UNIVERSITY OF HASSELT

BACHELOR THESIS

Machine learning techniques for flow-based network intrusion detection systems

Author:
Axel FAES

Supervisor:
Prof. Dr. Peter QUAX
Prof. Dr. Wim LAMOTTE
Bram BONNE
Pieter ROBYNS

Bachelorproef voorgedragen tot het behalen van de graad van bachelor in de informatica/ICT/kennistechnologie

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Networks and Security
Computer Science

March, 2016

Declaration of Authorship

I, Axel FAES, declare that this thesis titled, “Machine learning techniques for flow-based network intrusion detection systems” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a bachelor degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITY OF HASSELT

Abstract

Wetenschappen
Computer Science

Bachelor of Science

Machine learning techniques for flow-based network intrusion detection systems

by Axel FAES

Acknowledgements

Contents

List of Figures

List of Tables

List of Abbreviations

IDS	**I**ntrusion **D**etection **S**ystem
IPS	**I**ntrusion **P**revention **S**ystem
IDPS	**I**ntrusion **D**etection (and) **P**revention **S**ystem
NIDS	**N**etwork (based) **I**ntrusion **D**etection **S**ystem
HIDS	**H**ost (based) **I**ntrusion **D**etection **S**ystem
DDOS	**D**tributed **D**enial **o**f **S**ervice
ML	**M**achine **L**earning
MSE	**M**inimum **S**quared **E**rror (function)

List of Symbols

$H_0(x)$	Hypothesis function
$J(\theta)$	Cost function

Nederlandstalige samenvatting

Chapter 1

Introduction

The internet is constantly growing and new network sevices arise constantly. This has as effect that security flaws become more and more important. Considering this, it becomes more important to be able to detect and prevent attacks on network systems.

1.1 Intrusion detection systems

An intrusion detection system is a system which tries to determine whether a system is under attack, to detect intrusions within a system. There are different types of intrusion detection systems or IDS. There are network-based intrusion detection systems and host-based intrusion detection systems. This thesis will uses machine learning techniques to detect malicious network behaviour, as such only network-based intrusion detection systems are covered.

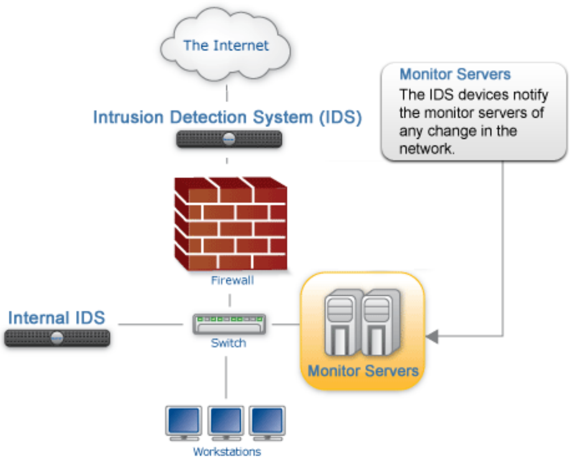
The Internet
Intrusion Detection System (IDS)
Monitor Servers
The IDS devices notify the monitor servers of any change in the network.
Firewall
Internal IDS
Switch
Monitor Servers
Workstations

FIGURE 1.1: An IDS can for example be placed within the network or just before the network.

1.1.1 Host-based Intrusion Detection Systems

Host-based intrusion detection systems are systems that monitor the device on which they are installed. The way they monitor the system can range from monitoring the state of the main system through log files, to monitoring program execution. In this way they can be quite indistinguishable from Anti-Virus programs.

1.1.2 Network-based Intrusion Detection Systems

Network-based intrusion detection systems are placed at certain points within a network in order to monitor traffic from and to devices within the network. The system can analyse the traffic using multiple techniques to determine whether the data is malicious. There are two different ways to analyse the network data. The analysis can be packet-based or flow-based.

