Creating firewall rules with machine learning techniques

Author:
Roland Verbruggen, Student at Radboud University Nijmegen
Roland-Verbruggen@student.ru.nl

Abstract

The amount and diversity of malware keeps growing [M. Fossi] while the same basic attack techniques are being used [M.v.j.]. This renders security defenses ineffective such that millions of computers are infected with malware in the form of computer viruses, internet worms and Trojan horses. This cost the society money [G. Lovet, M. Clement, C. Kanich]. Intrusion detection is a critical component when fighting cybercrime in the area of network security. Intrusion detection systems (IDS) look at characteristics of network packets or series of packets to determine whether the packets are malicious or not. The goal of this thesis is to show how better and more complex firewall rules can be created with the use of machine learning algorithms. These new firewall rules contribute to better computer intrusion detection systems. This thesis looks at existing machine learning methods such as random forests [L. Breiman] and neural networks [K. Hornik] and how firewall rules can be extracted from the resulting models. The models are trained and tested on a novel labeled network data set containing malicious and normal packets. This data set was created for this thesis and has not been used before. How the data set is created is also presented in this thesis. However, as machine learning techniques work well on networks with similar traffic such as SCADA networks, a special chapter in Appendix A is included to this thesis on SCADA networks.

Contents

1 Introduction

This chapter explains how machine learning techniques can help in the area of IDS and motivates the development of IDS systems.

1.1 Motivation

Secure computer systems should assure the following services: integrity, authentication, non-repudiation, confidentiality and availability[A. Simmonds]. Integrity assures that no cyber criminals change data that is stored on computer systems or being transmitted between computers. Confidentiality assures that no information is disclosed to unauthorized people. Availability assures that the information on a network can be requested if it is needed. If cyber criminals violate these services this costs our society money. It is hard to estimate how much money because it is hard to see what costs need to be taken into account and what costs should not be taken into account: there is no framework to access the economic costs [M. Eeten]. What also makes it harder to estimate the costs is that corporations do not like to make it public when they suffer from computer intrusions [B. Cashell]. However, cost estimates have been made [G. Lovet, M. Clement, C. Kanich]. A report of McAfee reports that the global cybercrimes costs range from $300 billion to $1 trillion and from $ 24 billion to $120 billion in the US alone [McAfee].

Intrusion detection systems are needed to put a stop to cyber attacks. These systems require a model of intrusion: what should the IDS look for? This model that distinguishes between an attack and normal data is created with machine learning techniques. Research on computer security with machine learning algorithms has been done before, for example, by Konrad Rieck [K. Rieck]. Others have tried to discover patterns or features that can be used to recognize anomalies and known intrusions [S. Mukkamala]. Using machine learning algorithms to detect intrusions has several advantages; zero-day malware can be detected through statistical analysis of previous examples. Machine learning can also help data analysts to analyze large amounts of data by classifying programs or data in groups that might be malicious or not. Traditionally data analysts looked at patterns in connections from certain IP addresses with histories of intrusive behavior. However, intrusions have become more complex. For example, intrusions can be low and slow which means that an attack consists of intrusive behavior over hours, days or weeks and they can have more than one network source. Machine learning can be used to help the data analyst by doing complex pattern recognition. Automating this work has the advantage that it can monitor and correlate large numbers of intrusive signatures or patterns.

1.2 Our contributions

This thesis contributes to the war against cybercrime. This is done by creating more complex firewall rules with the use of machine learning algorithms. This paper contributes by giving insight in how to use machine learning algorithms

to create these better firewall rules. This method is novel in the sense of how the dataset is created on which the machine learning algorithms are applied. How the data set is created is also described in this paper.

The goal of this thesis is to to create a model using machine learning techniques that can differentiate between malicious and normal network traffic, extract a set of rules from this model and then extract useful firewall rules from this set. The model can not directly be used in a real environment or as a firewall because it is unrealistic to assume that a model can be created of which the false positive rate of this model is low enough while still having a good true positive rate. The research Question therefore is:

Can one use machine learning techniques to build a model that differentiates between malicious and normal traffic, extract a set of rules from these models and then create useful firewall rules from this set that can be used in a real environment?

This question is answered by creating the model, testing its performance, extracting a set of rules and investigating these rules.

1.3 Organisation of this thesis

The introduction of this thesis explains why it is important to invest in computer intrusion detection systems and briefly motivates the goal of using machine learning methods to automate intrusion detection. The research question is also introduced in the first chapter. Chapter 2 describes the machine learning methods that are being used in this thesis. The third chapter describes the methodology for achieving the answer of the research question. In particular, it explains how the data set is created and how to apply the machine learning methods. The fourth chapter describes the data sets that are used to train and test the models on. The experiments that are run on the data set and their results are given in chapter 5. The empirical evaluation and the conclusions of the experiments are also presented in chapter 5. In chapter 6 the resulting firewall rules are given. The conclusions are discussed in Chapter 7. Appendix A suggests to create firewall rules for SCADA networks with machine learning techniques. Appendix D contains visualisations of models that resulted from the machine learning techniques.

2 Machine learning methods

Machine learning concerns the construction of systems that can learn from data. A broad definition of machine learning is given by [M. Tom]: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." In our case a machine learning system learns to distinguish between malicious and normal network traffic.

Its performance gets increased each time it trains on another example or instance during the training phase. Inputting a training instance can be seen as an experience. There are various kinds of machine learning methods. Machine learning can be supervised or unsupervised. If it is supervised the data that the computer program learns form is labelled. In other words, the computer program can learn from examples with the right answer. If the data does not have labels this means that the data does not contain the class to which the instance belongs. This is also called clustering. In our research the data is labeled, so it is a supervised machine learning task. In the subsections below some supervised machine learning approaches are discussed.

2.1 Multilayer perceptron

A multilayer perceptron (MLP) or neural network is a computational model that consists of interconnected perceptrons or nodes that can recognize patterns. Warren McCulloch and Walter Pitts were the first (1943) to create a computational model for neural networks based on mathematics and algorithms [S. Warren]. An MLP consists of at least 2 columns of nodes. On the first column the input is clamped and the last column outputs the results.The output neuron that has the biggest output is the class to which the input is classified. Each node is fully connected to all nodes in the column after it, the connections between them have weights. An MLP learns by adjusting these weights w. A network with only 2 column is called a pattern associator (PA). The PA is a linear classifier. When the PA is working correctly the output unit j linked to the correct class should have the highest activation. The activation a is calculated by taking the net-input n through sigmod function where the net input n is the sum of the input units i, times its weight w:

$$a_j = \frac{1}{1 - e^{-n_{aj}}}$$

where

$$n_{aj} = \sum_{1}^{i} a_i * w_{ij}$$

A PA cannot solve all input patterns. For example, the XOR-problem cannot be learned by the PA [M. Eldracher]. A multilayer perceptron network is used in this thesis. This is different from a pattern associator in the sense that it

has one or more hidden layers: the multilayer perceptron network has at least 3 columns of nodes. This network can learn the XOR-problem.

Before the MLP network is capable of linking the input data to the correct class, the correct weights should be determined. This is done by training the network with the supervised data set and updating the weights. How this works is explained with just one hidden layer. The learning process in an MLP is called back propagation. Back propagation has three stages:

1. Forward propagation. The network calculates the output o by subsequently computing:

The activation of the hidden units k

$$a_k = \frac{1}{1 - e^{-n_{ak}}}$$

where

$$n_{ak} = \sum_{i=1}^{k} a_i \cdot w_{ik}$$

$$o_j = \frac{1}{1 - e^{-n_{aj}}}$$

where

$$n_{aj} = \sum_{i=1}^{j} h_k \cdot w_{kj}$$

This is illustrated in figure 1:

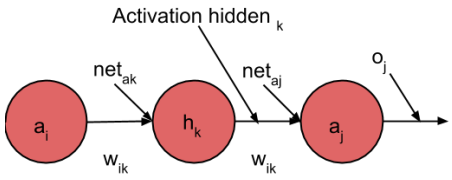

Figure 1: Illustration of forward propagation

2. Backward propagation

In the back propagating phase the learning takes place. The weights learn through the error e (= activation Output o minus the desired outcome t (the target) computed at the output layer:

$$e = o - t$$

The weight is then changed by e times the activation output a_j times the activation of the hidden neuron h_k. e Is a variable is a variable defined by the user. If it is higher it trains faster but it could jump over a global optimum.

$$\Delta w_{kj} = e \cdot a_j \cdot h_k$$

3. Update weights

The new weight will become:

$$w_{kj} = w_{kj} + \Delta w_{kj}$$

The learning will stop after some termination criteria (a number of epochs for example) or when certain recognition performances have been reached.

