vertical line should if the port scan is done ordered start from the bottom to the top of the aggregation map. This is done on each aggregation map.
Next the CSV chunks are evaluated. Each row represents one packet. The packets are iterated over within each chunk each time the destination port and the source IP address bucket is calculated. Each time the buckets are calculated this results in a coordinate tuple within the aggregation map. This resulting coordinate tuple then is incremented for each packet. Since each chunk in itself resembles a time frame, the timestamp is not evaluated in this step.

The process is shown example wise in 5.2. First the IP source address gets evaluated, next the destination port. There buckets get calculated and next the position in the array is incremented. The example contains a small resolution. The IP addresses are transformed into decimal integers to calculate their bucket. With TCAM rules this processing is not necessary. When each chunk gets iterated over, the amount of chunks is equal to the amount of aggregation maps created from the packet trace. *Numpy* vectors in CSV format are used to save these aggregation maps. These vectors can be reshaped into the aggregation map's original shape.

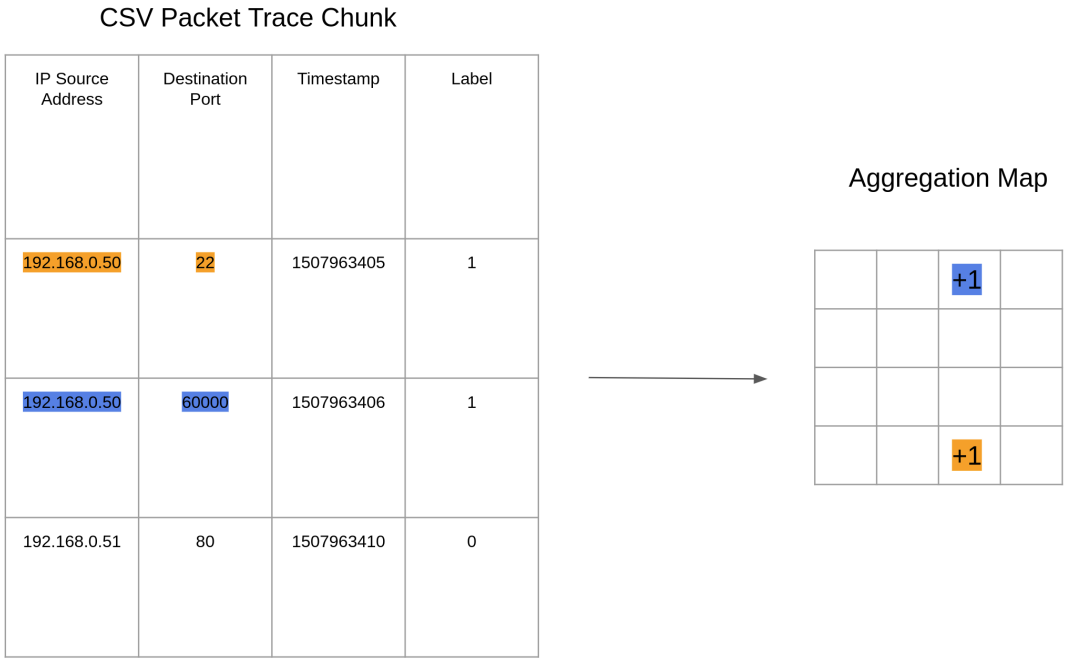


Figure 5.2: Aggregation Map Creation

5.2.6 Aggregation Map Labeling

Since the machine learning is done supervised in this thesis, each aggregation map needs a label. As mentioned in the design section a fixed threshold is used. For example if the threshold is defined as ten packets (labelled as port scan) each aggregation map with more than ten port scan packets is labeled as containing a port scan. The input for the aggregation map labeling is a CSV file (for one aggregation map) with a label column. The labels are counted and compared to the predefined absolute threshold. If the amount exceeds the threshold the entire aggregation map gets labeled as containing a port scan. The labeling of the packets is done as explained in the attack injection labeling section 5.2.4. This means that only the label column of the packet trace CSV is evaluated for the labeling of the aggregation map. The threshold in itself is a parameter of the aggregation maps. Therefore it is adjusted as explained in the experiment setup. The label gets appended to the *numpy* vector saved with the aggregation map.

5.2.7 Aggregation Map Balancing

The creation of the aggregation maps is separated from the machine learning classification. Therefore, the previous *numpy* vectors which are saved to CSV are loaded and transformed into their original shape. Each aggregation map has a label, therefore it is possible to identify the distribution of port scan maps and benign maps. To ensure that the metrics for the machine learning model are working properly the aggregation maps can be balanced. This means that either benign or port scan maps are reduced till the same amount of port scan and benign maps are within the dataset of aggregation maps. As explained in further experiments with CICIDS2017 balancing is not always used. However, concerning the experiments that are done on Monday CICIDS2017 PCAP network traffic, the balancing is applied as described.

5.2.8 Normalization

After balancing the aggregation maps, the next step is to normalize each aggregation map. This is done with a z-score. It is implemented so that the standard deviation is used for each data point. Each aggregation map for itself calculates its own standard deviation and mean. This means that not all possible data points in the dataset are evaluated but rather each aggregation map by itself. The z-score formula is applied to every coordinate tuple in the aggregation map.

5.2.9 CNN Framework Keras

In this thesis *keras* is used for the machine learning approach. An introduction to *keras* can be found in [14]. *Keras* is used to binary classify the aggregation maps. *Keras* is a python framework with many parameters. For example *keras* already uses backward propagation, and layers as described in the design section are configurable.
To provide input for *keras* first the label is extracted from each aggregation map and appended into a label vector. *Keras* receives an input shape of (1000,32,32,1) and a label vector of (1000,1). This example resembles 1000 different aggregation maps with a resolution of 32x32 for each aggregation map. Another important aspect of *Keras* is that it splits train and testing aggregation maps by itself if configured accordingly. This is described as a validation split and is set for 20 percent for the experiments.

5.2.10 CNN Model

Keras is provided with a given machine learning model. As shown in the Figure 5.3 the model is working through each layer sequentially. The first layer is the input layer. As described before a 32x32 shape with depth one together with a label array is fed to the model. The next layer is a convolutional layer. A kernel size of (2,2) is used for each convolutional layer. The next layer is the activation layer of the convolutional layer. As shown in the Figure 5.3 for this activation, a rectified linear unit (relu) is used. This means that only positive values of the input shape after the applying of the convolution are given to the next layer. The next layer is a max pooling layer. The max pooling layer reduces the size of the input. The pool size is (2,2). The three explained layers are repeated three times. The kernel size and pool size stay the same for these three iterations. However, since max pooling layers are used the size of the aggregation map is reduced. The next layer is a flatten layer. This means that the previously reduced size shape is transformed into a vector. Next a dense layer which resembles a fully connected

layer is used. After the fully connected layer only positive values are evaluated since relu is used as an activation function. The next step is to reduce the connections from the fully connected layer. This is done with the dropout layer. Finally another fully connected layer (dense) is used. For the last layer activation sigmoid is used.

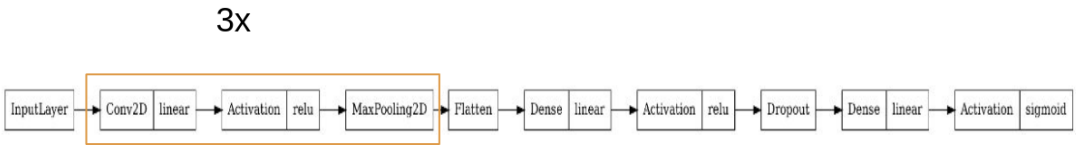


Figure 5.3: CNN Model

5.2.11 Keras Classification

The *keras* model is compiled as described in the above section. *Keras* is implemented to calculate metrics in each epoch. 100 epochs and a batch size of 128 is used for the experiments in this thesis. It is to note that the training input for the *keras* machine learning model is shuffled before using it as training data. Therefore, no patterns of aggregation maps can be detected. The last layer of the model returns a value that describes if the current map is either classified as a port scan or benign network traffic. The sigmoid activation decides if the input is closer to one or to zero. The result is a binary classification. This is done for all aggregation maps. *Keras* automates the process of creating metrics, by returning the confusion matrix values for each epoch. The resulting metrics are used to evaluate the performance of detecting port scans given different inputs defined in the upcoming experiments.

5.3 Experiments

As mentioned in the objective section, the first step is to identify which parameters of the aggregation map need to be optimized for the best performance of detection. This section introduces how the different parameter optimization experiments are set up. Next the experiments to find smaller and hidden ports scans are described. Finally the setup for the CICIDS2017 port scans on Friday are explained.

5.3.1 Implemented Assumptions

To research the different upcoming parameters of an aggregation map, assumptions about port scans were created in the requirements 3.5. An assumption is that the order of ports scanned is random. Tools like *nmap* do not scan ports in order one to 1000, therefore port scan attacks are injected accordingly.
Another implemented assumption is that the attacking node can appear in any bucket of the aggregation map. This means that the training and testing aggregation maps include attacks from 32 different different distributed IP source addresses.

5.3.2 Parameter Optimization

The first experiment is focused on optimizing the parameters of the aggregation scheme. The three parameters are exposure time, resolution and threshold. For this reason the Monday benign traffic is used. First, since the assumption is made that the attack can appear from any IP address, each bucket in a resolution of 32x32 receives an attack. This is example wise visualized in Figure 5.4.

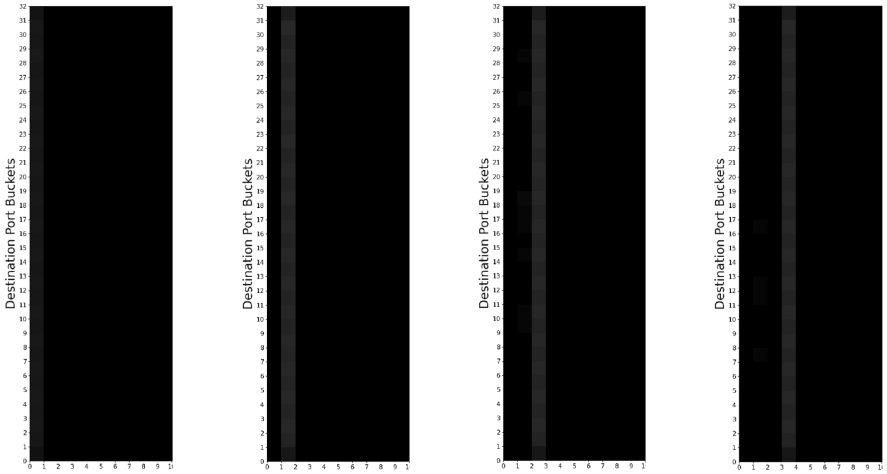


Figure 5.4: Different Source of Attack Learning Data

The first experiment focuses on the main goal to detect straight forward port scans. The port scans in this experiment scan the entire port range of 2^{16} ports. This is done with *IDT2* to the PCAP file of Monday. IDT2 per default scans ports in random order. The next step is to identify the time between each port scan. The Monday PCAP file contains about eight hours of network traffic. This means around 8x60x60 (2880) seconds are available. A port scan that scans the entire port range takes a significant amount of time. Tests with *IDT2* showed the port scans took about 17 minutes. This behavior can appear because port scanners reduce their sent probing packets amount, if no replies are received. To

create the training data the port scans are done every 15 minutes from another bucket. This means there is slight overlap between port scans, but since the aggregation maps data is shuffled this does not mean that there is a fixed pattern in the aggregation maps training data input. The result of this process is a CSV file which contains attacks from 32 different source IP addresses. The source IP addresses are chosen to appear from 32 different buckets. The CSV file is then tested with the different parameters of the aggregation map.