Packet-based analysis uses the entire packet including the headers and payload. An intrusion detection system that uses packet-based analysis is called a packet-based network intrusion detection system. The advantage of this type of analysis is that there is a lot of data to work with. Every single byte of the packet could be used to determine whether the packet is malicious or not. The disadvantage is immediately obvious once we look at networks through which a lot of data passes, such as data centers. Analysing every byte is very work-intensive and near impossible to do in such environments. [1]

Flow-based analysis doesn't use individual packets but uses general data about network flows. An intrusion detection system that uses flow-based analysis is called a flow-based network intrusion detection system. A flow is defined as a single connection between the host and another device. A flow can be defined using a (source_IP, destination_IP, source_port, destination_port) tuple. However flowdata also contains other information such as the duration of the connection, the start time, the amount of bytes and/or packets within the flow. Flow data can even contain data such as the amount of SYN packets within the flow. This could be useful to detect SYN overflow attacks. However not every flow collector collects this data Since flow data is much more compact than all the individual packets, it is much more feasable for data centers to use flow-based intrusion detection systems.

1.1.3 Intrusion Prevention Systems

An intrusion prevention system or IPS/IDPS is an intrusion detection system that also has to ability to prevent attacks. An IDS does not necessarily need to be able to detect attacks at the exact moment they occur, although it is preferred. An IPS needs to be able to detect attacks real-time since it also needs to be able to prevent these attacks. For network attacks these prevention actions could be closing the connection, blocking an IP, limiting the data throughput.

1.2 IP Flows

Flows are aggregated from all packet data that travels through the network. Flow exporters are programs which collect network packets and aggregate them into flow records. A flow is not the same as a TCP connection. A flow can be any communication between two devices with any protocol. Flows are defined using a (source_IP, destination_IP, protocol) tuple. This is why flows are also called IP Flows.

Since flow data does not contain any payload information, intrusion detection systems that use flow data cannot detect malicious behaviour embedded within payload data. [2]

1.3 Detection

There are mutliple different methods to detect intrusions. There are **Signatures based methods** and there are **Anomaly Based** methods. Both of these methods have their own strengths and weaknesses. [3]

1.3.1 Signature based methods

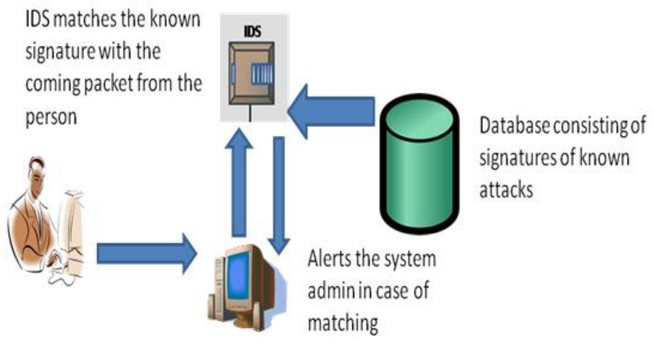
IDS matches the known signature with the coming packet from the person
IDS
Database consisting of signatures of known attacks
Alerts the system admin in case of matching

FIGURE 1.2: An Signature-based intrusion detection system.

Signature based methods compare so called "signatures" with an existing database of signatures. An packet or flow record is decomposed into features that together construct a signature. If the signature of an incoming flow or packet matches with a signature in the database, it is flagged as malicious. Signature-based methods have little overhead in both computation and preprocessing as it only tries to match incoming signatures to known signatures in the database. Because it only compares signatures, it is easy to deploy within a network. The system does not need to learn what the traffic within a network looks like.

Signature based methods are very effective against known attacks. New attacks cannot be detected unless the database is updated with new signatures. It is also for attackers to avoid being caught by signature based methods, only slight modification of the "signature" is required in order to bypass the exact matching. Updating the signature database requires a lot of technical effort, since new attacks are discovered all the time.

1.3.2 Anomaly based methods

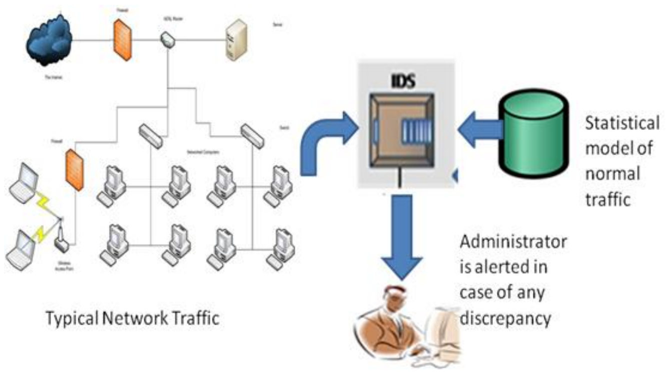
IDS
Statistical model of normal traffic
Administrator is alerted in case of any discrepancy
Typical Network Traffic

FIGURE 1.3: An Anomaly-based intrusion detection system.