As mentioned earlier the MLP is good at being able to learn not only from linear but also nonlinear data. A quote from Hornik: "The MLP is capable of approximating any measurable function to any desired degree of accuracy, in a very specific and satisfying sense." [K. Hornik]. Hornik even goes as far as calling the MLP a universal approximator. The MLP also has disadvantages. For one, there are difficulties in implementing and training the network [M. W. Gardner]. For example a common mistake is leaking information from the test set to the training set. Creating easy to use programs might solve this problem or make it easier but background knowledge of MLP is still required when using MLP's. Another disadvantage of the MLP is the difficulty of interpreting the MLP. The resulting model of the training phase is complex and it is hard to see why certain instances are classified into a certain class. There do however exist visualization methods for MLP's [F. Piekniewski].

The last disadvantage of the MLP classification method is the complexity of the classification process. Using an MLP in combination with large data sets requires alot of time to perform classification. Below the complexity of the forward propagation of the MLP is evaluated.

First the node net-input needs to be computed. The complexity of this is $2dn$ where d the number of training instances and n the number of features. The number of features is also the amount of input neurons.

Then the input is evaluated, this is done by activating each node in each layer. The complexity of the node activation is:

$$d \sum_{s=2}^{N} P_s$$

where P_s is the number of nodes at layer s. The activation of the nodes generates output. This needs to be evaluated. This is done with a residual evaluation. The residual evaluation gives simply the distance between the right outcome and the outcome given by the model. This needs to be calculated for each output node and for each training instance. The complexity of this is dP_N. We also need to calculate the Error. The complexity of Calculating the SSE is: $2dP_N$ The complexity of residual evaluation is dP_N. The complexity of the backward pass is also primarily dependant on the number of instances, the number of features and the number of nodes [E. Mizutani]. So when using large data sets with a large amount of features, this gives problems. Also using large amounts of hidden units in the MLP increases the complexity drastically.

2.2 Decision tree

Next to MLPs also decision trees were used to learn patterns from the network data. The decision tree takes as input the attributes of the messages and returns a "decision" - the predicted output value for the input. The input values can be discrete or continuous, this is an advantage over the MLP where the input cannot be continuous. The decision tree makes its decision by performing a sequence of tests. Each node in the tree corresponds to a test of the value of one of the attributes. The values of the leaves of the tree are returned and represent the class to which the instance belongs.

The basic idea behind decision tree learning is to test the most important attribute first. The most important attribute is the attribute that makes the biggest difference to the classification of a sample, put in other words, the attribute that has the highest information gain. The measure for information gain is called entropy. If an instance can be classified into C different values, then the entropy of S relative to this c-wise classification is defined as:

$$Entropy(S) = \sum_{i=1}^{c} P_{\mathrm{i}} \log_2 p_{\mathrm{i}}$$

where p_i is the proportion of instance S belonging to class i and the logarithm is base 2. [M. Tom].

By testing the most important attributes it is hoped that one gets the right classification with a small number of tests (small tree). As mentioned before, one advantage is that input values can be discrete or continuous. The output values of the decision tree are discrete. Another advantage is that decision trees are robust to errors. The training data may contain missing attribute values. A challenge in decision trees is determining the depth of the decision tree. If the tree becomes too large and the depth to high the tree is overfitted on the training data. If there is some noise in the data or when the number of training examples is too small to be representative for the real data set the overtrained model will not perform well on real data. The solution would be to stop at the correct tree size or stop the training early. The correct tree size could be determined by using separate data from the training data to evaluate the decision tree. This technique was also used in this research and is also called training and validation set approach. A simpler solution that was also used in this research is to determine a maximum depth to which one lets the decision tree expand or use another explicit measure to measure the complexity of the tree and halt at a certain threshold. Another option would be to use a statistical test to estimate whether the expanding of a particular node will improve the decision tree. For example, the chi-square test can be used. This method was not used in this research. Choosing the shortest decision tree for the observed data is also in line with Occams Razor. Occams Razor is a principle that states that among competing hypotheses (decision trees), the hypothesis with the fewest assumptions should be selected [A. Blumer].

The learning algorithms differ for each technique and will be discussed below.

2.2.1 Random tree

A Random decision tree is a decision tree where the input is not only the instances but also some random vector O_k. An example of a random tree is using split selection where at each node the split is selected at random from among the K best splits. Another approach to introduce randomness is to use bagging. If one has a training set S of size N, one creates m new training sets S_i each of size N. These subsets S_i are created by taking samples from D. It is possible that there are duplicate elements in the subsets. Bagging has several advantages. Bagging works good when there is classification noise [T. Dietterich]. Classification noise means that a small amount of the labels are wrong. Bagging also helps to avoid overtraining, this is shown in section 2 of [L. Breiman].

2.2.2 Random forests

A random forest is an ensemble of multiple random decision trees, in other words, a bag of random trees. In our setup we used an ensemble of 10 random trees. Each random tree in the forest is fully grown (no pruning). The definition of a random forest is given by Breiman [L. Breiman]: "A random forest is a classifier consisting of a collection of tree structured classifiers $h(x, O_k), k = 1, ...$ where the O_k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x."

The output of all trees is combined with a combination rule. One example of such a combination rule is majority voting. In majority voting, for each instance, the majority voting rule lets each classifier classify it into a class. It then just outputs the class that most of the trees classified it into. An advantage of majority voting is that it is straightforward.

3 Methodology

The methodology of this thesis is explained in the following six steps:

1. Create data
2. Feature extraction of data
3. Combine data
4. Train model
5. Test model
6. Visualize and interpret results

The methodology that is explained in this chapter is depicted in figure 2.

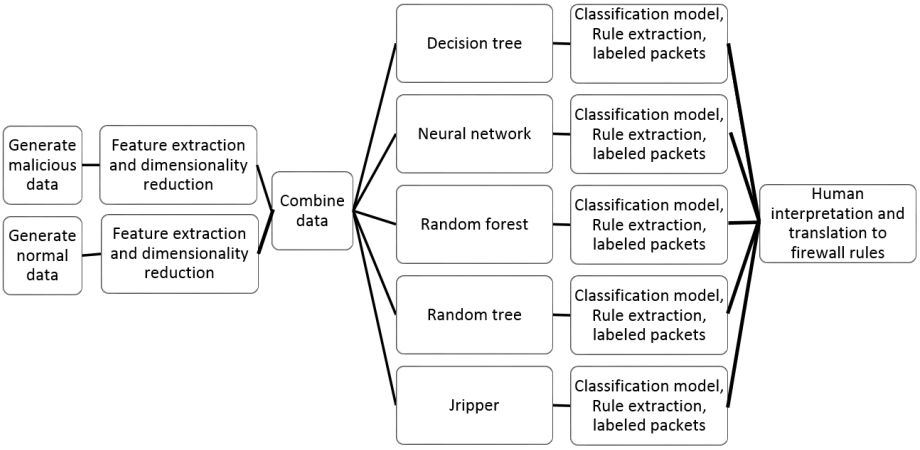

Figure 2: Based on the generated data mathematical models are created that can classify packets. These models can also be visualized. From these visualized models firewall rules can be created.

The main advantage of the machine learning methodology used is that attacks and exploits similar to the ones used in Kali [Kali] or Metasploit [Metasploit] will be detected. Another advantage is that the packets that are similar to normal data will not be marked as an attack thus reducing false positives. It is suspected that many attacks these days are run with tools like Kali or Metasploit. If the attack is not performed by a tool the chance is big that the attack is an altered existing attack from Kali or Metasploit or a similar tool.

3.1 Creating the data set

One way data is created is by placing 2 computers next to each other on a LAN and generating traffic. The attacking computer that is running Kali is called the attacking machine and the computer that is being hacked is called the target machine. The traffic generated needs to be both normal traffic and malicious traffic. To achieve this a specific setup was used. On one laptop the special hacking operating system Kali Linux was installed. This operating system contains the Metasploit framework that contains over 1,200 exploits and an average of 1.2 new exploits are added every day. These exploits can be executed against the target machine. In order for the exploits to work, certain services and programs that contain vulnerabilities need to be running on the target machine. For this, Metasploitable 2.0 is used. A list of some of the vulnerabilities of Metasploitable 2.0 can be read here [vulns]. Some examples of vulnerable applications that run on Metasploitable 2.0 are phpMyAdmin, tikiwiki and webdav.