5.3.3 Exposure Time Optimization

The first parameter is the exposure time. The lower bound of the exposure time is defined as the time needed to classify one aggregation map. On the server about 0.15 seconds were measured to classify one map, therefore a range from 0.1 seconds to 60 seconds is used. The upper bound of the scale is limited by the training data. The Monday benign data contains only eight hours which means there are only 480 minutes. This means if more than 60 second exposure times are used, the training data for the machine learning model decreases below measurable standards. Further reduction is expected because of balancing the dataset. To set up this experiment the CSV file is split into the described different exposure times from 0.1 second to 60 seconds. Each time a classification of the given aggregation maps is done over 100 epochs.

5.3.4 Resolution Optimization

The next parameter researched is the resolution of the aggregation maps. The exposure time is fixated to one second for this part of the experiment. The *keras* model needs to be adjusted to display smaller resolutions. It is observable from Figure 5.3 that convolutional layers and pooling layers are used multiple times. If the resolution from the aggregation map starts with a smaller value than 16, this results in the need of padding. For example an 8x8 resolution map cannot be reduced three times with pooling layers (filter size (2,2)). Therefore, only in this optimization process for the resolution, the model is allowed to use zero padding. This means a slightly adjusted CNN model is used for this experiment.

5.3.5 Threshold Optimization

The threshold parameter needs to be optimized since it is important to observe if the classification performance declines with higher thresholds. With a higher threshold less maps are evaluated as port scans since each aggregation map needs to contain a higher amount of packets of port scan packets. This also means that the balance of the input data suffers. If the threshold is higher, less aggregation maps are labeled as port scan. The threshold optimization is done on a resolution of 32x32 and exposure time of one second.

5.3.6 1000 Ports Port Scans

The next experiment is focused on port scans which have a smaller size than the entire port range. Since there are 2^{16} ports and each bucket needs to apply a port scan for learning a port amount of around 1000 is chosen. 1000 ports is also a recommended size described in the *nmap* documentation.

There has to be a trade off between port scanning a high amount of port ranges and the amount of port scans. This is for the reason that port scans have to happen after each other with a realistic time difference. Also they need to appear from different source IP addresses, so the CNN model can learn that a port scan does not only reside from one source IP address. If there is a realistic time spawn between each port scan, the eight hours can not contain complete port range port scans. The time for a port scan of 1000 ports in the CICIDS2017 network traffic PCAP is measured as taking around 20 seconds. This calculation is done by *IDT2*. Therefore, to reduce overlap a port scan is done every 30 seconds. The total amount of usable seconds is around 28800 (eight hours). If a port scan happens every 30 seconds a reasonable amount of port scans can be done. In the experiment each of the 32 attackers from the different source IP addresses does seven port scans. This trade off resembles more realistic smaller port scans that do not contain the entire port range of 2^{16}.

5.3.7 Stealth Port Scans

Stealth port scans in this experiment are expected to scan a small range of ports over long time durations. Therefore, in this experiment only three ports are scanned from each bucket. This is done over a relatively high amount of time. The time fixated in this experiment is one minute between each port scan. This results in only 480 port scans. Each bucket is applying 15 port scans on different ports. This concludes that also in this experiment not all ports are scanned. The experiment is set up to display a patient attacker which scans specific services. This experiment is given to show the limits of the aggregation scheme based detection. A careful attacker should not be detected with this small number of ports scanned.

5.3.8 CIC Port Scans Balanced vs Unbalanced

The CICIDS2017 dataset in this experiment is reduced to the relevant minutes, where the port scan's take place. This resembles the time duration where the firewall is off. The UTC timestamps are from 17:50 to 18:30. In this time duration the CICIDS2017 dataset includes a variety of different port scans. A detailed list of all the attacks done in the dataset is displayed in Figure 5.5. The table is created from the website of CICIDS2017 and the port scan types are evaluated given the abbreviation provided by the authors.

Port Scan Type	Time (UTC Timezone) on 7.7.2017	Abbreviation
TCP SYN	17:51-17:53	sS
TCP Connect	17:54-17:56	sT
TCP FIN	17:57- 17:59	sF
TCP XMAS	18:00-18:02	sX
TCP Null	18:03-18:05	sN
Protocol Scan?	18:06-18:07	sP
Version	18:08-18:10	sV
UDP Scan	18:11-18-12	sU
IP Protocol Scan	18:13-18:15	sO
TCP ACK	18:16-18:18	sA
TCP Window	18:19-18:21	sW
Resolve Host?	18:22-18:24	sR
List Scan	18:25-18:25	sL
Min Rate Scan	18:26-18:27	sI
FTP Bounce Scan	18:28-18:29	b

Figure 5.5: CIC Attack Table

As observable from the table 5.5 there are port scan types which are not part of the objective in this thesis. Further details are given in the evaluation section 6.9 concerning this experiment. The dataset on Friday differs in comparison to the Monday benign traffic in already having attacks, and also in the victim system possibly having non attack network traffic. In the previous experiments the labeling is done exclusively by identifying the victim system and all traffic sent in the direction of the target is labeled as port scan. In this experiment the labeling is done attacker to victim based. This means only packets which are sent from the attacker node to the victim node are labeled port scan. In addition only the time frame where the attacks take place is considered for evaluation. In conclusion the dataset contains a mix of straight forward non-stealth port scans and hidden port scans.

This experiment, besides showing that the CIC port scans can be detected, is used to show the difference in balanced and unbalanced training and testing data. For this reason classification is done ten times on balanced and ten times on unbalanced aggregation maps. The amount of ten times is used to check how much the classification results fluctuate. The aggregation map parameters are: one second exposure time, resolution of 32x32 and threshold of one. By using an exposure time of one second, the 40 minutes of port scan behavior resemble 2400 aggregation maps (40*60). When balancing the dataset it was observed that only 243 aggregation maps out of 2400 are labeled as containing a port scan. This means that the machine learning model gets fed about 500 aggregation maps. When using the unbalanced aggregation map it is to note that the accuracy metric loses value. This is for the reason that with a validation split of 20 percent it can not be determined how the distribution of port scan aggregation maps and benign aggregation maps are in the testing data. Therefore, to compare the balanced vs unbalanced classification results, the precision and recall are of higher value. Precision and Recall are calculated with comparison to true positives. Also the false negative negative rate describes the possible to find port scan aggregation maps more precisely. The false positive rate can also be used to evaluate the performance, since it does not rely on the distribution of port scan and benign aggregation maps.

5.3.9 Implementation of Metrics

Machine Learning results can vary strongly due to their random initial weights. Therefore, the results of classification can fluctuate similarly. For this reason, the experiments besides the parameter optimization are run ten times. The fluctuation can be observed by min and max shading in the plots.

6. Evaluation

The chapter evaluation is given to introduce the results, from the experiments described in the previous implementation chapter. Also interpretation of the performance metrics are described.

6.1 Parameter Optimization Experiments

The first experiment as described in implementation is done to explore the optimal parameters of the aggregation maps. For this reason each parameter is evaluated by using different values for the range of possible parameter values. The metrics shown in this section are created by using the x-axis to show the parameter change impact.

6.2 Experiment Exposure Time

The first experiment shows the different possible exposure times that aggregation maps have been created with. As explained in the implementation section 5.3.3 the minimum exposure time is corresponding to the time the classification process needs. Since the time was measured as 0.15 seconds, the range starts with 0.1 second's and ends with one minute. In the Figure 6.1 the accuracy, precision and recall are shown. These values are created by taking the maximum value over 100 epochs. The x-axis shows the different exposure times and the y-axis the metric value. As observable by the y-axis range the classification performance is always over 88 percent. The Figure describes the maximum value for the metrics over the different exposure times. This is done for the reason that each exposure time describes 100 epochs. At the start of the epochs the performance metrics are lower meaning that the maximum is the best value achieved over 100 epochs.

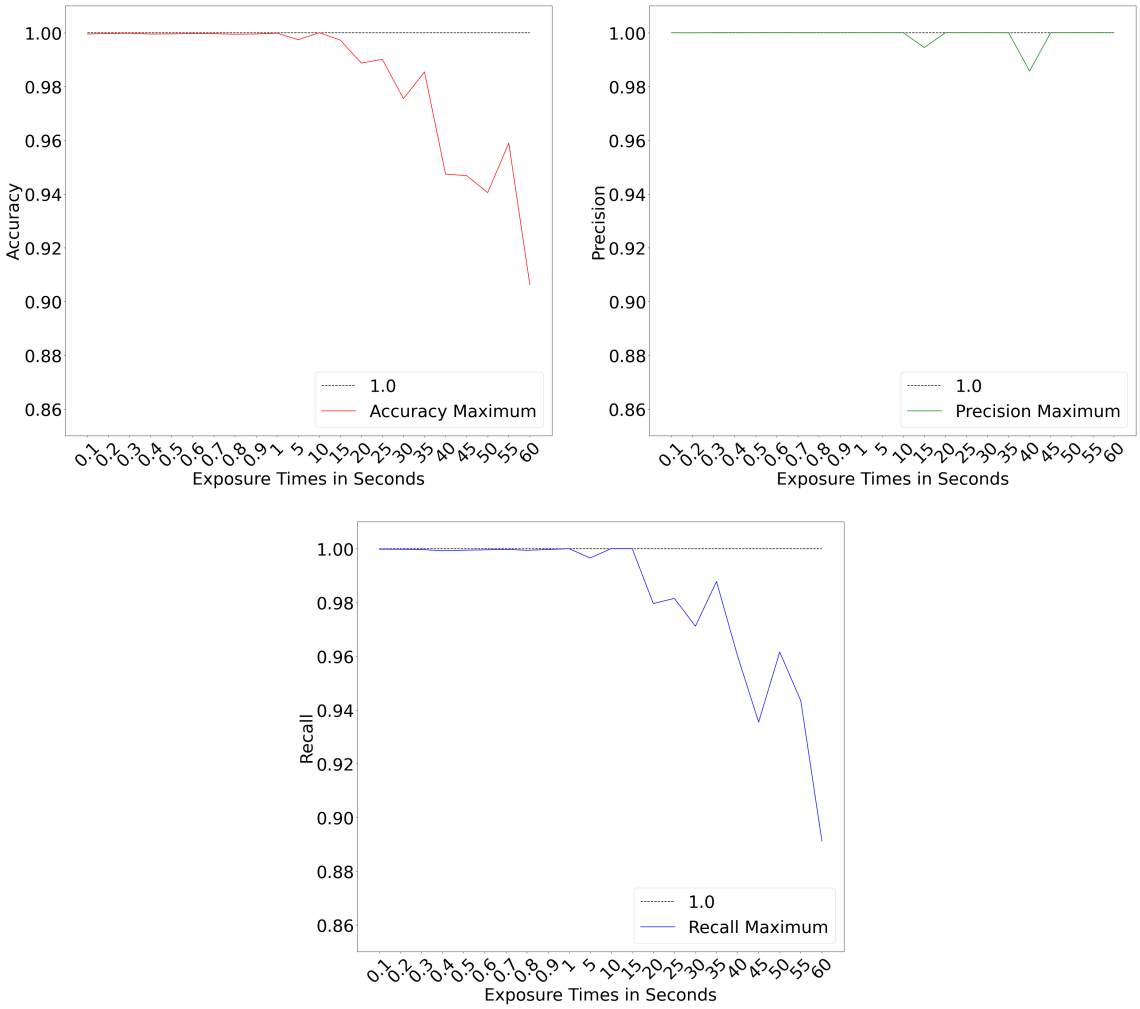


Figure 6.1: Performance Metrics for Exposure Time

However, the accuracy is declining after an exposure time of 15 seconds. This is partly due to the imprecision of the machine learning model. Another possibility is that the total amount of aggregation maps that are provided for learning is decreasing (Figure 6.2). Since there are only eight hours the amount of maps decreases with every tick on the x-axis. This results in a lower accuracy since the relative amount of wrongly classified aggregation maps is increasing. It can also be observed that the total amount of false positives and false negatives is not increasing. This creates the assumption that the machine learning model does not classify worse because of the quality of the aggregation maps. The precision metric supports this assumption, because if the aggregation maps would be easier to classify as false positives the precision would decline.