Anomaly based methods, also called Behaviour based methods are methods in which the IDS tries to model the behaviour of network traffic. When an incoming packet deviates from this model, it is flagged as malicious and an alert is send. Because they use a statistical model of normal behaviour, they should be able to detect all deviations from this normal behaviour. As a result, new attacks that deviate to much from normal behaviour are detected aswell.

Since a model of the network traffic needs to be created, the system cannot be deployed into a network and be expected to work. The system needs to learn the behaviour of the network traffic. Problems, such as generating a lot of false positive alarms, can arise when training data includes mistakes, such as misclassifications.

Machine learning algorithms can be used as a anomaly based method. Machine learning techniques have the ability to learn from data and decide whether new data is malicous.

Chapter 2

Attack Classification

An intrusion detection system can use multiple methods to detect malicious behaviour. Since flow-based intrusion detection systems only have access to the flows and not the payload, they cannot detect every kind of attack. In order to make the IDS as effective as possible, the exact classifications of attacks that can be detected need to be known.

2.1 Classification

There are several types of attacks that can occur. Some of these attacks occur only on the network, other attacks infect computers, called malware. The exact classifications are not mutually exclusive. Some types of malware utilise network attacks. However it is important to make a distinction between these attacks. Every attack is identified by different characteristics. Knowing these characteristics is usefull to be able to tweak the IDS to make identification more effective.

2.2 network attacks

There are **Physical attacks**, these are attacks which attempt to destroy physical equipment and hardware. **Buffer overflows** are attacks that try to execute arbitrairy code or crash a process by overflowing a buffer on the targeted system. **Password attacks** attempts to break into a system by gaining the password that the system uses. The simplest password attacks are brute-force password crackers. **DDOS** attacks are attacks which attempt to make a network resource temporarily or permanently unavailable for the users of that resource. An attack could happen by flooding a system with TCP SYN packets. **Network scans** are information gathering attacks. They do not cause any damage by themselves but usually serve the purpose to gather information about a system that could be used in further attacks. Network traffic sniffing or port scans are examples of network scans. [2]

2.3 Malware

There are several types of malware. There are four distinct categories of malware. There are **botnets**, **viruses**, **trojan horses** and **worms**. Malware are actual programs that infect a system to execute a specific task. The task of the malware defines which categorie the malware belongs in.

Trojan horses are programs disguised as harmless applications but contain malicious code. **Worms** are programs that replicate themselves among a network. They can spread extremely fast. **Viruses** are similar to worms. However they only replicate themselves on the infected host computer. Thus they require user interaction in order to be spread around a network. The virus can accomplish this by attaching itself to an email-attachment, embed itself within an executable, etc.

Botnets is malware that causes infected computers to become "slaves" to the master. An infected computer is controlled externally by the bot-master without the knowledge of the owner of the infected computer. The bot-master can use the distributed network of "slave" computer to perform other malicious tasks, such as performing an DDOS attack. [2]

2.4 Detection

An NIDS only monitors the network. As such not every attack can be detected by an NIDS. Only the attacks that actually use the network can be detected. Flow-based IDS have the additional constraint that they can only use flow data. This further limits the attacks that can be detected. The attacks that can be detected using a flow-based network intrusion detection systems are:

- DDOS
- Network scans
- Worms
- Botnets

Other attacks either do not use network communication, or they are not visible within the header information of network traffic. In order to detect other attacks, including **viruses**, **trojan horses** and **Buffer overflows**, other detection systems such as HIDS or Packet-based NIDS should be used.

2.4.1 Distributed Denial of Service

A distributed denial of service can be detected by the amount of data that is being received. However, there are many different types of DDoS attacks. There are ICMP floods, SYN floods, etc. These attacks can be described in terms of traffic patterns. A traffic pattern is expressed in a couple features. These features include the number of flows and packets, the packet size, and the total bandwidth used during the traffic. For example UDP flooding can be characterised by a traffic pattern which contains a lot of packets. These patterns can be searched for during the detection phase. [4]

2.4.2 Network scans

There are three categories of network scans.

- Horizontal scans: a single port is scanned across many different devices.
- Vertical scan: several different ports are scanned on a single device

- Block scan: a combination of both a vertical and a horizontal scan.