During the attack of the attacking machine on the target machine, all network data is logged on the target machine. This is done by making tcpdumps [tcpDump] on the target machine. tcpdump is a command in unix that captures all network traffic from a specified ethernet adapter. By setting the -s flag to 0 we make sure that we capture the whole packet (65535 bytes is the maximum size of a packet). When all attacks are run and the PCAP file is created that contains all packets, the PCAP file is loaded into Wireshark to visualize and inspect the data. Wireshark is also used to filter out everything but TCP/IP traffic: all traffic that is not TCP/IP is thrown away. A sample of resulting data is shown in figure 3.

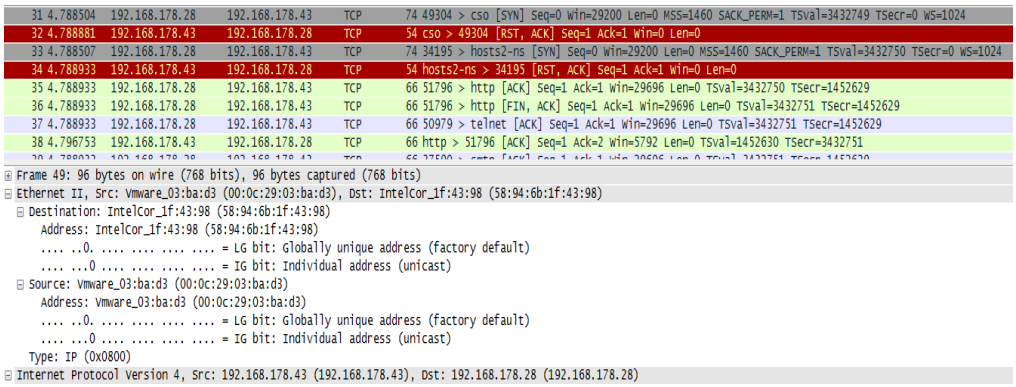

Figure 3: Sample of 38 TCP packets in Wireshark

The normal data is generated by normally using the target machine and perform activities such as web-browsing, downloading files, and e-mailing. In this case the pcap files containing the packets are also created using the tcpdump

command.

3.2 Feature extraction of data

The network traffic that is captured consists of packets. They can be seen as letters being send over a network. These packets also have things like an address and sender. Information like the address, sender, the time the packet was sent and the time to live are in header information. We want to separate the header information from the actual message of the packet. The actual message of a packet is called the payload. Figure 4 displays one packet. In the top of the image one can see all the header information and the payload starts with the blue selected text GET. The blue selection matches the hexadecimal selected bytes and with the corresponding ASCII characters that these bytes represent: 47 45 54. In the ASCII figure 5 we can see that the hexidecimal numbers 47, 45 and 54 indeed correspond to the capital letters G,E and T.

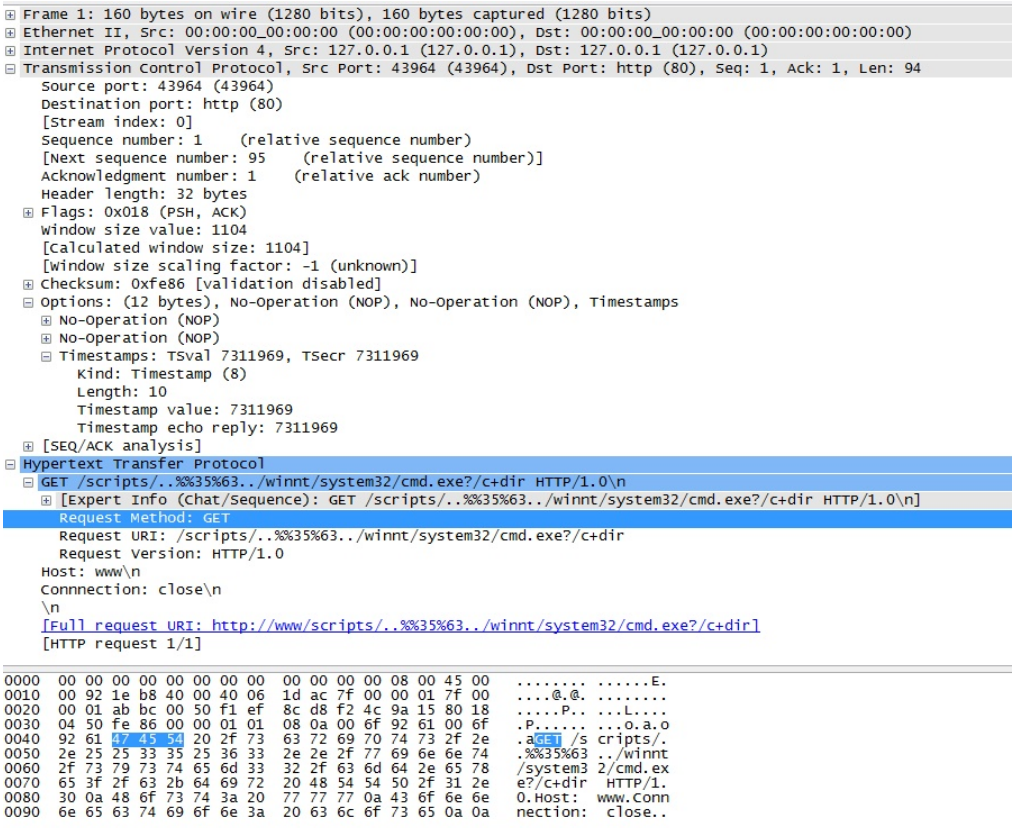

Figure 4: Sample of a single Packet

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Figure 5: ASCII table (Not all 256 bytes but only until 127), Source: [LookupTables]

One way features are extracted from the payload of the packet is by counting the frequency of each possible byte. This is also called a bag of bytes. More advanced methods for the feature extraction are possible and will be explained below. There are 256 possible bytes so that gives 256 numerical features. Next to the features a label is added to specify to which class a packet belongs (malicious or normal). The actual extracting of the payload of the packets and the counting of the bytes is done in a JAVA program. This Java program uses the winpcap library to be able to read in pcap files [Winpcap].

Next to these 256 features other features are extracted from the packet:

- Len: total length of packet
- Length: length of header
- Caplen: amount of actual data which was stored
- Ident: identifier
- Flag: flagged true or false

- StdevTime: The standard deviation of the time the packet itself was received and the last 7 packets send before it. It is a measure of the amount of variation or dispersion there is between the times of the last 8 received packets.

3.2.1 N-grams

A more advanced method to extract features from packets is to use $n-grams$. In this case a Packet has many features; one for each possible $n-gram$. An $n-gram$ is a contiguous sequence of n bytes. The value of the feature is the frequency of occurrence of that $n-gram$. For example, of the 6 consecutive bytes (hexadecimal coding) 00, 01, 02, 03, 04, 05 the 2-gram is 00 01, 02 03, 04 05 and the 3-gram is 00 01 02, 03 04 05. Not only $n-grams$ are used but also skip grams or $n-v-grams$. The difference here is that a space v is left between each pair of bytes. One can imagine this as if a v+2 long sliding window with a gap of length v between the first and last byte that goes over the bytes of the payload. This is illustrated in figure 6 and 7; the pictures are a representation of the payload of a packet.

```
00 60 97 de 54 36 00 10  7b 38 46 32 08 00 45 00
01 48 59 c1 40 00 3f 06  59 0f ac 10 72 94 d1 b9
97 81 7a 81 00 50 9c 8c  d5 88 d5 dc 1f a8 50 18
7d 78 d2 37 00 00 47 45  54 20 2f 69 6d 61 67 65
73 2f 63 68 5f 6d 6f 6e  65 79 2e 67 69 66 20 48
54 54 50 2f 31 2e 30 0d  0a 52 65 66 65 72 65 72
3a 20 68 74 74 70 3a 2f  2f 77 77 77 2e 68 6f 74
62 6f 74 2e 63 6f 6d 2f  0d 0a 55 73 65 72 2d 41
67 65 6e 74 3a 20 4d 6f  7a 69 6c 6c 61 2f 34 2e
30 34 20 5b 65 6e 5d 20  28 58 31 31 3b 20 49 3b
20 4c 69 6e 75 78 20 32  2e 30 2e 33 32 20 69 36
38 36 29 0d 0a 48 6f 73  74 3a 20 73 74 61 74 69
63 2e 68 6f 74 62 6f 74  2e 63 6f 6d 0d 0a 41 63
63 65 70 74 3a 20 69 6d  61 67 65 2f 67 69 66 2c
20 69 6d 61 67 65 2f 78  2d 78 62 69 74 6d 61 70
2c 20 69 6d 61 67 65 2f  6a 70 65 67 2c 20 69 6d
61 67 65 2f 70 6a 70 65  67 2c 20 69 6d 61 67 65
2f 70 6e 67 2c 20 2a 2f  2a 0d 0a 41 63 63 65 70
74 2d 4c 61 6e 67 75 61  67 65 3a 20 65 6e 0d 0a
41 63 63 65 70 74 2d 43  68 61 72 73 65 74 3a 20
69 73 6f 2d 38 38 35 39  2d 31 2c 2a 2c 75 74 66
2d 38 0d 0a 0d 0a
```

Figure 6: The sliding window moves on one step and sees "47, 20". So the frequency of that skip gram is increased with one.