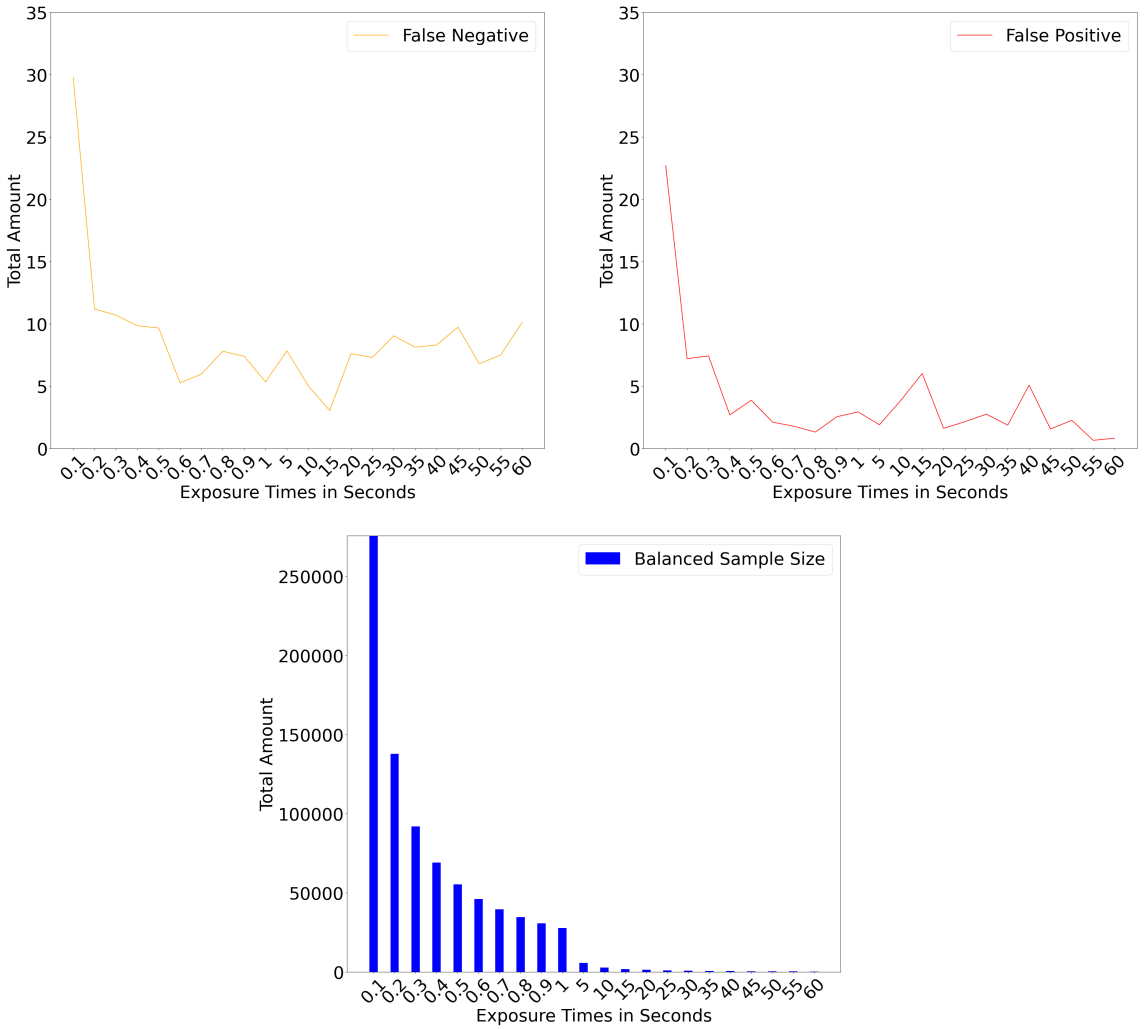


Figure 6.2: Total Amounts of False Positives and False Negatives and Samples

The precision metric shown in 6.1 is relatively unaffected by higher exposure times. The precision shows how many port scans are classified correctly in comparison to how many false positive port scans got classified. This metric indicates that increasing the exposure time does not introduce false positive problems. The total amount of false positives is not increasing even though the exposure time is strongly increasing and the samples are reduced. Relatively seen however the false positives are growing. This can also be observed in Figure 6.3, in difference to the false negative rate, the false positive rate is staying below 20 percent.
Next the recall is as shown in Figure 6.1 as declining after 15 seconds. Recall describes how many of the total possible port scans got detected. The recall loses performance strongly with higher exposure times. This leads to the assumption that the port scans which could have been detected with lower exposure times, get ignored more with longer exposure time. At one minute only about 89 percent of port scans get detected. This supports the previous assumption that if more background traffic is within one aggregation map the map appears to be clustered for the machine learning model.

A more detailed look is taken with the totals of false positives and false negatives. The y-axis ranges from zero to fifty. In the Figure 6.2 on the right the amount of aggregation

maps is described. Since the complete dataset contains only eight hours with each exposure time extension the total amount of aggregation maps is reduced. Another factor is that if balancing is used less aggregation maps can be considered. Aggregation maps can contain multiple port scans with higher exposure times. The main observation in Figure 6.2 is that the total amount of false positives and false negatives are decreasing or staying equal with higher exposure times. The lower bound in this case means that absolute amount of false negatives as well as false positives are the highest at 0.1 seconds of exposure time. The observation made is that even though the training data is significantly smaller this even decreases the false positives and false negatives.
The false positive and false negative rate can be observed in Figure 6.3. Since the false

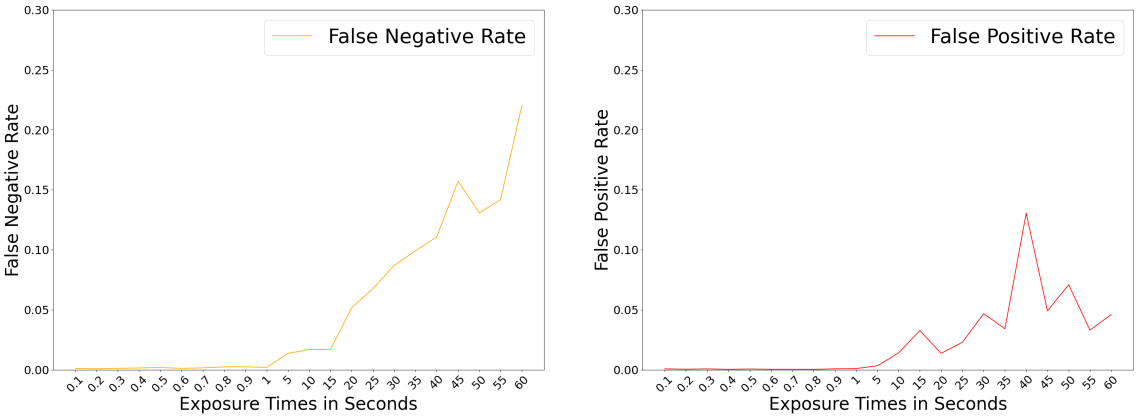


Figure 6.3: Relative Performance for False Negatives and False Positves

negative rate is increasing as shown in Figure 6.3, this metric also suffers from higher exposure time. This means that the amount of port scans that could have been detected is decreasing relatively. The assumption drawn from this, is that with higher exposure time the actual port scans are not detected as well as with lower exposure time. The recall from Figure 6.1 supports this as well. A conclusion can be drawn that a higher aggregation exposure time results in a low amount of false positives but with one minute the success rate of detecting port scans is at around 88 percent. However, since there are only eight hours of traffic available this conclusion can not be proven. In conclusion this means that longer exposure times result in slightly worse performance. The source of the problem is not clear since the absolute amount of false positives and false negatives decreases. The amount of undetected port scans increases faster than the amount of background traffic classified as port scan.

6.3 Experiment Resolution

The next parameter of the aggregation map that is researched is the resolution. The resolution maximum is derived from the amount of TCAM rules. The maximum amount is 32x32 since otherwise a to high amount of TCAM rules would be needed.

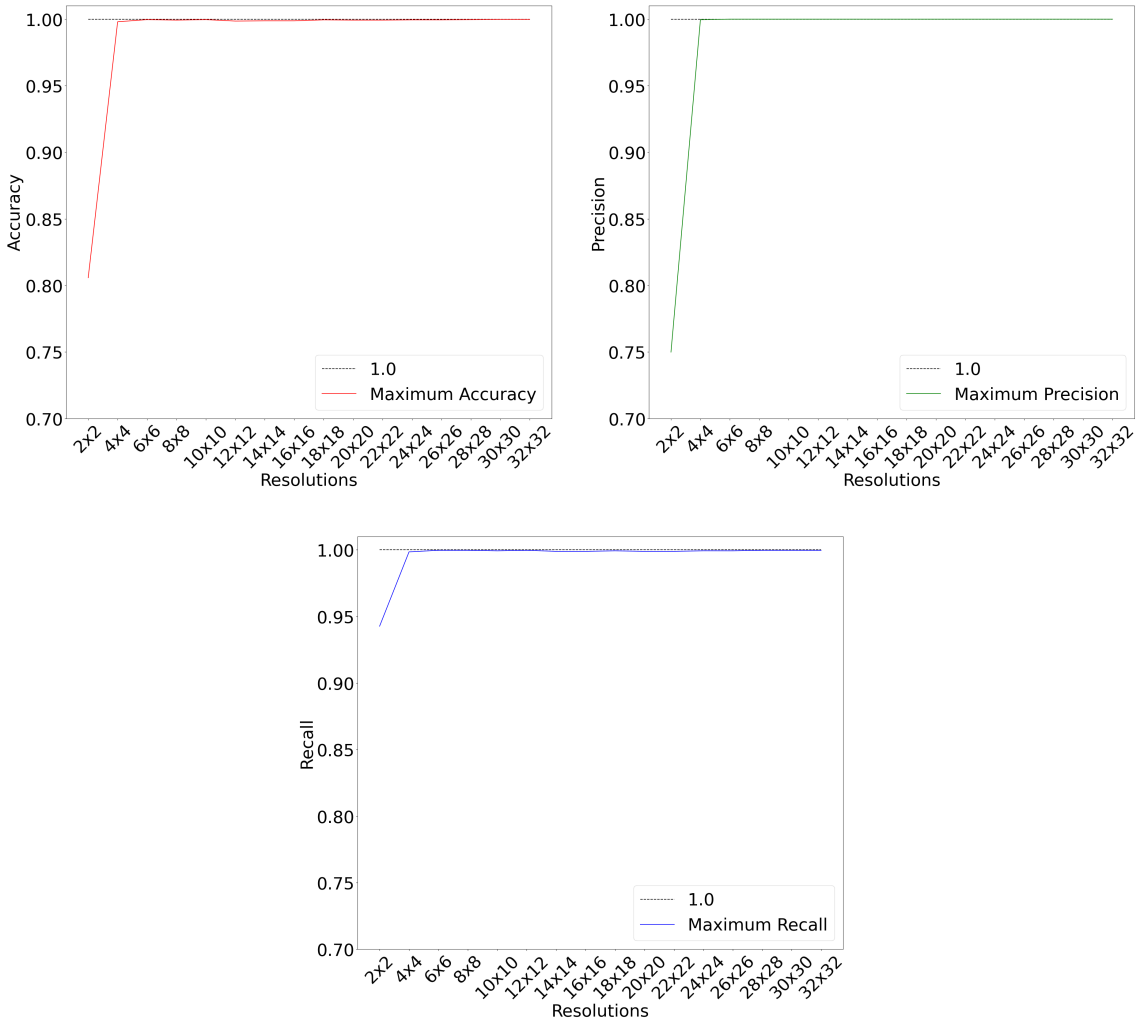


Figure 6.4: Performance Metrics for Resolutions

This experiment focuses on the necessity of the amount of TCAM rules. In Figure 6.4 the overall performance can be observed. The x-axis shows different resolutions. The y-axis describes the metric value for accuracy, precision and recall. The y-axis is starting at 70 percent. This shows that the performance with resolutions higher than 4x4 is near optimum. For this experiment one second exposure time and a threshold of one is used. As mentioned in the implementation section zero padding is needed for the lower resolution. It can be observed that this does not reduce the performance when the resolution is higher than four. The conclusion drawn from this experiment is that even with a small resolution the normalization processing of the images seems good enough to classify very precisely. This is likely due to the burst behavior of port scans. Since a port scan has a high amount of packets over a short amount of time the aggregation map has identifying values that create high performance. The main result in this experiment shows that way less than 32x32 buckets are needed to identify port scans.