Scans can also be described using traffic patterns. They are characterised with a high number of flow and a low number of packets. These can again be used to detect whether a vertical or horizontal scan occurs. [2] [4]

2.4.3 Worms

Worms exhibit different behaviour depending on their current state. Their are two different states, a target discovery state and a transfer state. In the target discovery state, the worm explores the network to find vulnerabilities and a host to infect. During the transfer state, the worm actually transfers itself to the targeted host. The Sapphire/Slammer worm is an example of this type of behaviour. [5]

Since transfering of the worm itself happends within the payload data, a flow-based NIDS cannot detect this state. The target discovery state can be detected. Worms use techniques similar to network scans in order to find vulnerable hosts. So similar detection techniques can be used to detect worms. [6]

2.4.4 Botnets

Botnets usually consist out of a huge amount of infected slaves controlled by a central bot-master. Locating the individual infected slaves and isolating them is a difficult problem but is also insignificant due to the huge amount of remaining slaves. Detecting the bot-master and isolating that device is key to taking down a botnet. However indentifing botnet behaviour is a far more difficult problem than detection other types of malicious activities. [7] Malicious behaviour alone is not enough to detect botnets.

Botnets often use IRC channels in order to communicate between slaves and the bot-master. These can be indentified using flows since they often use specific ports. It is possible to use a method that does not require specific port numbers. This requires flows including extra information such as the number of packets for which the PUSH flag is set.

Chapter 3

Machine learning

3.1 What is machine learning

Machine learning is a subfield from Computer Science. It is a type of Artificial Intelligence which allow programs to learn without being explicitly programmed.

There are two classes of machine learning algorithms. There is **supervised** learning and **unsupervised** learning. Supervised learning is trained using labeled data. Labeled data is data which consists of input data and the corresponding output data. Unsupervised learning uses unlabeled data. The data used to train machine learning algorithms is called a **training set**.

A **training sample** is a data point in an available training set that is used in a predictive modeling task. For example, if machine learning is applied to spam filtering, the training set is a collection of emails, some of which are spam emails. A training sample would be a single email. For example, if we are interested in classifying emails, one email in our dataset would be one training sample. Alternative names are a training example or training instance.

Machine learning is applied for predictive modeling. In predictive modeling, a particular process is modeled. Using a training set, the model tries to learn or approximate a particular function that, for example, let's us distinguish spam from non-spam email. This function is called the target function. To model the process, one or more **features** are extracted from the process. A feature is an individual property of a process. In the example of mails, a feature could be the textbody, or it could be the sender of the email.

3.1.1 Hypothesis

Machine learning relies heavily on a hypothesis. This is a function that transform a given input to the machine learning algorithm into the required output. It is a function that tries to model the the target function. When only one feature is considered, a hypothesis is a function of the form:

$$H_0(x) = \theta_0 + \theta_1 * x$$

For example, we could use the grades of high school students to predict their chance of success at university. x represents the grades (on a scale from 0 to 10) for students and y is the chance of success. Given is input

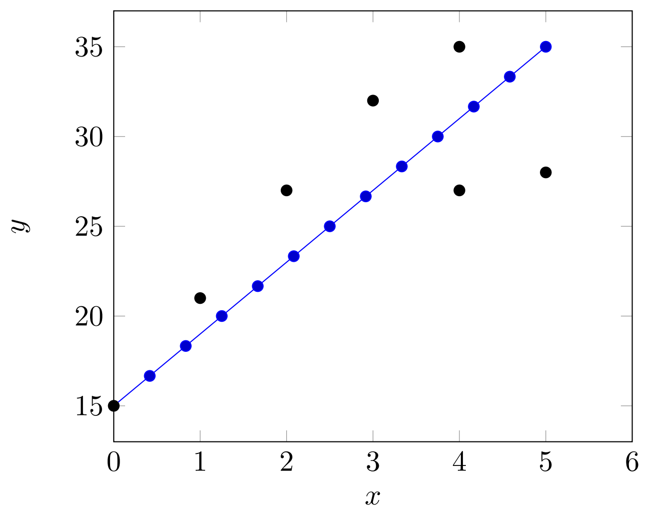

FIGURE 3.1: Regression example

data (the black points) and from that data a hypothesis (the blue line) is constructed.