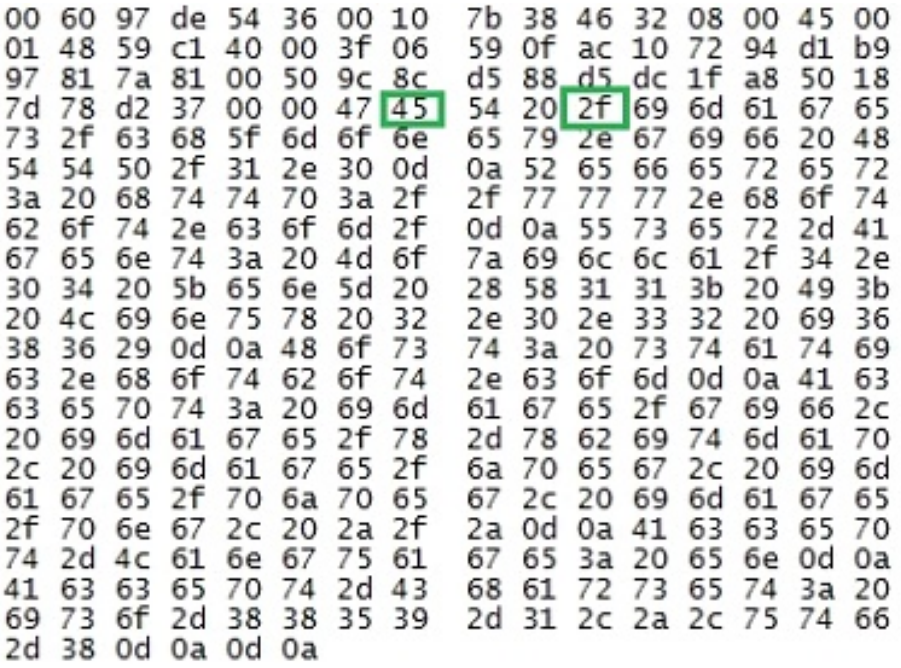

Figure 7: The sliding window sees the 2-2-gram "45,2f". So the frequency of that skip gram is increased with one.

3.2.2 Dimensionality

The reason that gaps were used between the bytes is because the amount of features and thus dimensionality greatly increases with n and it is wanted to extract structural information. The amount of possible features grows exponentially with n. By using a gap v it is still possible to extract some information related to the $n-grams$, with n >2. With n = 2 the amount of dimensions is still large (65536). This means that the frequency matrix will also be that large. The frequency matrix will have a lot of zero values because a lot of skip grams will not occur in the payload. Another common use of $n-grams$ and frequency matrix is in document classification, were documents are classified into categories based on $n-grams$ (word pieces in the document). It has been shown that even if they have to deal with very large amounts of features (more than 10000) and thus more than 10000 dimensions; SVM classifiers still work for text categorization [T. Jaochims]. In this thesis SVMs are used to categorize packets as malicious or normal. SVM classifiers still work in case of high dimensional problems because the documents are very sparse and only the $n-grams$ that occur in the document are considered. You could effectively process feature vectors of 10 000 dimensions, given that these are sparse. Even though skip grams are used, our feature space is big. The dimensionality of the feature space can

be reduced by applying a clustering algorithm. The cluster algorithm used is a clustering algorithm proposed by Dhillon et al in [I. S. Dhillon]. This clustering algorithm is somewhat similar to well-known k-means algorithm [C. Mihaescu].

3.2.3 Clustering

Clustering

The clustering algorithm uses the following as input:

- The occurrence frequency matrix of skip grams.
- The desired number of clusters k. (This is also the number of features that each item will have after clustering. So this is also the number of dimensions you will have).
- The tolerable information loss t.

It then initializes random clusters. This just means that each feature is randomly assigned to one of the k clusters. Each feature is then moved from cluster to cluster until the information loss is less than t. The information loss is calculated with Kullback-Leibler divergence. The Kullback-Leibler divergence of Q from P is a measure of the information lost when Q is used to approximate P. KullbackLeibler divergence measures the expected number of extra bits required to code (code to binary) samples from P when using a code based on Q, rather than using a code based on P. The output of the clustering algorithm are the k clusters, where each cluster is the sum of a set of features (adding up the probabilities). The features themselves are cells from the conditional probability matrix. Each cell in this matrix is still the conditional probability matrix holds the probability that a skip gram occurs with relation to the target class $P(X \mid T)$. So each new feature in the new k-dimensional feature space can be computed in the following way:

$$P(S_{\mathrm{h}}|f_{\mathrm{i}}) = \sum_{X_{\mathrm{j}} \in S_{\mathrm{h}}} P(X_{\mathrm{j}}|f_{\mathrm{i}})$$

Where h = 1,...,k and $P(S_{\mathrm{h}}|F_{\mathrm{i}})$ is the conditional probability that the set of skip grams S occurs in file i. If there are ten features, which all have a chance of 0.1 of occurring in the payload of a packet, and one wants to decrease this to 9 features, 2 features have to be put together in a cluster where the chance of either of the features occurring is the sum of the 2 features.

3.2.4 Combined packets

Because computer intrusions or attacks over a computer network usually consist of more than one packet it is preferred to use groups of packets. This means multiple packets are put into a group and the group of packets is classified as malicious or normal. In our case the group consists of only malicious or only normal packets. Our case is not to be confused with an other method of

grouping packets together called Multiple Instance learning [O. Maron] where groups can consist of malicious and normal packets and a group is malicious if one packet of the group is malicious and normal otherwise. How to create the groups is based on other papers. In [I. Weon] it is suggested to use time related and risk related parameters. It is decided to use a simple time related parameter: timestamps. A packet is stamped with the time at which it arrives at the destination machine. This is done by making bags of all packets that arrived in the first 0.001 seconds, then another bag of all packets in the 0.001 seconds after that etc. A disadvantage of this method is that sometimes the bags only contain 1 instance because there is only one packet in those 5 seconds. But normally the bags are of size of about 8. This was tuned to 8 because in most data sets the attacks were also consisting of about 8 packets. As explained in [I. Weon], many other bag sizes can be used, ranging from 20 to even 50 packets. This might be different for each data set but is interesting for further investigation.

3.3 Train model

The model is trained on the resulting data set. This is a two-class classification problem (malicious and normal data). A simple and easy to understand and visualize way of classifying is to use decision trees. The disadvantage of decision trees is that they are not good at obtaining good classification rates and low false positives. In a first attempt the following decision tree was created as a sample that can be seen in figure 8.

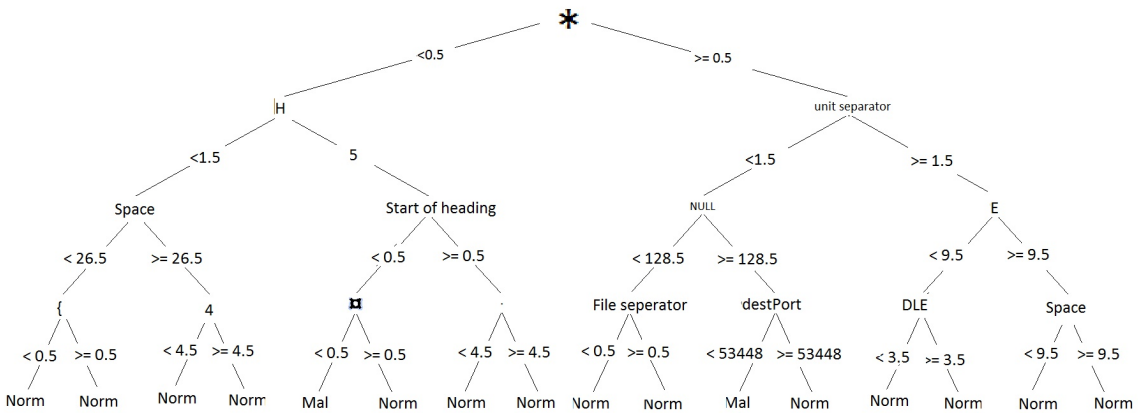

Figure 8: Sample tree

In the tree every node and leaf is numbered, and behind the number of the node and the ":" is a number indicating which byte is used by that node. The frequency of that byte determines what path through the tree should be

followed. This is a simple tree and gives an idea of the technique. However this decision tree did give a recognition rate of 91.5%.