6.4 Experiment Threshold

The threshold parameter performance is shown in Figure 6.5. The x-axis resembles the different thresholds tested. The y-axis ranges from 98 to near 100 percent. The metric shows that there is nearly no performance loos by increasing the threshold. It is to mention that in this experiment the port scans are done over the full range of 2^{16} and exposure time of one second is used. Finally after the threshold 200, there is no classification possible. The reason is that no labeled aggregation maps could be created. The next threshold step in the experiment was 400. Therefore, the amount of port scan packets within one labeled aggregation map is between 200 to 400 at one second exposure time. This shows that with this exposure time 200 to 400 packets of port scans are within each aggregation map. The main goal of this experiment was to determine if a higher threshold introduces classification problems. As shown in the Figure 6.5 classification problems do not appear with higher thresholds, as long as they are lower than 200.

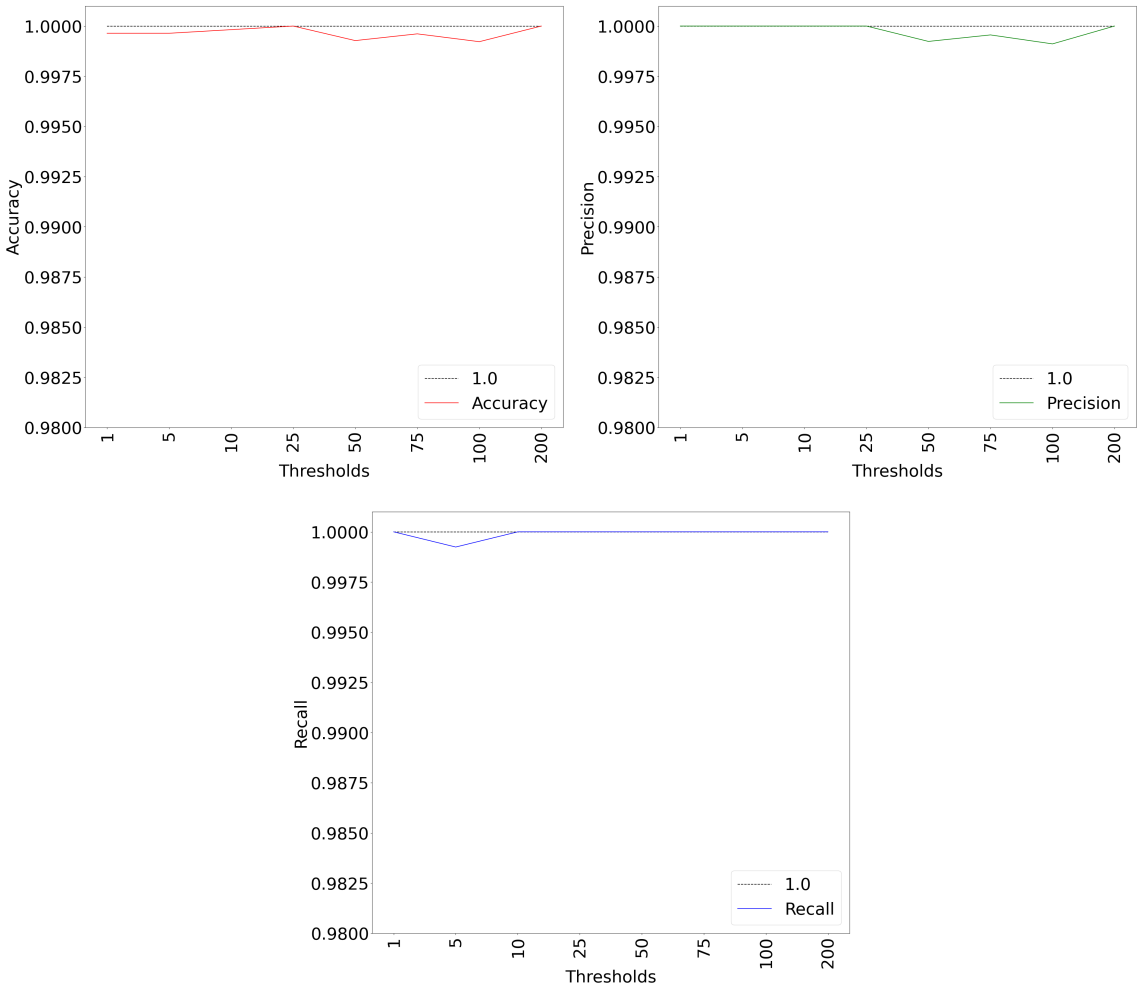


Figure 6.5: Performance Metrics for Thresholds

6.5 Experiment Complete Port Range Attacks

The experiments for exposure times have shown that near optimum performance can be achieved with small exposure times. In Figure 6.6 the performance is shown at one second exposure time, with a resolution of 32x32 and a threshold of one. The y-axis starts at 90 percent. It can be seen that the accuracy, precision and recall are near their optimum, at the given parameters for the aggregation map. It is to note that this experiment was done on a balanced aggregation map input. Also the fluctuation is low. The precision and accuracy fluctuate a bit but not lower than 95 percent. The measured highest accuracy is 0.99965, the highest recall is 0.99963 and the maximum precision is 1.0. For this reason it is derived that an exposure time of one second with resolution of 32x32 and threshold of has a high performing detection for synthetic created port scans. For the upcoming experiments the same parameters are chosen since they resemble a good enough trade off between a higher amount of aggregation maps and detection performance (a shorter exposure time might be beneficial because more aggregation maps are created, but the performance at one second is sufficient).

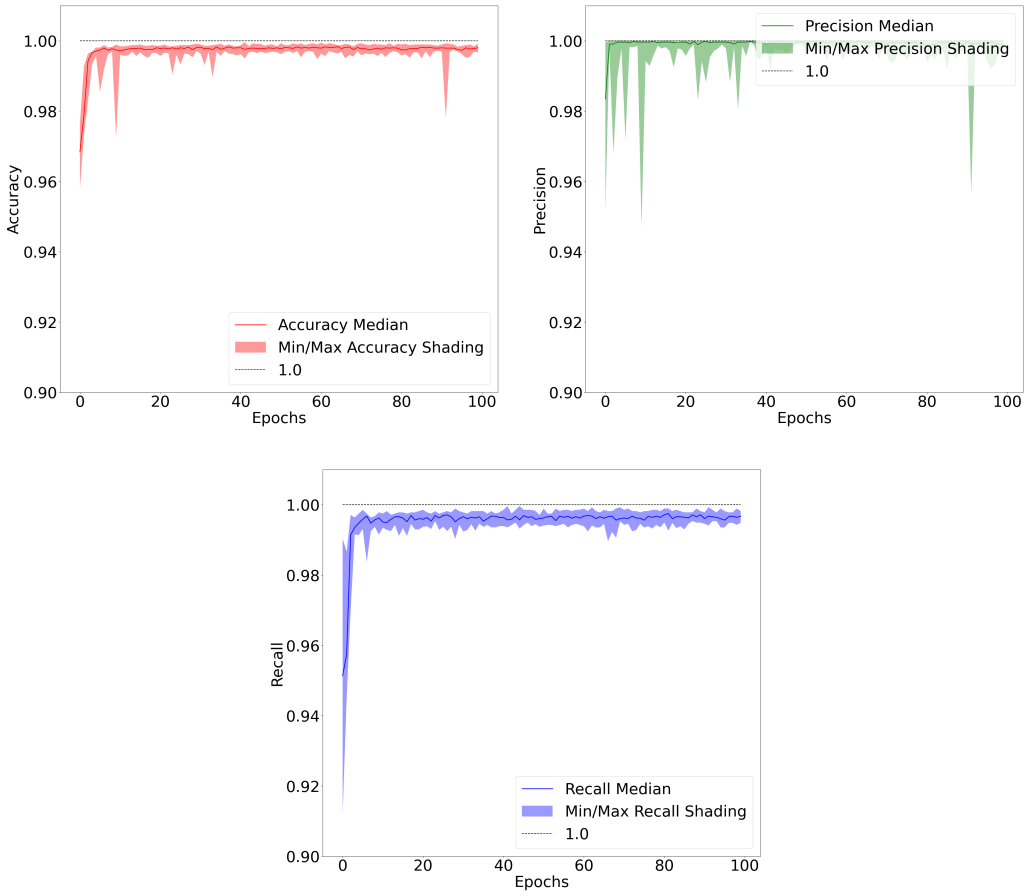


Figure 6.6: Complete Port Range Classification Performance

The Figure 6.6 shows in every observed metric a near ideal metric value. The accuracy shows that nearly no aggregation maps got classified wrong. The recall shows that almost all possible port scans were detected. And the precision shows that while detecting the port scans nearly no false alarms got created.

6.6 Complete Port Range Attacks Training Performance

With the same experiment from before the training performance is explained. In Figure 6.7 the loss and accuracy during training are shown. Loss represents the sum of errors the machine learning model does. It can be observed that the loss reduces strongly with more epochs. The accuracy is near optimum after approximately ten epochs. A low loss function during training means that the optimal parameters for classification have been determined. The *keras* model compiles a scalar loss. Meaning that the predicted results are compared with the given labels for the loss value. In the experiments the training accuracy and loss are near optimum for nearly all experiments in this thesis. The exception being the experiment to test the limits of the detection, with only three ports scanned (section 6.8).

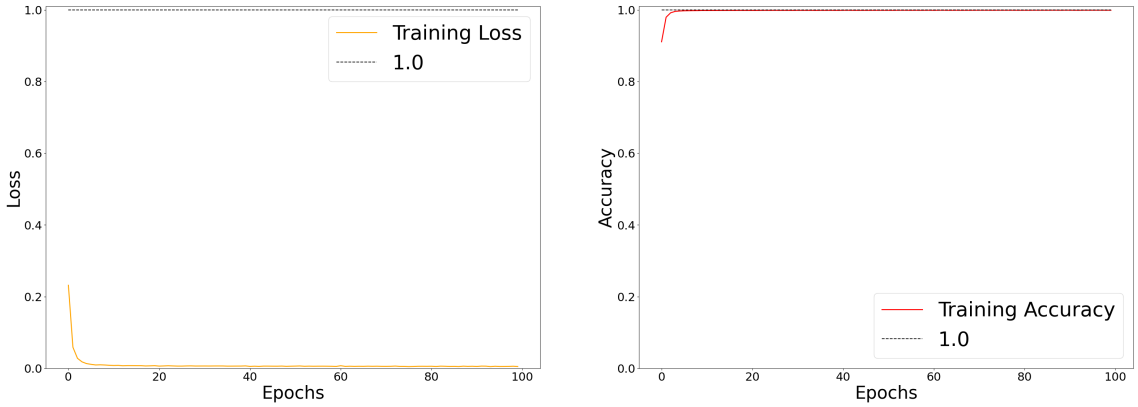


Figure 6.7: Complete Port Range Classification Training Performance

6.7 Experiment 1000 Ports Port Scan Attacks

In the 1000 Ports port range experiment the evaluation is done on how well the classification works over epochs. For this reason the classification is done ten times to ensure quality of the metric measurement. The median results are plotted in Figure 6.8.