3.1.2 Multiple feature hypothesis

The hypothesis function can be generalised for N properties. It has the form:

$$H_0(x) = \theta_0 x_0 + \theta_1 x_1 + ... + \theta_n x_n$$

x_0 always has the value 1. The θ values and the x values can be represented using vectors:

$$\theta = [\theta_0, \theta_1, ..., \theta_n]^T$$

$$X = [1, x_1, ..., x_n]^T$$

Now the hypothesis function can be written as:

$$H_0(x) = \theta^T X$$

3.1.3 Cost function

In order to construct a good hypothesis function, good values of θ have to be found. This can be seen as a minimization problem. The difference between any output y and $H_0(x)$ has to be minimized. More concretely, the squared difference has to be minimized. This is the MSE, "Minimum Squared Error" function or also called the cost function. With dataset size m, the MSE is:

$$J(\theta) = \frac{\sum_{i=1}^{m} (H_0(x^i) - y^i)^2}{2m}$$

3.1.4 Gradient descent

To solve the minimization problem, the first step is to start with θ and keep changing the values to minimize $J(\theta)$, this is an iterative approach. Gradient descent is such an algorithm that can be used to find a solution to the minimization problem. Gradient descent is an algorithm that uses the gradient or derivative of a function to find a local minimum of that function.

A gradient descent algorithm does this using the following algorithm with a simultanious update for all values of θ:

```
repeat until convergence {
    $\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$
}
```

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\sum_{i=1}^{m} (H_0(x^i) - y^i) * x_j^i}{m}$$

α is the learning rate of the gradient descent. The value of α describes how fast the gradient descent algorithm approaches the local mimimum. If α is too small, the gradient descent can be very slow. In the other case, if α is too large, the gradient descent can overshoot the local mimimum. It may fail to converge and could even diverge.

The value of α does not need to change during the gradient descent, since the closer the gradient descent gets to the local mimimum, the smaller the derivative becomes, and smaller steps will be taken. If θ_j is already a local minimum, the derivative is 0 and the gradient descent will not change the value of θ_j.

3.1.5 Feature scaling

The main idea behind feature scaling is to make sure that the different features are on a different scale. The reason behind this is to optimize the gradient descent algorithm. When the scale is very different, the gradient descent will not alter θ_j much after each step. The range that should approximately be used is $-1 < x < 1$.

Mean normalization could be used. This replaces each x_i with $\frac{x_i - \mu_i}{s_i}$, where μ_i is the average value of all x_i values and s_i is the standard deviation.

3.2 Supervised learning

3.2.1 Lineair Regression

Lineair regression is an statistical approach to model the relationship between an "output" value y and one or more "input" values X. An example of this can be seen in Figure 3.1. The black dots represent the data to be modeled. The blue line is the model. This example only has one "input" value, this specific case of regression is called simple linear regression.

When data follows a polynomial model, you could manipulate the feature so that the hypothesis forms a polynomial function, for example:

$$H_0(x) = \theta_0 + \theta_1 * x + \theta_2 * \sqrt{x}$$

Gradient descent executed on a lineair regression cost function will always provide a optimum, absolute minimum. For lineair regression, there is another method that could be used in place of gradient descent, a normal equation. This is a method to solve for θ analytically. But this method becomes slow when there are a lot of features. X is a mxn matrix constructed by putting all training samples (vectors of features) together.

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\sum_{i=1}^{m} (H_0(x^i) - y^i) * x_j^i}{m} = 0$$

$$\theta = (X^T X)^{-1} X^T y$$

But what happens when $(X^T X)$ is non-invertible, or singular. This means there are redundant features or more features than training samples.

3.2.2 Classification

In a classification model, the machine learning algorithm tries to sort data into different classes. The simplest version is binary classification. For example, is the IP flow malicious or not. In lineair regression, the hypothesis can output values other than the classes that exist. However there is a method, logistic regression, which constrains the hypothesis to the available classes.

All classes have to be discrete values.

3.3 Unsupervised learning

3.3.1 Clustering

3.4 Machine learning algorithms

3.4.1 Support Vector Machines

3.4.2 K-Nearest Neighbor

Chapter 4

Machine learning for an IDS

4.1 Using ML for an IDS

An intrusion detection system has to detect whether some data it receives is either malicious or regular web traffic. This can be seen as a classification problem which means an machine learning algorithm for classification could be used. It needs to be determined whether data is either normal network traffic or malicious behaviour.