Next to decision trees there are many more machine learning techniques to create a model. Some examples are a neural network, random trees, random forest, and many more.

3.4 Test model

In this chapter some validation methods that are used in this research are explained. In order to test the performance of the model, benchmarking methods are used. In this process, a model is trained and tested using different partitions of the benchmark data. For each sample of the data, the output hypothesis of the model is compared to the corresponding (known) category. It is desired to train our model on a data set with targets and outliers. The resulting model should give as many true positives and true negatives as possible, and as few false negatives and false positives as possible. A false negative occurs if the hypothesis says it should be negative but in fact it is positive (the data sample is normal data but is wrongly classified as an attack). A false positive occurs if the hypothesis says that the data sample is an outlier (attack), whereas in fact it is a target (normal). A true positive is when the hypothesis says it should be a positive and it is in fact a positive. A true negative is when the hypothesis says it should be a negative and it is in fact a negative. A model is tested by counting how many false positives and false negatives there are. As few as possible false positives and false negatives are desired. In the case the model is used directly in practice it is very important to have a very low false positive rate of the data set because in the kind of data sets that is used there are hundreds or thousands of normal data packets. Even if there is a false positive rate of 0.01% there will still be a lot of false alarms going on.

3.4.1 Percentage split

A model such as a decision tree or MLP is a classifier. It is a function that maps an unlabelled instance to a label. The validation is often done on a separate part of the (labeled) data set. The easiest validation method is to use a percentage split. The original data set is split into a training set and a test set. The model is trained on the train set then the test sets without the labels are inputted into the model. After the model is trained the output of the model is compared with the label and the amount of errors are calculated. The percentage of instances that are correctly classified is called the recognition rate. The validation methods need to measure the accuracy of the model (classifier) so that we can compare them. Another validation method is the cross validation method.

3.4.2 Cross validation

Cross validation is done in a number of rounds. At each round the data set is split up in a train set and a test set. The model is trained on the train set

and then tested on the test set, at each round a different part of the data set is the test set. The rounds are often called folds. For example a 10 fold cross validation means splitting the data up in 10 parts and doing 10 rounds. Cross validation is also called the "leave one out method" because at each round you leave one part of the data set out to test on. At the end the average of the performance of all the rounds is calculated.

Figure 9: Illustration of 4 fold cross validation, also called leave one out.

An advantage of cross validation is that one can test and train the model on a limited amount of data. Another advantage is that your test and training is more precise since it was tested and trained on the whole data set. A disadvantage is that at each round a model is trained and tested. The training of the model takes time and at the end all but one of these models are discarded.

3.4.3 Confusion matrix

In this thesis the Confusion matrix is also used as a validation method. A Confusion matrix is a matrix where the each row represents the target class to which a value belongs and each column represent the target class to which it was classified. In this matrix one can easily see if the model is confusing classes. Figure 10 is an illustration of a small example of a confusion matrix.

		Prediction	
		Cat	Dog
Actual	Cat	15	35
	Dog	40	10

Figure 10: Example of small confusion matrix.

If the values on the diagonal are high it means the results are good, if the values on the diagonal are low and the other values high it is an indication that the classifier did not work well.

3.5 Visualize model

The resulting model can be visualized in the form of a decision tree. A human can interpret the results and decide what rules should be extracted and implemented into the firewall. The visualised models are given in Appendix D.

4 Data sets

All data sets were created with the methodology described in the methodology chapter. In this chapter first the motivation for creating the data set with this methodology is described. After the motivation for the data set the descriptions of specific data sets are given.

4.1 The motivation for creating the data set

Before one can create an intrusion detection model data is needed. Data is often hard to obtain because of privacy issues. Not many corporations want to share network traffic data. Network packets often contain private information. Especially labeled data is hard to obtain because labeling the data is labor intensive. Half of the work of this thesis is creating the labeled data set. A quote:

> It is very difficult and expensive to obtain a labeled data set that is representative of real network activates and contains both normal and attack traffic

[J. McHugh].

Obtaining network traffic from a network is hard because not many corporations will want to give data. Other researchers also have trouble with obtaining these data sets. Papers nowadays still discuss data sets that are from the year 1999 and older. The most used data set is the DARPA data set [DARPA]. This data set is from 1999. Of course this 15 year old data set is outdated. The normal traffic is not representative for traffic of these days, also the data of attacks is outdated because different attacks are used these days. Many papers still use this data set. An example of a paper of 2014 that still uses this data set is [K. Rajitha]. Next to the DARPA data set there is also the KDD Cup data set [KDDcup]. The KDD Cup data set is a subset of the DARPA data set. A paper from 2011 that still used this data set is [J. Jonathan], there are even papers published in 2013 and 2014 that use the 1999 KKD cup data set to test models on. Fore example [P. Ahmed], [S. Kumar], and [U. Subramanian]. Many more of these papers can be found, this shows the need of a new data set and how difficult it is for researchers to obtain these data sets.

4.2 Data sets

All data sets consist of 1876 maliciouse packets and 11444 normal packets. In data sets 1,2,3 and 4 the feature extraction of the packets were done with unigram. The unigrams give 256 features. In data set 1 the amount of milliseconds the packets needs to travel, the source port and the destination port were added as a feature. Data set 2 was the same as data set 1 but the following features were added:

- len: total length of packet

- length: length of header
- caplen: amount of actual data which was stored
- ident: identifier
- flag: flagged true or false

Data set 3 is the same as data set 2 but the source port and destination port features were removed. Also a new feature was introduced: the standard deviation of the time between the last 8 packets that were received. This gives information of how fast packets were send after each other. Data set 4 uses combined packets. Each row consists of the last eight packets send. In order to reduce dimensionality not all 256 bytes of each of the eight packets are used. The following features where used for each packet and combined together:

- Bytes 1 and 2
- Bytes 30 trough 63
- Bytes 120 trough 142
- srcPort: source port
- destPort: port destination
- len: length of header
- length: length of packet
- stdevTime: standard deviation between the time of last eight packets
- ident: identifier
- caplen: amount of actual data which was stored
- flag: flagged true or false

Data set 5 is special because it uses the skip grams or n grams discussed in chapter 3.2.2. In this data set a gap size of 1 between each combination of 2 bytes is used. This results in 256 * 256 = 65536 features, this is too much. Therefore dimensionality reduction is used as described in section 3.2.3. In this data set the dimensionality is reduced from 65536 to 260. This is still a high dimensionality for this data set because there are only about 14000 instances. Therefore datas set 6 is created where the dimensionality is reduced from 65536 to 45.

Next to the 260 clusters of skip bytes the following features were added:

- srcPort source port
- destPort port destination

- len: length of header
- length: length of packet
- stdev: standard deviation of time last 8 packets received
- caplen: amount of actual data which was stored

Data set 6 is similar to data set 5 with the only difference that the amount of skip bytes are clustered together to 45 clusters instead of to 260.

Data set 7 is a data set where packets are grouped together. The packets in a group are either all malicious or all normal. Packets are grouped together based on time. The time between the first and the last packet in a group is no more than 10000 ms. Also groups are never bigger than 8 packets.

This is not typical multiple instance learning. In the case of multiple instance learning bags are mixed positive and negative packets and a bag is considered negative if one packet in the bag is negative.

Just like data set 6 this data set also contains 45 features containing information of byte frequency's and the same extra features as in data set 5 and 6 such as source port and length.

5 Performance evaluation

This chapter describes the performance of various machine learning methods with different parameters applied to a number of data sets. In each section a number of experiments, their differences and the performances are discussed. All data sets were created with the methodology described in the methodology chapter. In chapter 4.2 the data sets where also numbered, in this chapter 3 is a reference to these data sets with these same numbers. In column three of each table the number of the data set used is given.