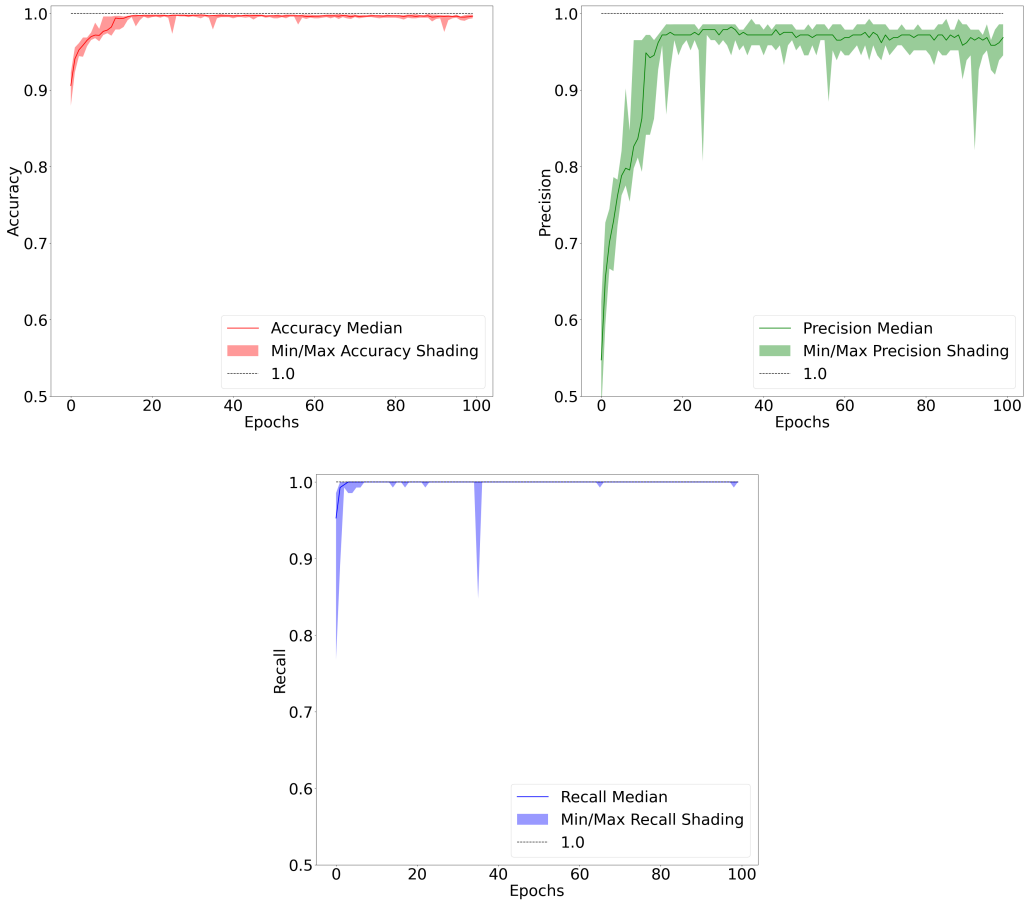


Figure 6.8: 1000 Port Range Classification Performance

The plot also includes shading for the minimum and maximum for each epoch that the different classifications took. The used parameters of the aggregation map are one second exposure time, a resolution of 32x32 and a threshold of one. The y-axis ranges from 50 to near 100 percent. The accuracy after about 20 epochs is consistent, this identifies that the overall correct classification has a high success rate. The precision has a high score, however deviation is possible as shown from the shading. This shows that false positives can occur. Deriving from the median being stable this means that the classification in some of the ten runs creates a larger amount of false positives. The precision as shown in Figure 6.8 performs the weakest. The recall has low min and max fluctuation over the entire epochs. This means that the CNN model does not have different results with more epochs. A high recall means that almost no port scans are undetected. The accuracy is near optimum with almost no fluctuation which further contributes to the conclusion that smaller port scans of 1000 ports scanned can be detected with a high chance.

Another aspect are the false positive and false negatives rates. False negative means a port scan is not classified correctly by the machine learning model. In Figure 6.9 the rates medians are displayed. Also shading for min and max fluctuations are shown. The y-axis starts from zero and foes to 30 percent. The false positives rate needs to be near zero to assert that the detection alarm can be trusted. As shown in the Figure 6.9 there is no issue with false positives.

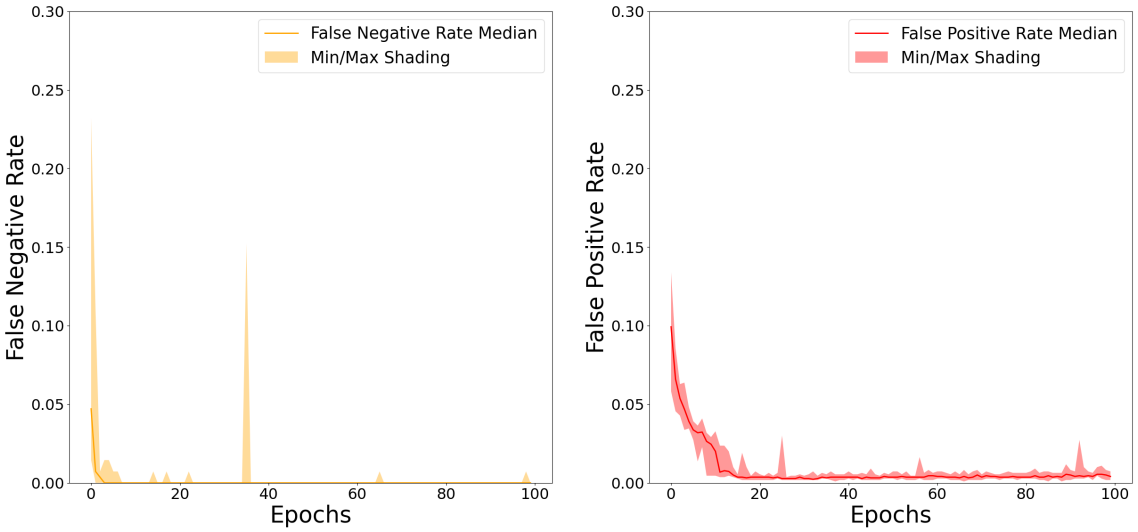


Figure 6.9: 1000 Port Range Classification Performance

It is to note that the total amount of aggregation maps in this experiment is strongly reduced compared to the previous experiment. With less port scans over eight hours that scan less port ranges the resulting data contains about 6000 aggregation maps. This is partly due to the balancing of the aggregation maps. A summary of this experiment is that the machine learning model predicts false positives slightly more than ignoring port scans when it comes to port ranges that scan 1000 ports. However, the overall performance after 100 epochs is near optimum as shown by the performance metrics.

6.8 Experiment Three Ports Port Scans

The experiment with only three ports scanned was designed to show the limits of the aggregation scheme based classification. The Figure 6.10 shows that the accuracy in comparison to the previous experiment is fluctuating way more. Also the overall accuracy is reaching only about 80 percent in the median metric. Since this experiment has only a small amount of aggregation maps (578) this is partly a problem with the training data. The precision median is consistently under 20 percent. This means that the amount of false positives is significantly higher than the amount of correctly identified port scans. This is understandable since the aggregation maps which are labeled as containing port scans only contain three packets that are supposed to display a port scan. This leads the CNN model to take different features of normal aggregation maps and explains them as port scans. The recall fluctuates strongly, this is due to the fact that the amount of port scans that have not been detected varies strongly for the different classification rounds. The CNN model has to guess if an aggregation map contains a port scan. The training aggregation maps did not contain features that help in identifying aggregation maps.

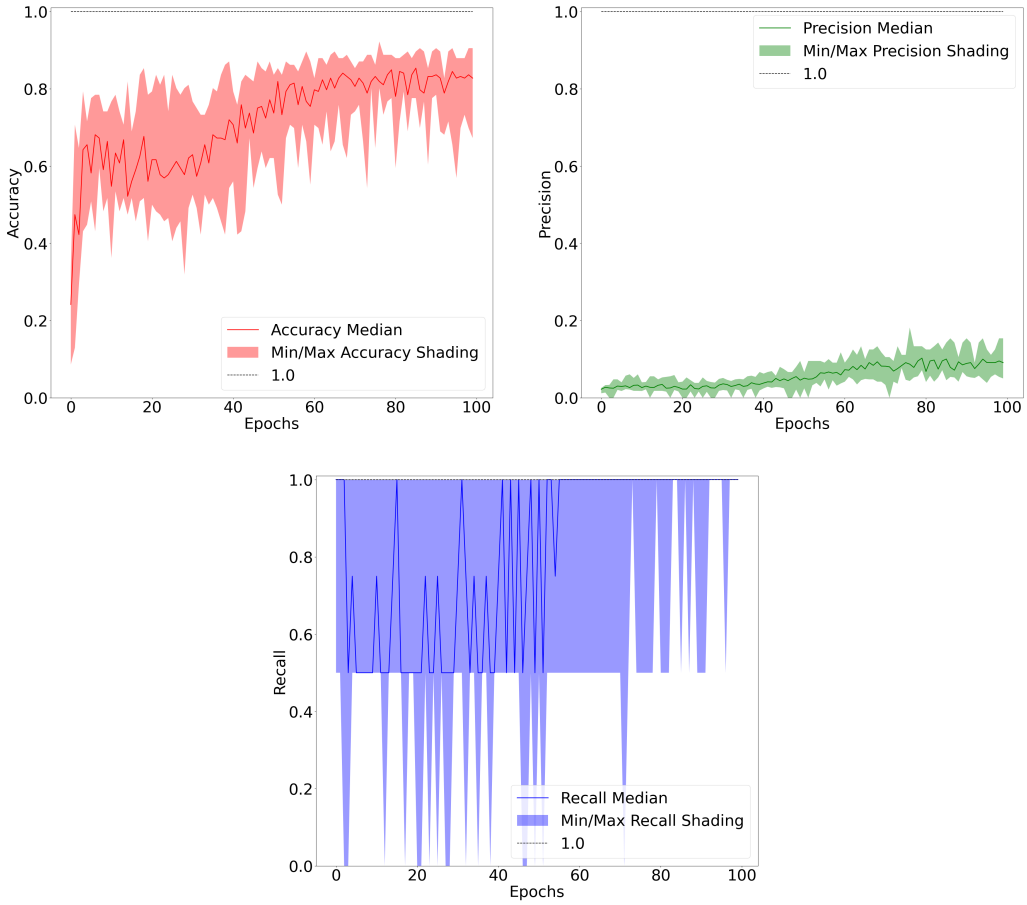


Figure 6.10: Three Ports Port Scan Classification Performance

Next the false negative and false positive metrics are shown in Figure 6.11. The false negatives fluctuate strongly since the machine learning model in some cases does not detect any port scan. The median for the first 55 epochs is about 50 percent which means the machine learning model has to guess if a port scan took place or not. The conclusion drawn from this is that when only three ports are scanned the machine learning model is not able to identify them as a port scan. Since the CNN model learns in every epoch, the false positive rate is slowly declining. However, even if the machine learning model can not identify port scans, it more often tends to classify benign traffic as port scan. This does not apply for the other experiments, since they do not learn on training data which does not properly show a port scan.