Some parameters have to be chosen that will be feed into the machine learning algorithm.

4.2 Disadvantages of using ML for an IDS

4.2.1 Problems

As said before, machine learning for an intrusion detection system is a classification problem. More precisely, it can be said that intrusion detection systems have to detect abnormal behaviour in a network with mostly normal behaviour. There are several problems that can be encountered when using machine learning techniques.

The first problem is the ability to detect new attacks. A machine learning algorithm compares incoming data with a model that it has created internally. An new type of malicious behaviour might appear to be closer to normal network traffic as compared to the model of known attacks.

Another problem is the diversity of network traffic. The notion of "normal network traffic" is difficult to actually define. The bandwidth, duration of connections, origin of IP addresses, applications used can vary enormously through time. This makes it quite difficult for machine learning algorithms to distinguish between "normal network traffic" and malicious behaviour.[8]

4.2.2 Solutions

There are several solutions that can be used in order to make machine learning algorithms more effective for intrusion detection systems. One option is to chance the way the classification problem is defined. Instead of defining the classes, "normal" and "malicious", there might be different classes for different types of malicious behaviour. In the same way, different classes can be defined for different types normal traffic.

4.3 Advantages of using ML for an IDS

Chapter 5

Flow data

5.1 How to use flow-data

The following attributes are available with flow-data:

- Source IP
- Destination IP
- Protocol name
- Source port
- Destination port
- Starting time of the flow
- Duration of the flow
- Amount of packets in the flow
- Amount of bytes in the flow

However, should an flow exporter be implemented, some additional features can be generated from packet data. 1.2

- Amount of TCP SYN within the flow
- Source and Destination Type of Service
- Payload size

These data can be used within the machine learning algorithms. However some variables have undesirable effects on the accuracy of the algorithm. Some care should be taken when training the machine learning algorithms with the additional data. Not all data, both training data as predictive data, will have the additional features.

Most machine learning libraries use numeral data instead of string data. All string data has been hashed in order to be able to use it in machine learning algorithms. The probability on a collision is low enough to be able to ignored.

5.1.1 IP addresses

Flow data can contain multiple forms of IP addresses. Both IPv4 and IPv6 data can be found. For some protocols the flow data can also contain the MAC-addresses instead of the IP-address. These addresses are hashed, so they become numeral, discrete data and are then fed into the machine learning algorithm.

Using the IP, it is possible to find the country or region of origin. Tests where the country of origin was fed into the machine learning algorithms, have been done. Results however showed, that the accuracy of the IDS became lower.

TABLE 5.1: The effects of using IP country-of-origin on accuracy of IDS.

With Country-of-origin	Accuracy
Yes	96.16%
No	98.57%

5.1.2 Ports and protocol name

Both the source and destination port are discrete data. They are usually received in decimal form, however some data-sets might use them in hexadecimal data or refer to ports as "ssh port" instead of "22". Port data, in decimal form, can be directly fed into the machine learning algorithm.

The protocol name can simply be converted to a standard string in lower case, in order to avoid errors by lower and uppercase forms of the same name (for example "tcp" and "TCP"). This string can than be hashed into a discrete value.

5.1.3 Timing

5.1.4 Size

The amount of packets used in the flow and the amount of bytes are both discrete data. They are always received in decimal form. They can immediately be fed into the machine learning algorithm.

Chapter 6

Prevention

This chapter will only be done if this is made in the thesis

6.1 Real-time detection

6.2 Data limiting

6.3 Connection closing

Chapter 7

Implementation

7.1 Structure

7.2 Class diagram

Chapter 8

Datasets

8.1 Cegeka

8.2 CTU Datasets

8.3 Own generation

8.4 Inline placement

Chapter 9

Visualisation

9.1 Logging

9.2 Graphing

Chapter 10

Conclusion

Appendix A

Meetings

A.1 Meeting 1: 9 Feb 2016

aanwezigen: Peter Quax, Bram Bonne, Pieter Robyns, Axel Faes

Dit is de eerste bijeenkomst met de begeleiders en promotor. Er is dus geen rapportering mogelijk van een vorige bijeenkomst. Tijdens de bijeenkomst is beslist om een *intruder detection system* te bestuderen en te implementeren.