5.1 Depth of random tree

In the first five experiments random trees were used. The depth parameter D indicates how many nodes one needs to follow to get from the first node to the leaf of the tree. In every experiment a different depth was used in order to investigate the effect of the depth on the recognition rate (rr). A small depth generates a smaller tree that is more simple to interpret. A higher depth generates a more complex tree with better recognition rates. For example, with a depth of 9 the recognition rate is 99.744%.

nr	**Method**	**Data set**	**parameter's**	**rr**	**TP**	**FP**
0	Random Tree	1	$D = 4$	94.099%	0.785%	0.035%
1	Random Tree	2	$D = 9$	99.744%	0.997%	0.007%
2	Random Tree	3	$D = 3$	93.210%	0.932%	0.223%
3	Random Tree	3	$D = 4$	94.400%	0.944%	0.157%
4	Random Tree	3	$D = 5$	95.910%	0.959%	0.156%
16	Random Forest	6	$D = 22$	99.825%	0.995%	0.009

Table 1: Performance results of Random Trees. D is the depth of the tree, rr stands for recognition rate.

5.2 Neural networks

In this section neural networks with various parameters and data sets are compared. The recognition rate of experiment five is very low. In the confusion matrix in Appendix E.1 one can see the problem: the model just classified everything as normal data. This is because there is only one hidden layer making it impossible for the neural network to learn the complex patterns. The recognition rate is still 85.6% and not 50% because there are more normal packets than malicious packets in the original data set. In experiment 6 this experiment is repeated without the indent feature and with 2 extra hidden layers. The higher amount of hidden layers results in a better recognition rate of 98.65%.

From experiment 5 it was learned that the ident attribute weighs very heavy. One can see this as a fault in the data set or as "cheating". Ident is probably

unique to many instances, this is not what one wants the model to learn, therefore the ident variable is left out in experiment 6.

In the third column of experiment 5 and 6 it is noted that "66% of data set 3 is used. This just means that 66% of the packets of data set 3 are used. This is because neural networks take a long time to train and test and using the full data set would take even longer.

In experiment 14,15 and 16 the only difference is the amount of hidden layers. If the number of hidden layers is increased the recognition rate also increases. Experiment 20 was run with 7 hidden layers. 7 was chosen because this is a good trade off between the amount of time needed to train and test the model and the recognition rate. The difference between experiment 20 and 15 is the data set. The difference between data set 6 and 7 is that in data set 7 packets are grouped together. The results of these experiments show that grouping the packets together this way increases the recognition rate with 0.32%.

nr	**Method**	**Data set**	**Vars excluded**	**parameter's**	**rr**	**TP**	**FP**
5	Neural network	66% of 3	-	$N = 133$, $L = 1$	85.600%	0.000%	0.055%
6	Neural network	66% of 3	ident	$N = 133$,$L = 3$	98.650%	0.987%	0.077%
14	Neural network	6	-	$N = 27$, $L = 4$	99.051%	0.95%	0.003%
15	Neural network	6	-	$N = 27$, $L = 7$	99.119%	0.957%	0.003%
17	Neural network	6	-	$N = 27$, $L = 10$	99.149%	0.954%	0.003%
20	Neural network	7	-	$N = 27$,$L = 7$	99.151%	0.992%	0.036%

Table 2: Performance results. N is the number of nodes per hidden layer, L is the number of hidden layers and D is the depth of the tree, rr stands for recognition rate.

5.3 Grouped packets

Experiment 19, 20 and 21 were all run on data set 7. In data set 7 packets are grouped together. In the table 5.3 the results of the learning algorithms applied on data set 7 are shown (experiment 13, 15 and 22). Next to the results of algorithms applied to data set 7 the results of machine learning algorithms applied to data set 6 are also shown. Data set 6 is the same as data set 7 with the only difference that the packets are not grouped. It can be seen that our way of grouping packets together does not increase performance in the case of random trees. The performance of the Neural Network is increased in performance with 0.32% when packets are grouped together. In the case of the random forest there is a small increase of 0.001% when packets are grouped together.

nr	Method	Data set	Vars excluded	parameter's	rr	TP	FP
13	Random tree	6	-	$D = 9$	99.006%	0.956%	0.005%
20	Random tree	7	-	$D = 9$	98.970%	0.954%	0.004%
15	Neural network	6	-	$N = 27$, $L = 7$	99.119%	0.957%	0.003%
20	Neural network	7	-	$N = 27$,$L = 7$	99.151%	0.992%	0.036%
22	Random forest	6	-	$D = 9$	99.741%	0.997%	0.01%
21	Random forest	7	-	$D = 9$	99.742%	0.997%	0.01%

Table 3: Performance results. N is the number of nodes per hidden layer, L is the number of hidden layers and D is the depth of the tree, rr stands for recognition rate.

5.4 Jripper

Experiments 8, 9, 12 and 18 are performed with Jripper. All results are very good, it appears that Jripper is the best overall performer of all methods tested. The best performance by Jripper is when it is applied to data set 5, here the skip gram is used. The second best is experiment 6, here also skip grams are used, but there are only 45 clusters used instead of 260. It is logically expected that when the amount of clusters is reduced the performance is lower because one throws information away.

nr	Method	Data set	Vars excluded	parameter's	rr	TP	FP
8	Jripper	1	-	-	99.804%	0.998%	0.006%
9	Jripper	1	src and dest port	-	98.678%	0.987%	0.038%
12	Jripper	5	-	-	99.835%	0.994%	0.001%
18	Jripper	6	-	-	99.704%	0.985%	0.001%

Table 4: Performance results of Jripper. rr stands for recognition rate.

Table 5 shows all the results together.

nr	Method	data set	Vars excluded	parameter's	rr	TP	FP
0	Random tree	1	-	$D = 4$	94.099%	0.785%	0.035%
1	Random tree	2	-	$D = 9$	99.744%	0.997%	0.007%
2	Random tree	3	-	$D = 3$	93.210%	0.932%	0.223%
3	Random tree	3	-	$D = 4$	94.400%	0.944%	0.157%
4	Random tree	3	-	$D = 5$	95.910%	0.959%	0.156%
5	Neural network	66% of 3	-	$N = 133$, $L = 1$	85.600%	0.000%	0.055%
6	Neural Network	66% of 3	ident	$N = 133$,$L = 3$	98.650%	0.987%	0.077%
7	Random tree	4	-	$D = 3$	95.872%	0.987%	0.077%
8	Jripper	1	-	-	99.804%	0.998%	0.006%
9	Jripper	1	src and dest port	-	98.678%	0.987%	0.038%
10	Neural network	5	-	$N = 133$, $L = 3$	98.717%	0.947%	0.006%
11	Random tree	5	-	$D = 9$	98.975%	0.939%	0.002%
12	Jripper	5	-	-	99.835%	0.994%	0.001%
13	Random tree	6	-	$D = 9$	99.006%	0.956%	0.005%
14	Neural network	6	-	$N = 27$, $L = 4$	99.051%	0.95%	0.003%
15	Neural network	6	-	$N = 27$, $L = 7$	99.119%	0.957%	0.003%
16	Random forest	6	-	$D = 22$	99.825%	0.995%	0.001%
17	Neural network	6	-	$N = 27$, $L = 10$	99.149%	0.954%	0.003%
18	Jripper	6	-	-	99.704%	0.985%	0.001%
19	Random tree	7	-	$D = 9$	98.970%	0.954%	0.004%
20	Neural network	7	-	$N = 27$,$L = 7$	99.151%	0.992%	0.036%
21	Random forest	7	-	$D = 9$	99.742%	0.997%	0.01%

Table 5: Performance results. N is the number of nodes per hidden layer, L is the number of hidden layers and D is the depth of the tree, rr stands for recognition rate.

6 Resulting rules

In this chapter methods are compared and resulting rules that can be drawn from the experiments are discussed.

It is clear that more complex models have better recognition rates. Neural networks also work well (the recognition rates are over 98.6%).