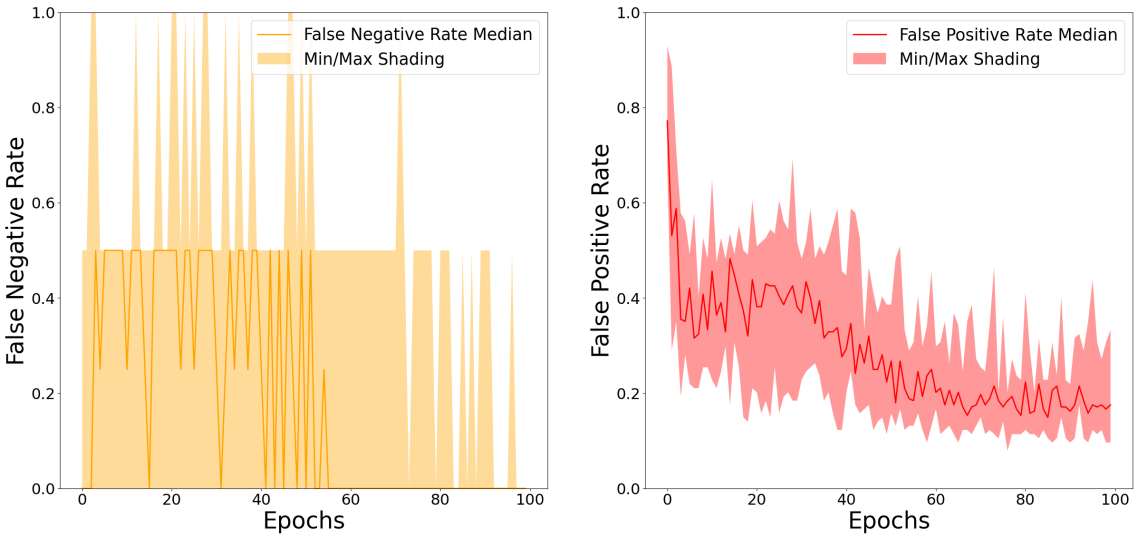


Figure 6.11: Three Ports Port Scan Classification Performance Rates

The results of the experiment is that port scans with only three ports scanned can not be reliably predicted by the CNN model.

6.9 Experiment CIC Port Scans Balanced vs Unbalanced

To show the overall performance of detecting port scans within the CICIDS2017 dataset the balanced experiment accuracy is presented in Figure 6.12 on the left. The y-axis ranges from zero to 101 percent. The maximum accuracy over all epochs is 0.92857. The precision in 6.13 has a maximum of 100 percent, but overall the fluctuation is higher than the accuracy. The median of the precision is around 91 percent. This means that while classifying the true positives only a few benign aggregation maps are classified falsely as port scan. The recall in Figure 6.14 fluctuates the most compared to accuracy and precision. In some exception epochs the recall is 100 percent meaning that all port scans were detected. The recall median slightly improves over epochs to about 80 percent. This means that every second a port scan takes place, there is 80 percent chance to detect the port scan.

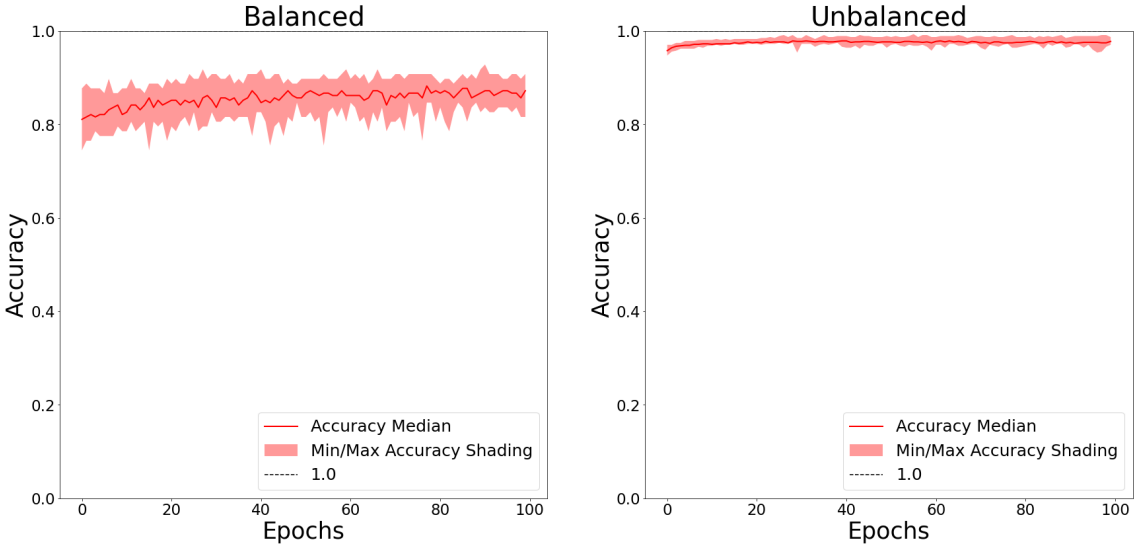


Figure 6.12: Balanced Aggregation Maps vs Unbalanced Aggregation Maps Accuracy

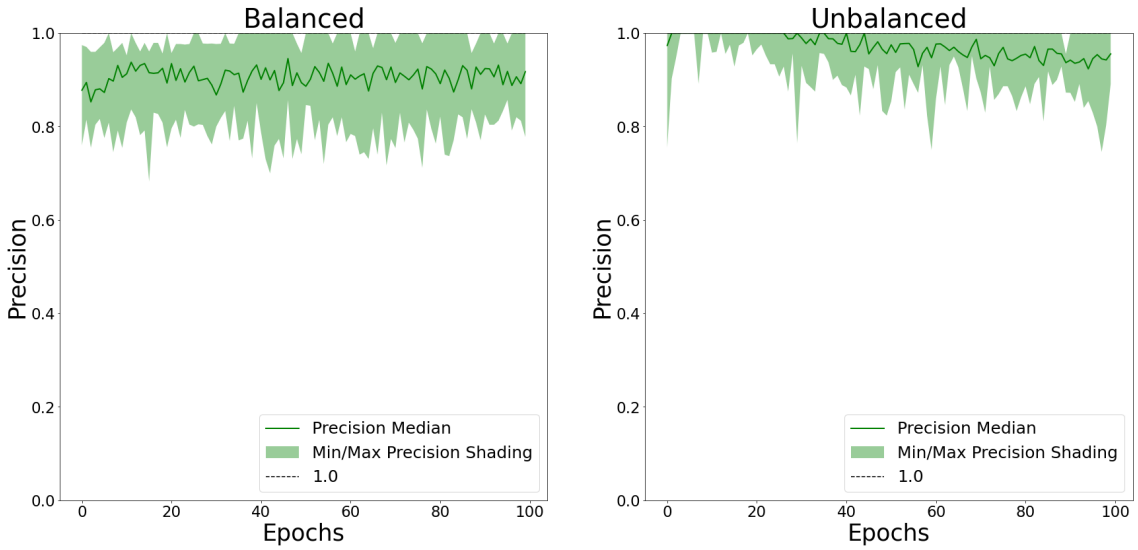


Figure 6.13: Balanced Aggregation Maps vs Unbalanced Aggregation Maps Precision

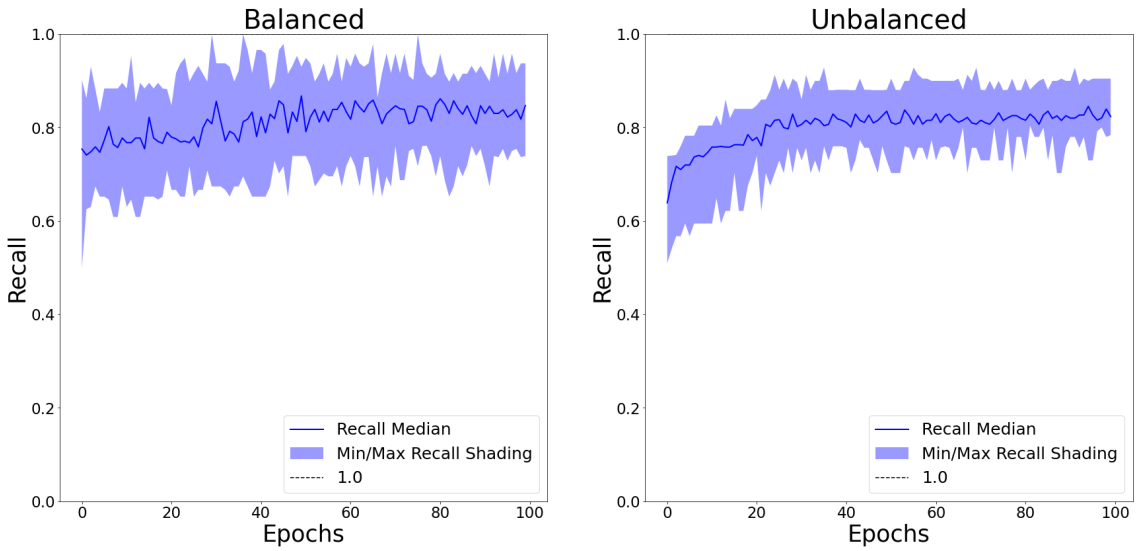


Figure 6.14: Balanced Aggregation Maps vs Unbalanced Aggregation Maps Recall

Next, the unbalanced results are shown in the same Figures on the right. It can be observed that the accuracy improves the most. However, since the testing data distribution of port scans and benign aggregation maps is unclear the accuracy can not be evaluated alone. The precision declines, but overall is higher than the precision from balanced aggregation map experiment. In this experiment a higher amount of aggregation maps is used than within the balanced experiment. Around 2400 aggregation maps are used with unbalanced, since no aggregation maps need to be filtered. Next, the recall shows that amount of possible port scans that got detected is similarly high as with the unbalanced aggregation maps. However, the fluctuation declines slightly.

The precision decline can not be explained with the previous Figure, therefore the false positive rates for the balanced and unbalanced aggregation map classification is shown in Figure 6.15.

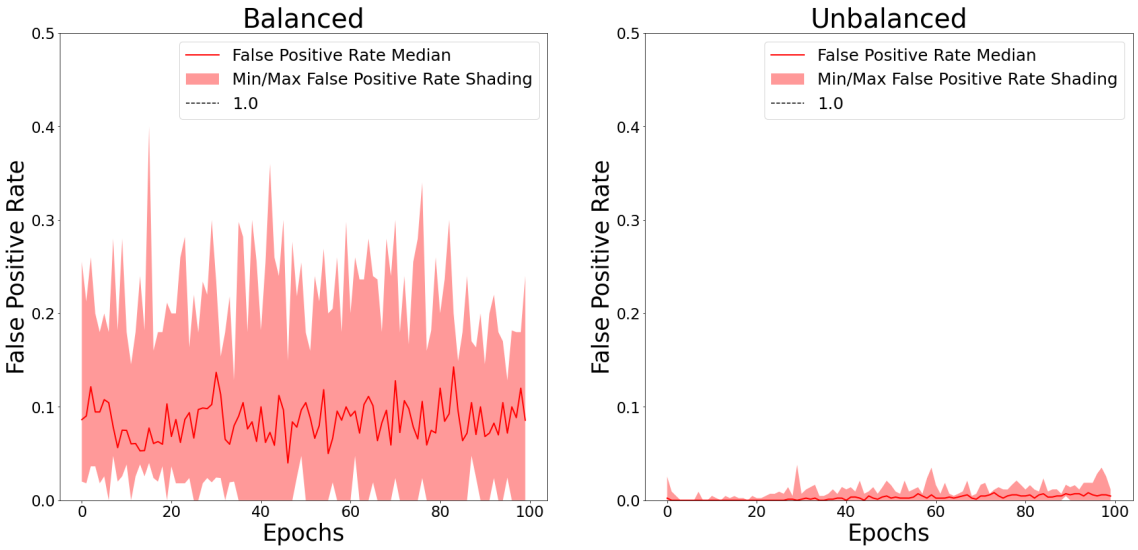


Figure 6.15: Balanced Aggregation Maps vs Unbalanced Aggregation Maps FPR

The y-axis in these figures ranges from zero to 50 percent. In the right unbalanced graph it can be seen that the false positive rate is near zero percent, however a small fluctuation occurs. The result of this graph is that the false positives do not seem to be a problem when using an unbalanced dataset. On the left in Figure 6.15 the false positives rate can be seen for balanced aggregation maps. In comparison to the unbalanced aggregation map classification it can be observed that the false positive rates fluctuate more with a balanced training approach. A possible conclusion is that the smaller amount of aggregation map is a reason for this fluctuation. However, this metric is set relative to the total possible benign classifications. Further, it can be understood that the machine learning model needs to learn background traffic. The main difference between the two classifications is that the unbalanced aggregation map dataset contains significantly more background traffic aggregation maps. The result being that it does not classify background traffic as port scan as much as the balanced aggregation maps.