De actiepunten die gedaan moeten worden:

- Beslissen voor wie het systeem gemaakt moet worden. Gaat dit voor end users zijn, of voor grote data centers. Hieraan hangt vast welke data (packets of netflow) gebruikt moet worden.
- Bekijken hoe machine learning algoritmes gebruikt kunnen worden in een *intruder detection system*.
- Bekijken wat netflow is.
- Er moet gekeken worden naar de manier waarop anomalies gegenereerd gaan worden om het systeem te testen/trainen.

Volgende afspraken zijn gemaakt:

- Er is gevraagd om te zorgen dat het systeem ook op correcte wijze informatie kan weergeven aan gebruikers. Tijdens het semester moet bekeken worden hoe deze weergave moet gebeuren.
- Libraries gebruiken indien mogelijk, om te vermijden dat het wiel opnieuw uitgevonden word.
- Er is de mogelijkheid geboden om aan de thesis te werken op het EDM.
- Er is afgesproken om *Overleaf* te gebruiken om de thesis in te schrijven.
- Een ruwe planning voor het werk moet gemaakt worden tegen 12 Feb.
- Een wekelijkse meeting is vastgelegd. Dit om 10:00 elke vrijdag.
- Begin mei moet een eerst draft van de thesis klaar zijn en eind mei moet de finale draft af zijn.
- Er moet een vulgariserende tekst gemaakt worden en een postersessie gegeven worden (op 29 juni).

A.2 Meeting 2: 12 Feb 2016

aanwezigen: Bram Bonne, Pieter Robyns, Axel Faes

Dit is de tweede bijeenkomst met mijn begeleider. Netflow bevat op zichzelf niet zoveel informatie, maar het is toch handig om te kijken welke bevindingen gemaakt kunnen worden met deze data. Mogelijks kan er, indien gevonden wordt dat netflow alleen niet genoeg informatie bevat, ook gebruikt gemaakt worden van packet data.

Er is de mogelijkheid besproken om eventueel meerdere machine learning algoritmes te implementeren en te bekijken in welke situaties welke algoritmes beter werken.

De actiepunten die gedaan zijn:

- *Beslissen voor wie het systeem gemaakt moet worden.*: Dit gaat gedaan worden voor data centers
- Er zijn verschillende classificaties van machine learning algorithmes gevonden die gebruikt kunnen worden.
- Verschillende grote data sets van netflow en packets met sporen van anomalies zijn gevonden. Alsook programma's om verkeer te genereren.

Volgende actiepunten zijn besproken:

- Verder uitwerken van welke machine learning algoritmes gebruikt kunnen worden
- Bekijken netflow v9

A.3 Meeting 3: 19 Feb 2016

aanwezigen: Bram Bonne, Pieter Robyns, Axel Faes

Professor Quax is aan het bekijken ofdat ik (gelabelde) netflow data kan verkrijgen van Cegeka. Dit zou heel handig zijn om mijn implementatie te testen op real world data.

Voorlopig moet ik enkel focussen op een passive intrusion detection systeem, geen preventie en niet direct inline in het netwerkverkeer. Ook de visualisatie moet later bekeken worden, de gebruiker is een netwerkadministrator. Er is tevens besproken dat python zelf mogelijks te traag is om packet sniffing op een goede snelheid uit te voeren. Hiervoor zou ik wireshark kunnen gebruiken (of de command line versie). Er is besproken om eventueel zelf datasets te genereren door malware te runnen op een VM of aparte machine.

De datastructuur voor de machine learning algoritmes is bekeken. Ik moet eens bekijken hoe de timestamps van de flowdata gebruikt kunnen worden. Om de effectiviteit (van de machine learning algoritmes) mogelijks te verhogen ga ik eens bekijken of ip-adressen ingedeeld kunnen worden in country-of-origin of iets dergelijks. Dit zou de machine learning algoritmes de mogelijkheid bieden om ook op deze parameter te bekijken of data malicious is of niet.

De actiepunten die gedaan zijn:

- Er is al een basis implementatie uitgewerkt voor het IDS
- De netflow structuur is bekeken en er is een datastructuur opgestelt die gefeed kan worden aan verschillende machine learning algoritmes.
- Progressie in de machine learning cursus: chapter 3 van de 18.

Volgende actiepunten zijn besproken:

- Beginnen aan de thesis: het schrijven van een hoofdstuk over machine learning en over hoe deze algoritmes toegepast kunnen worden op een intrusion detection systeem.
- Verder werken in de machine learning cursus.
- Ik moet eens bekijken ofdat ik een programma vind om pcap files om te zetten naar netflow. Anders moet ik dit zelf schrijven.