After looking at the the images in the Apendix E it appears that the important ASCII characters are 0, 8, 12, 32, 39, 43, 50, 58, 67, 73, 98, 110, 115, 142, 237, 250. With important it is meant that

Some resulting Jripper rules from experiment 8:

$(destPort \leq 56118) and (srcPort \leq 55177) and (3 \leq 1) \Rightarrow category = mal(1513.0/0.0)$

$(0 \leq 0) and (53 \leq 3) and (50 \geq 10) and (47 \geq 7) \Rightarrow category = mal(101.0/0.0)$

$(destPort \leq 55757) and (srcPort \geq 57277) \Rightarrow category = mal(78.0/0.0)$

$(destPort \leq 55757) and (srcPort \leq 56118) and (0 \geq 12) \Rightarrow category = mal(87.0/2.0)$

$(destPort \geq 57314) \Rightarrow category = mal(36.0/0.0)$

$(0 \leq 0) and (58 \leq 4) and (srcPort \geq 6667) and (77 \geq 1) \Rightarrow category = mal(22.0/1.0)$

$(0 \leq 0) and (55 \leq 1) and (87 \geq 4) and (1 \leq 0) \Rightarrow category = mal(9.0/0.0)$

$(srcPort \geq 8080) and (srcPort \leq 55757) and (99 \geq 5) and (132 \leq 3) \Rightarrow category = mal(13.0/1.0)$

Some Resulting Jripper rules from experiment 9:

$(0 \leq 0) and (42 \leq 0) and (85 \geq 1) and (46 \geq 7) and (40 \geq 1) \Rightarrow category = mal(1171.0/1.0)$

$(44 \leq 0) and (3 \leq 1) and (10 \leq 0) and (32 \geq 1) and (34 \leq 0) \Rightarrow category = mal(291.0/5.0)$

$(123 \leq 0) and (58 \leq 5) and (3 \leq 1) and (115 \geq 4) \Rightarrow category = mal(111.0/15.0)$

$(44 \leq 0) and (3 \leq 1) and (60 \leq 0) and (68 \leq 0) and (0 \geq 3) \Rightarrow category = mal(107.0/10.0)$

$(58 \leq 1) and (3 \leq 0) and (255 \leq 0) and (0 \leq 18) and (10 \leq 0) \Rightarrow category = mal(71.0/10.0)$

Some resulting Jripper rules from experiment 19:

$(23 \leq 21) and (23 \leq 5) and (destPort \leq 56553) \Rightarrow category = mal(1257.0/0.0)$

$(23 \leq 23) and (lenght \geq 315) and (destPort \leq 56915) \Rightarrow category = mal(147.0/1.0)$

$(stdevTime \geq 14813) and (23 \leq 17) and (srcPort \leq 55609) and (0 \leq 0) and (srcPort \leq 6667) \Rightarrow category = mal(106.0/1.0)$

Rules can also be created from the models that resulted from the experiments. One must first interpet the visualised model, For example, in the resulting model of Figure 8 one can derive the following rule: If ascii character "*" occurs less than 0.5 times *and* character "H" occurs more than 5 times *and* the "start of heading" occurs more than 0,5 times the packet is normal.

7 Conclusion

The research question was :

Can one use machine learning techniques to build a model that differentiates between malicious and normal traffic, extract a set of rules from these models and then create useful firewall rules from this set that can be used in a real environment?

The answer to the research question is yes. In this thesis a data set with malicious and normal traffic was created as described in chapter 3.1. Chapter 3 also describes and defines the difference between malicious and normal traffic for this thesis. After these feature extraction steps where done and machine learning techniques were applied to this pre-processed data. The resulting models lead to rules as described in chapter 6.

In this conclusion it is assumed that the data set is realistic and corresponds to a real environment. Compared to the KDD [KDDcup] data set this data set is very realistic, the data set used in the KDD cup is a simulated network and is from the year 1999. For future work I suggest to repeat the technique and experiments on other bigger labeled data sets from various environments. When this is done resulting models, rules and recognition rates can be compared.

Another conclusion that can be drawn from the research done in this thesis that the high amount of false positives is a problem. Using the models described in this thesis on normal personal computers or in the office environment is not realistic at this time. If a program with such a classificaton model would be installed on a personal computer of a normal person that uses it in everyday life it would not work. The false positives and false negatives are still too high. What false positive rate is desired is different for different applications. If for example 1000 packets travel over a small network every day a false positive rate of 0.0001 is manageable. You would have one false alarm every day on average. It might be worth it to develop a machine learning algorithm for very specific goals. For example, in SCADA networks it could work because the network traffic is more constant (see Appendix A).

Appendices

A Creating firewall rules for SCADA networks

A more specialised area of applying machine learning techniques to create firewall rules is the area of SCADA networks. However, because SCADA network traffic is more similar and constant machine learning methods will work better here. This is one reason a special SCADA chapter is included in this thesis but there are more reasons why developing IDS in SCADA networks should be investigated.

In our commercial world Supervisory Control and Data Acquisition (SCADA) systems are often modernized and connected to the Internet to reduce costs and increase efficiency [R. Williams]. This introduces the risk of being targeted by cyber criminals and even governments[C. Saeed]. Cyber criminals keep trying to compromise the integrity, confidentiality or availability of SCADA Systems. Cyber criminals keep using the same basic attack techniques [M.v.j.] but the amount and diversity of malware grows [M. Fossi]. This renders security defenses ineffective such that millions of computer networks and also SCADA networks connected to the Internet are infected with malicious software. Governments attack SCADA systems as part of their cyber warfare. One way governments do this is by hiring hackers [Infosecurity], another way is by putting together teams of cyberwarriors [Army magazine]. The hackers spy, or sabotage SCADA systems. A well known example of this is the U.S. and possibly Israel attacking the centrifuges in a Iranian uranium enrichment facility[C. Saeed]. This chapter explains how and why machine learning methods are useful in IDS on SCADA networks and presents an idea how to create a labeled data set with malicious SCADA network traffic and normal SCADA network traffic.

A.1 Introduction

A.1.1 SCADA

SCADA systems collect, transmit, process and visualize measurement and control signals of remote equipment. SCADA sytems are deployed in many critical infrastructures such as power generation, public transport and industrial manufacturing [A. Nicholson].

A.1.2 Number of SCADA systems connected to the Internet

The precise number of SCADA systems that are connected to the Internet is unknown. Researchers from Free university used Shodan find SCADA systems connected to the internet worldwide[Cyber arms]. On Shodan one can search for computers connected to the Internet based on software, geography, operating system, IP address, manufacturer and other properties. Shodan is also called the Google for hackers because one can use it to find specific vulnerable computers connected to the Internet. The researchers from Free university published the

figure 11 showing the SCADA systems created by German manufacturers that are connected to the Internet.

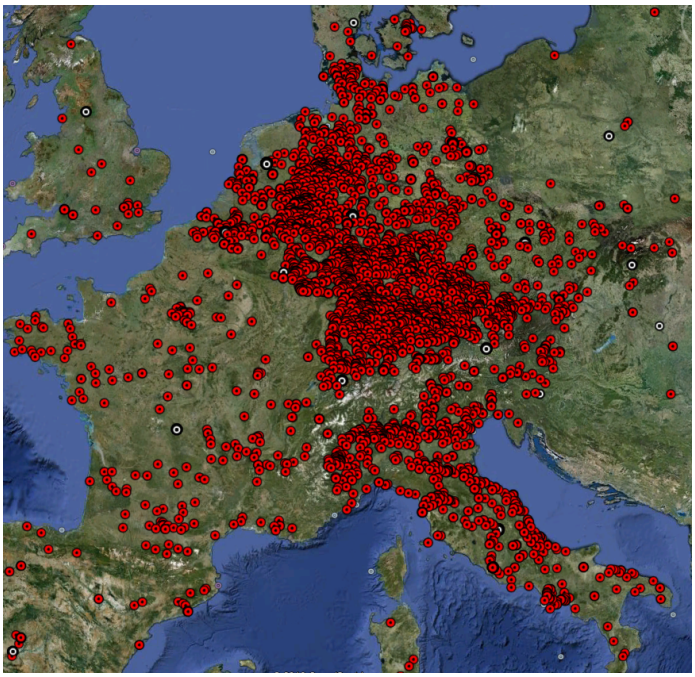

Figure 11: Amount of SCADA systems of German manufacturer connected to the Internet in Europe. Source: [Cyber arms]

If one is a clever searcher and uses for example manufacturers names and specifies a country or region parameter one can find vulnerable SCADA systems nearby. Table A.1.2 is created to give an idea what SCADA systems and how many one can be find on the internet. The table tells how many SCADA systems of a manufacture can be found on one day (16th of April 2014) in the Netherlands alone:

Search query	Info	n-results
PLC country:NL	programmable logic controller (PLC)	40
Allen Bradley country:NL	Allen Bradley manufacturers PLC	2
Rockwel country:NL	automation company	2
Citect country:NL	company creates SCADA systems	2
RTU country:NL	interfaces real objects with SCADA systems	30
Modbus Bridge country:NL	device that connects Modbus serial products	1
telemetry gateway country:NL	Building Automation and Control Network (BACNET)	1
Simatec country:NL	a PLC manufractured by Siemens	30
Siemens -...er -Subscriber	part of ISP infrastructure (Siemens)	301
Schneider country:NL	Energy infrastructure management systems	6

Table 6: Table showing how many SCADA devices where found (n-result) for each search query in the first column