The next finding is presented in Figure 6.16 which shows the false negative rate metrics for balanced and unbalanced aggregation maps. In this Figure on the left the false negative rate for balanced aggregation maps is shown. First it can be observed that it fluctuates slightly more than with the unbalanced aggregation maps. However, the median is similar to the unbalanced classification performance, with more epochs. This means that with more epochs the ignored port scans are only about 20 percent in both cases. Since the classification is done every second, this means that a port scan can be detected when the port scan, has a time of execution longer than a few seconds.

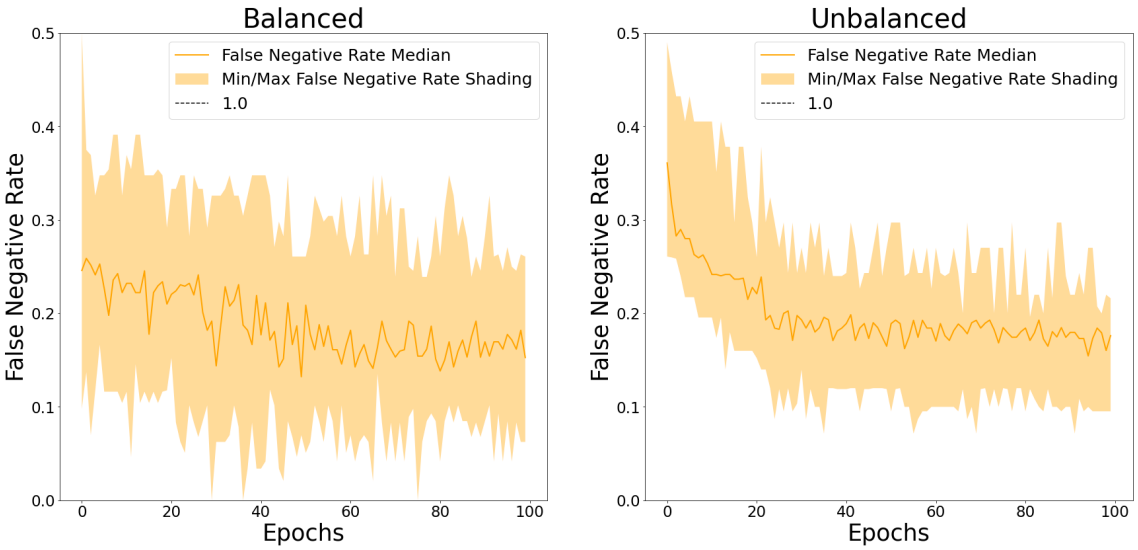


Figure 6.16: Balanced Aggregation Maps vs Unbalanced Aggregation Maps FNR

6.10 Evaluating Requirements

This section introduces the success of the detection concerning the defined requirements in the analysis section 3.5 (Reaction Time, Recall, False Positive Rate). Further it discusses the results compared to related work.

Reaction Time

The reaction time is depending on the exposure time used with the aggregation maps and the classification time from the CNN. For optimizing the reaction time aggregation maps with an exposure time of 0.15 seconds should be used. On the server used in this thesis a prediction time of 0.15 seconds was measured. This means that every 0.15 seconds an aggregation map can be predicted by the machine learning model. With TCAM rules the creation time for an aggregation map is within one processor clock cycle. The created aggregation map then can be predicted with the trained CNN model. This requires 0.15 seconds plus the processor clock cycle. In the exposure time experiment Figure 6.1 it can be observed that the accuracy is near 100 percent at 0.1 seconds till five seconds. The high accuracy continues till up to five seconds. Therefore, it can be estimated that the accuracy is near optimum at the exposure time of 0.15 seconds.

Next the reaction time is compared to [1]. In [1] also the CICIDS2017 dataset is used. This means the same port scan attacks are classified as in this thesis. In [1] support vector machine and deep neural network classification is used. The approach in [1] uses the 85 features provided by the CICIDS2017 dataset. Therefore, it is classified as a micro-flow detection approach. This means each feature needs to be calculated when the network traffic flow ends. This means that flow completion needs to finish till classification by the machine learning model can be done. From measuring the CICIDS2017 dataset the following flow durations were calculated. The value for average flow durations is approximately 5379330 micro-seconds when using all flows in the Friday port scan CSV provided by CICIDS2017. This means that a flow created in the CICIDS2017 dataset can further be processed after 5.37 seconds. In [1] 85 features need to be calculated after the flow ended.

In comparison the reaction time in this thesis is only 0.15 seconds plus processor clock cycle. The processor clock cycle in most cases can be neglected since with a GHz processor one clock cycle resembles nano-seconds. This assumes TCAM rules for aggregation map creation. Therefore, the approach presented in this thesis is significantly faster.

Recall

A high recall is important for detecting port scans when they occur. From the experiments on balanced and unbalanced aggregation maps it can be seen that the recall is slightly higher with balanced datasets. The best recall over epochs can be observed in the 1000 ports scanned example. The recall is near 100 percent for the entire experiment. This means nearly no port scans are missed when it comes to non-stealth port scans.

The recall in related work is higher when it comes to the CICIDS2017 dataset. In [1] the macro flow detection has a recall of 99 percent. However, they need to wait for flow

completion. In the approach presented in this thesis, the port scan will be detected with a chance of over 80 percent 6.14. This is used with a balanced aggregation map set over only the port scan section. Technically, these are the same port scans as in [1]. A port scan can take several minutes. Each second as shown in 6.14 the detection chance is over 80 percent. It can be assumed that a port scan with a duration of several minutes will be detected.
The conclusion drawn from this is, that the recall might be slightly lower than in related work. However, the actual detection of a complete port scan is similarly high and faster when it comes to reaction time.

False Positive Rate

A small false positive rate needs to be achieved, otherwise the detection alarm can not be trusted. In the experiments it is observed that background traffic training is important for the false positive rate. For background training an untouched distribution of labeled and unlabeled aggregation maps was used for better results. It can be seen from 6.15 that a false positive rate from nearly zero percent can be achieved. However, a larger amount of background traffic is necessary for this small false positive rate.
It can be concluded that with large amounts of background traffic as training data the occurrence of false positives can be neglected.

Since in related work, the metric for false positives is not described clearly, the precision can be used to compare results. However, precision only focuses on the amount of predictions made by the machine learning model. In [22] the precision is nearly 99 percent. The research done in [22] is a flow based approach. It can be observed that the precision in the research in this thesis is not as high. At least not when using multiple detection attempts. The highest precision when it comes to the CICIDS2017 dataset is shown in Figure 6.13. The precision here per second is around 95 percent after 100 epochs.

Small Port Range Port Scan Detection

One requirement is that port scans that are declared as normal can be detected. Normal in this specific case means that the attacker does not specify the ports to be scanned. In *nmap* this results in the first 1000 ports scanned. This port scanning behavior is expected to be slightly harder to detect than a port scan scanning the entire port range of 2^{16}. However, the experiment 6.7 shows that these port scans can be detected with a high chance.

6.11 Summary

In this section the main results from the combined experiments are given. The first experiments focus on optimizing the parameters for the aggregation map. The parameters are exposure time, resolution and threshold.
The first parameter exposure time has the most impact on the classification performance. It can be observed that with higher exposure time the falsely classified background traffic is not increasing significantly when observing totals. The undetected port scans grow with higher exposure time, the assumption is that aggregation maps get clustered. However, even at the highest exposure time the success rate of classifying port scans is at around 88 percent according to the recall metric.
The resolution experiment identifies that even with small aggregation maps the detection performance is nearly a hundred percent accurate. Even though resolutions, which are lower than 16x16 need to be zero padded, the detection performance only declines with a 2x2 resolution.
The threshold experiment shows that even though a high threshold is used the detection performance is not declining. The main result is that at one second each aggregation map that should be classified as port scan contains between 200 to 400 port scan packets. Further smaller port scans got researched. First port scans that only consist of 1000 ports scanned were researched. With a balanced dataset and exposure time of one second almost all port scans are detected as proven by the recall. However, in this experiment the precision is reduced, meaning that more false positives are created.
The next experiment provided are port scans that are supposed to be undetectable. They are done over a higher time duration, while there are only three ports scanned. The results show that the wrongly predicted port scans increase extremely. And the recall is fluctuating strongly meaning that the amount of correctly detected port scans is unpredictable. The false negative rate is fluctuating the most and the false positive rate is similarly high. This yields the results that hard to detect port scans when trained with the machine learning model lead to low detection performance.

After concluding the synthetic port scan detection the next experiments were done on the CICIDS2017 dataset. In the CICIDS2017 dataset on Friday multiple different port scan types are used. Two experiments were done with the CICIDS2017 dataset. One using a balanced aggregation map approach and one using an unbalanced aggregation map approach. It can be observed that the un-balanced experiment has slightly higher performance. The key difference being that the false positive rate is fluctuating significantly less than with the balanced approach. The conclusion drawn from this is that training enough background traffic is important for not classifying benign network traffic as a port scan.
Next the requirements from the analysis were evaluated. The reaction time of the aggregation based CNN approach presented in this thesis is estimated at around 0.15 seconds. In related work micro-flows are classified which have in the CICIDS2017 dataset an average of about five seconds of flow duration. This concludes that the reaction time is faster in the approach of this thesis.
Another requirement was a high recall. Meaning that port scans are detected with a high chance. In related work the recall is around 99 percent where the approach in this thesis only reaches slightly more than 80 percent. However, as mentioned before the reaction time is faster and the attempts to detect a port scan occur more often with the approach in this thesis.

The false positive rate has to be low, otherwise alerts of the system can not be trusted. As described in the corresponding experiment with the CICIDS2017 when using an unbalanced aggregation map dataset, the false positive rate is near zero percent.

Concluding from the experiments it is understood that the exposure time is an important parameter for detection, the resolution and threshold if not applied with extremes, have low impact on the classification results. The CICIDS2017 experiments highlighted that training background traffic is important and that unbalanced training results in improved classification performance.