Ik heb ook een korte planning gemaakt van hoe de thesis eruit zou zien:

- Inleiding:
  - wat is een IDS
  - Waarom is er gekozen voor dit type IDS (host vs netwerk)
  - Waarom voor data centers
  - Waarom netflow
  - Waarom machine learning
- Wat is machine learning
- Hoe passen we machine learning toe op IDE en wat zijn de voor/-nadelen
- Welke machine learning algortimes zijn wel/niet gebruikt
- Wat zijn de voor/nadelen van netflow
- Hoe met combinatie netflow/packets (Als dit gedaan zou worden)
- Welke data sets zijn gebruikt
- Wat zijn de bevindingen
- Hoe kan visualisatie/feedback gebeuren (richting admin en richting automatische preventie)
- Conclusie

A.4 Meeting 4: 26 Feb 2016

aanwezigen: Bram Bonne, Axel Faes

Deze week is voornamelijk besteed aan de implementatie. Er is een netflow exporter geschreven. Er is bekeken ofdat timestamps gebruikt kunnen worden en ofdat ip-adressen opgedeeld kunnen worden per land. Er is besloten dat dit zeer weinig effect heeft op de accuraatheid van de machine learning algoritmes.

Momenteel zijn Support vector machines en K-nearest Neighbor Classifier algoritmes bekeken. Het K-nearest Neighbor Classifier algoritme is zeer efficient (98%).

In een later stadium kan bekeken worden om eventueel verdere analyse te doen op de data die malicious gevonden is, eventueel door pakketten te analyseren, of nogmaals door machine learning technieken. Er kan ook eens bekeken worden om een VM op te zetten, en daarin malware te runnen en dit verkeer te monitoren. Herbij zouden eigen datasets gegenereerd kunnen worden.

De machine learning cursus is gevolgd tot hoofstuk 7. De cursus zou normaal af moeten zijn binnen 2 weken.

De actiepunten die gedaan zijn:

- Er is al een netflow exporter geschreven
- Er zijn experimenten uitgevoerd m.b.t de datastructuur die meegegeven wordt aan de machine learning cursus.
- Progressie in de machine learning cursus: chapter 7 van de 18.
- Er is begonnen aan de thesis.
- Het zou interessant zijn om eens te kijken ofdat ip-addressen opgedeeld kunnen worden in subnets.

Volgende actiepunten zijn besproken:

- Focussen op de thesis
- Verder werken in de machine learning cursus.

Bibliography

[1] H. Alaidaros, M. Mahmuddin, and A. Al Mazari, “An overview of flow-based and packet-based intrusion detection performance in high speed networks”, in *Proceedings of the International Arab Conference on Information Technology*, 2011.

[2] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, “An overview of ip flow-based intrusion detection.”, *IEEE Communications Surveys and Tutorials*, vol. 12, no. 3, pp. 343–356, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/comsur/comsur12.html#SperottoSSMPS10.

[3] M. J. N. Jayveer Singh, “A survey on machine learning techniques for intrusion detection systems”, *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 11, 2013.

[4] A.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, “A flow-based method for abnormal network traffic detection”, in *Network operations and management symposium, 2004. NOMS 2004. IEEE/IFIP*, IEEE, vol. 1, 2004, pp. 599–612.

[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, “Inside the slammer worm”, *IEEE Security & Privacy*, no. 4, pp. 33–39, 2003.

[6] Y. Abuadlla, G. Kvascev, S. Gajin, and Z. Jovanovic, “Flow-based anomaly intrusion detection system using two neural network stages”, *Computer Science and Information Systems*, vol. 11, no. 2, pp. 601–622, 2014.

[7] Z. Zhu, G. Lu, Y. Chen, Z. J. Fu, P. Roberts, and K. Han, “Botnet research survey”, in *Computer Software and Applications, 2008. COMPSAC’08. 32nd Annual IEEE International*, IEEE, 2008, pp. 967–972.

[8] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection”, in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP ’10, Washington, DC, USA: IEEE Computer Society, 2010, pp. 305–316, ISBN: 978-0-7695-4035-1. DOI: 10.1109/SP.2010.25. [Online]. Available: http://dx.doi.org/10.1109/SP.2010.25.