7. Conclusion

An adapted aggregation and machine learning based method for detecting port scans is introduced. The detection is executed in two steps. First network traffic is aggregated into aggregation maps. These aggregation maps can be understood as a two-dimensional representation. The y-axis represents the destination ports and the x-axis the source IP address space, each divided into equally sized buckets. The second step is to feed the aggregation maps to a machine learning model that trains supervised. The machine learning model classifies binary if the aggregation maps from the testing data contain a port scan or benign traffic. Further experiments have been done on the CICIDS2017 [21] dataset which already contains port scans. Experiments were done to optimize the parameters of aggregation maps. The amount of aggregated traffic divided in different time durations for aggregation maps, got researched. The results show that higher exposure times than 15 seconds for aggregations decrease the relative amount of detected port scans. The resolution experiment which researched the bucket sizes showed that an aggregation map with more than 2x2 resolution is able to perform high accuracy detection. The threshold on which aggregation maps are labeled as containing port scans was researched. It is shown that a threshold lower than 200 did not have a negative impact on the detection performance. Conducting the detection method with the parameters of exposure time one seconds, resolution of 32x32 and threshold one, showed that synthetic injected port scans over the complete range of 2^{16} ports as well as port scans which only contain 1000 ports scanned, can be detected with high accuracy of nearly 100 percent. The limits of the detection method are observed in an experiment where only three ports are scanned in larger time durations. The performance metrics show that these types of port scans are not consistently detectable. Further the experiments done on the CICIDS2017 dataset showed that the CNN performance increases with a higher amount of background traffic. In this experiment it is observed that less benign traffic is classified falsely and more port scans are detected when using unbalanced datasets. The results show that the reaction time is tuneable to sub-seconds with TCAM. Meaning that a port scan is detected within 0.15 seconds. While this short reaction time is used, the false positive rate is low. The false positive rates maximum in the CICIDS2017 dataset was only 3.8 percent. The recall in the CICIDS2017 experiment shows how many of the possible port scan aggregation maps were classified correctly as port scan. The median of the recalls over epochs shows that with more than 80 percent

a port scan will be detected. Combined with the reaction time this means that every 0.15 seconds a port scan would be detected with a 80 percent chance. The research in this thesis shows that an aggregation based detection with CNN's provides desirable results in detecting port scans. The high performance can be observed with already one second time durations for each aggregation map. This shows that a normal port scan with 1000 ports scanned (Figure 6.8), which could take about half a minute, would be detected with a more than 99 percent chance each second it runs. This method supported with TCAM rules could be applied in SDN networks with public access, to detect if port scans are taking place.

7.1 Future Work

The following paragraphs describe future work to motivate new research in the field of aggregated port scan detection with CNN's.

Exposure Times

The most impacting limit in the research of this thesis are the eight hours of traffic which are used for attack injection. With higher exposure times the amount of aggregation maps shrinks dramatically as shown in Figure 6.2. Further research needs to be done on larger datasets. Especially the context of the consistent false positive and false negative totals over exposure times is to be researched. The question arises what happens if the machine learning model is not limited by the amount of aggregation maps when the exposure time is more than one minute.

Thresholds

In this thesis, only port scan detection with ideal parameters for the aggregation map have been considered. However more research needs to be done, if port scans with only 200-300 ports scanned get critical because they are close to the thresholds observed maximum in this thesis.

Attacks against the CNN

Another field of research is if the machine learning model can be injected with wrongly labeled training data to hinder the classification performance. In the eight hour dataset from CICIDS2017 attacker to victim labeling was used. This introduces less precise labels. However the performance of the metrics does not suffer significantly. Research should be done on actual forged wrong aggregation maps.

Attacks against the Aggregation Scheme

Further attacks need to be discussed. Is the aggregation scheme vulnerable if the attacker knows the set parameters for the aggregation maps? For example if the threshold for each aggregation map is set to 200. If the attacker is careful and only scans 199 ports at a given time, is the attacker able to hide from detection?

Real Environment Alarm Management

As shown in the results in this thesis the false positives of port scan detection can be reduced to nearly zero. However, they can still happen. Knowing that the total amount of false positives could be as high as 30 in eight hours of traffic, 30 alarms every eight hours would be problematic. However, the strength of the aggregation scheme is the high detection rate every second. Alarm thresholds should be tested. Meaning that if more than ten aggregation maps are classified as a port scan within 30 seconds an alarm is given.

Identifying Attacker and Victim

The research in this thesis mainly focuses on the destination port and source IP address. This results in the fact that it is clear a port scan happened, but which IP address did the port scan and which system got scanned is not clear. Identifying the actual attacker and target could be done with subsampling buckets. The process could be to when an attack is noticed, start adjusting the monitored subnet. This means that the bucket representing the subnet is used in the next aggregation map as the only address space, the attacker source IP address becomes more clear. Given a system that identifies the attacker IP address with high accuracy, is lower resolution meaning the fewer buckets hindering the detection, since the subnets contained in each bucket are larger? Motivation for this research is given, by that a system admin most likely does not want to filter an entire subnet connection request. This behavior could also potentially be abused by an attacker, to disable other IP addresses.

UDP Port Scans

In this thesis it is focused only on TCP port scans. However UDP port scans are also done on ports. It is to research if UDP port scans can be included into the aggregation scheme. Meaning that if a packet does not contain a TCP destination port, the packet is checked for a UDP destination port and the resulting tuple is used in the aggregation map.

Machine Learning and Port Scanning over time

In this thesis the aggregation maps are shuffled before they are fed into the CNN model. However, it can be assumed that a small amount of aggregation maps consist of port scans that just started or ended. Therefore, these aggregation maps contain nearly no detectable port scan behavior. If the CNN has access to previous aggregation maps, while they are also fixed in a timeline, this probably helps in identifying these aggregation maps more precisely. This can be realized with an Long Term Short Memory (LSTM) approach for CNN's.

Aggregation Map Dimensions Limits

In this thesis only IP source address and Destination Port are discussed. However, further filtering could be applied. In [23] it is described that anomaly scores are given to network events. This is likely not implementable with TCAM. However, if different dimensions are applied with only certain different TCP header fields new results could be achieved. The idea could be that the TCAM rule checks for a TCP header field and

SYN flag. Further, the TCAM rule can check if the payload of the SYN flagged packet is empty (highly suspicious). These packets would be kept longer or possible increment the aggregation map position more

Also new header fields could be that, only TCP SYN Packets are observed. Another approach is to observe the RST Flagged Packets by the running hosts. This flag filtering could be used with TCAM, by using fixed values in the bit sequence. The overall idea here being adjusting the aggregation process in a direction where it only aggregates anomaly events.

Bibliography

[1] Aksu, Dogukan and M Ali Aydin: *Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms.* In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pages 77–80. IEEE, 2018.

[2] Cordero, Carlos Garcia, Emmanouil Vasilomanolakis, Nikolay Milanov, Christian Koch, David Hausheer and Max Mühlhäuser: *ID2T: A DIY dataset creation toolkit for Intrusion Detection Systems.* In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 739–740. IEEE, 2015.

[3] Cordero, Carlos Garcia, Emmanouil Vasilomanolakis, Aidmar Wainakh, Max Mühlhäuser and Simin Nadjm-Tehrani: *On generating network traffic datasets with synthetic attacks for intrusion detection.* ACM Transactions on Privacy and Security (TOPS), 24(2):1–39, 2021.

[4] Curtis, Alexander E, Tanya A Smith, Bulat A Ziganshin and John A Elefteriades: *The mystery of the Z-score.* Aorta, 4(04):124–130, 2016.

[5] De Vivo, Marco, Eddy Carrasco, Germinal Isern and Gabriela O De Vivo: *A review of port scanning techniques.* ACM SIGCOMM Computer Communication Review, 29(2):41–48, 1999.

[6] Ghiëtte, Vincent, Norbert Blenn and Christian Doerr: *Remote identification of port scan toolchains.* In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2016.

[7] Huang, Huawei, Song Guo, Jinsong Wu and Jie Li: *Green datapath for TCAM-based software-defined networks.* IEEE Communications Magazine, 54(11):194–201, 2016.

[8] Jiang, Xuefeng, Yikun Wang, Wenbo Liu, Shuying Li and Junrui Liu: *Capsnet, cnn, fcn: Comparative performance evaluation for image classification.* International Journal of Machine Learning and Computing, 9(6):840–848, 2019.

[9] Jung, Jaeyeon, Vern Paxson, Arthur W Berger and Hari Balakrishnan: *Fast portscan detection using sequential hypothesis testing.* In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 211–225. IEEE, 2004.

[10] Kennedy, David, Jim O'gorman, Devon Kearns and Mati Aharoni: *Metasploit: the penetration tester's guide.* No Starch Press, 2011.

[11] Kopmann, Samuel, Hauke Heseding and Martina Zitterbart: *HollywooDDoS: Detecting Volumetric Attacks in Moving Images of Network Traffic.* In *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, pages 90–97, 2022.

[12] Kumar, Vinod and Om Prakash Sangwan: *Signature based intrusion detection system using SNORT.* International Journal of Computer Applications & Information Technology, 1(3):35–41, 2012.

[13] Lamping, Ulf and Ed Warnicke: *Wireshark user's guide.* Interface, 4(6):1, 2004.

[14] Lee, Hagyeong and Jongwoo Song: *Introduction to convolutional neural network using Keras; an understanding from a statistician.* Communications for Statistical Applications and Methods, 26(6):591–610, 2019.

[15] Muelder, Chris, Kwan-Liu Ma and Tony Bartoletti: *Interactive visualization for network and port scan detection.* In *International Workshop on Recent Advances in Intrusion Detection*, pages 265–283. Springer, 2005.

[16] Pale, Paulino Calderón: *Nmap 6: Network Exploration and Security Auditing Cookbook.* Packt Publishing Ltd, 2012.

[17] Panigrahy, Rina and Samar Sharma: *Reducing TCAM power consumption and increasing throughput.* In *Proceedings 10th Symposium on High Performance Interconnects*, pages 107–112. IEEE, 2002.

[18] Panjwani, Susmit, Stephanie Tan, Keith M Jarrin and Michel Cukier: *An experimental evaluation to determine if port scans are precursors to an attack.* In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 602–611. IEEE, 2005.

[19] Rehman, Rafeeq Ur: *Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID.* Prentice Hall Professional, 2003.

[20] Sakib, Shadman, Nazib Ahmed, Ahmed Jawad Kabir and Hridon Ahmed: *An overview of convolutional neural network: its architecture and applications.* 2019.

[21] Sharafaldin, I, AH Lashkari and AA Ghorbani: *Intrusion detection evaluation dataset (CIC-IDS2017).* Proceedings of the of Canadian Institute for Cybersecurity, 2018.

[22] Soltani, Mahdi, Mahdi Jafari Siavoshani and Amir Hossein Jahangir: *A content-based deep intrusion detection system.* International Journal of Information Security, 21(3):547–562, 2022.

[23] Staniford, Stuart, James A Hoagland and Joseph M McAlerney: *Practical automated detection of stealthy portscans.* Journal of Computer Security, 10(1-2):105–136, 2002.

[24] Vasilomanolakis, Emmanouil, Shankar Karuppayah, Panayotis Kikiras and Max Mühlhäuser: *A honeypot-driven cyber incident monitor: lessons learned and steps ahead.* In *Proceedings of the 8th International Conference on Security of Information and Networks*, pages 158–164, 2015.

[25] Wen, Xitao, Bo Yang, Yan Chen, Li Erran Li, Kai Bu, Peng Zheng, Yang Yang and Chengchen Hu: *RuleTris: Minimizing rule update latency for TCAM-based SDN switches.* In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 179–188. IEEE, 2016.

[26] Yin, Chunmei, Mingchu Li, Jianbo Ma and Jizhou Sun: *Honeypot and scan detection in intrusion detection system.* In *Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No. 04CH37513)*, volume 2, pages 1107–1110. IEEE, 2004.

[27] Yu, Fang, Randy H Katz and Tirunellai V Lakshman: *Gigabit rate packet pattern-matching using TCAM.* In *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, pages 174–183. IEEE, 2004.

[28] Zhang, Xu, Jeffrey Knockel and Jedidiah R Crandall: *Original SYN: Finding machines hidden behind firewalls.* In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 720–728. IEEE, 2015.