Figure 12: Example of vulnerable PLC

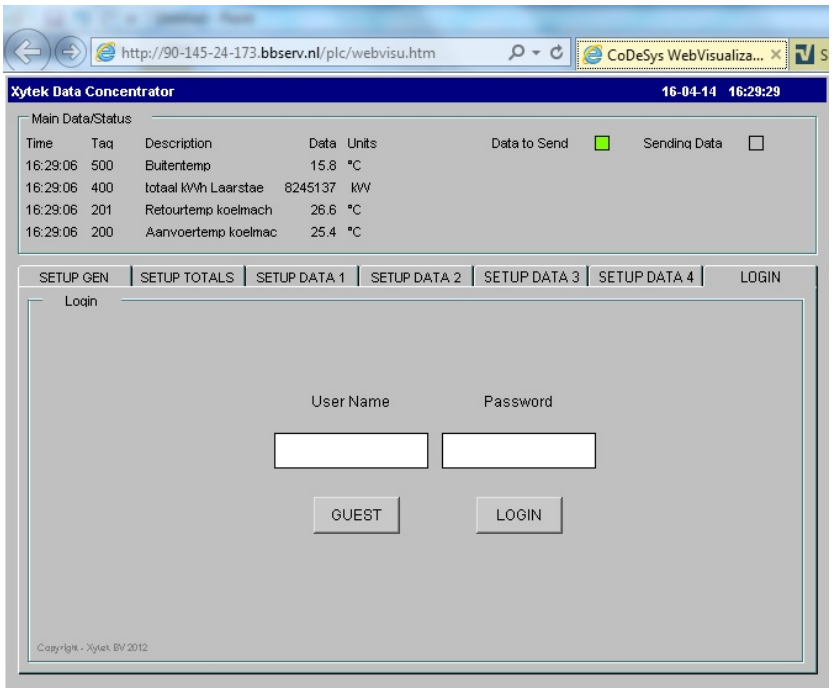

Figure 13: PLC that is vulnerable and accessible

A.2 Obtaining the data set

Before one can create a intrusion detection model data is needed. Data is often hard to obtain because of privacy issues. Network packets often contain private information. Especially labeled data is hard to obtain because labeling the data is labor intensive. Half of the work of this thesis is creating the labeled data set. A quote:

> It is very difficult and expensive to obtain a labeled data set that is representative of real network activates and contains both normal and attack traffic

[J. McHugh].

The work of creating the data set can be split up in 2 parts:

1. The actual obtaining of network traffic on a SCADA network and capturing raw packets.
2. Processing the data set: removing mistakes, labeling and doing feature extraction.

Part one: obtaining network traffic from a SCADA system is hard because not many corporations will want to give data. Also other researches have trouble with obtaining these data sets. Papers of this time still discuss data sets that are from the year 1999 and older. The most used data set is the DARPA data set [DARPA]. This data set is from 1999. Of course this 15 year old data set is outdated. The normal traffic is not representative for traffic of these days, also the data of attacks is outdated because different attacks are used these days. Many papers still use this data set. An example of a paper of 2014 that still uses this data set is [K. Rajitha]. Next to the DARPA data set there is also the KDD Cup data set [KDDcup]. The KDD Cup data set is a subset of the DARPA data set. A paper from 2011 that still used this data set is [J. Jonathan], there are even papers published in 2013 and 2014 that use the 1999 KKD cup data set to test models on. Fore example [P. Ahmed], [S. Kumar], and [U. Subramanian]. Many more of these papers can be found, this shows the need of a new data set and how difficult it is for researchers to obtain these data sets.

A.3 Sample attack (Kingview)

This chapter gives an idea of how the data set with malicious instances can be generated. It will be explain by giving a example of how malicious network traffic can be created by interacting with the Kingview software.

Kingview is a program that that can be installed on MAC OSX or windows. In our example we will install Kingview on windows XP. Kingview is software that is used worldwide for SCADA applications. A description of Kingview from the developers website:

> KingView software is a high-performance production which can be used to building a data information service platform in automatic field. KingView software can provide graphic visualization which takes your operations management, control and optimization. KingView is widely used in power, water conservancy, buildings, coalmine, environmental protection, metallurgy and so on.

The machine to be exploited should be installed first. This is a small network with a computer with Kingview installed on it. This is a small network with its own Internet connection. The machine on which Kingview is installed runs on windows xp sp3 and has some other software installed.

On a other machine Kali [Kali] will have to be installed which includes Metasploit [Metasploit]. The ruby and python exploit modules will be downloaded from [Symantec]. These modules will be placed in the applicable module directory of Metasploit and then used. The module exploits a buffer overflow in Kingview 6.53. By sending a specially crafted request to port 777. The payload of the specially crafted request is depicted in figure 14.

It must be noted here that the exploit has to run about 10 times before it succeeds. After having success one has the meterpreter shell and control of the KingView Server.

```
import os
import socket
import sys

host = sys.argv[1]
port = int(sys.argv[2])

print " KingView 6.53 SCADA HMI Heap Smashing Exploit "
print " Credits: D1N | twitter.com/D1N "

shellcode = ("\x33\xC0\x50\x68\x63\x61\x6C\x63\x54\x5B\x50\x53\xB9"
"\x44\x80\xc2\x77"
"\xFF\xD1\x90\x90")

exploit = ("\x90" * 1024 + "\x44" * 31788)
exploit += ("\xeb\x14") # our JMP (over the junk and into nops)
exploit += ("\x44" * 6)
exploit += ("\xad\xbb\xc3\x77") # ECX 0x77C3BBAD --> call dword ptr ds:[EDI+74]
exploit += ("\xb4\x73\xed\x77") # EAX 0x77ED73B4 --> UnhandledExceptionFilter()
exploit += ("\x90" * 21)
exploit += shellcode

print "  [+] Herrow Sweeping Dragon..."
print "  [+] Sending payload..."

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host,port))
s.send(exploit)
data = s.recv(1024)

print "  [+] Closing connection.."
s.close()
print "  [+] Done!"
```

Figure 14: Kingview Heap Overflow [Symantec]

The network traffic will be captured with Wireshark [Wireshark]. Wireshark will be installed on the target machine or on a computer attached to the network. This is a network monitoring tool. We will be able to capture all network packets that are part of the attack (malicious) by looking at the time the packets were send (we know the exact time since I execute the attack).

In figure 14 one can see the payload of the packet that Wireshark captures. In this message one can see shellcode. This is typical for an attack and is an example of a pattern that could be learned by a machine learning algorithm. (this will be further explained later on)

A.4 Project determination and specification

It is important to determine and specify the project. This chapter describes the project and explains how the project's boundary's are determined.

A important reason for determining the project and its boundary's is the Data that is needed for the project. Raw network data is hard to obtain. If raw data cannot be obtained one alternative solution is to not use all the network data but only look at data flows. In the data flow approach the flow of data through the network is analyzed, instead of the contents of each individual

packet. This way privacy is preserved. Another advantage of using data flows is that it can be performed at high-speeds. A disadvantage is that this throws away allot of information.

There are various possibilities to create intrusion detection systems. Below is a summation of some options. For each option the data that is required is described and some possible ways of how to obtain it. The advantages and disadvantages are also discussed.

1. **SCADA system with labeled data** SCADA systems have more regular data. When the data is more regular it is easier to distinguish between normal traffic and intrusions. Having labeled data also makes it easier to train on the malicious traffic. Labeled data is probably hard to come by because labeling the data is labor intensive. Another problem is that data also varies for each SCADA network.

2. **Anomaly detection on SCADA system** Anomaly detection is less difficult than intrusion detection. Intrusion detection is a one class machine learning problem. This means that one learns only on normal data. When new network traffic is inputted into the model the similarity of that traffic with the normal traffic is measured. If the dissimilarity is above some configured threshold an alarm goes off. No labeled data is needed. One only needs normal traffic from a SCADA network. Obtaining normal data might still be a challenge because of privacy issues.

3. **Create own labeled data on SCADA system** Creating a labeled data set is also challenging because one would have to attack or infect an existing SCADA system. The attacks could be performed with existing tools such as Kali [Kali]. Kali contains a lot of programs and scripts These programs and are also altered and then used to perform attacks.

4. **Labeled data from some network** When using just any network it is hard to have a high recognition rate. On normal big networks there is a high degree of traffic going over the lines with much variation. One problem is that this causes high false positive rates. If one can create some model that has a false positive rate of 0.001% this still means that alarms are going off all day because so many network traffic is generated.

5. **Create own labeled data on some network** One could easily setup some network of some computers with different operating systems, install programs and generate network traffic. This network could then also be attacked by the scrips that are in Kali. One could track the packets caused by the attackers from Kali and this way create a labeled data set. The disadvantage of this approach is that the real data is not that real (its "simulated") and you will only detect attacks that are similar to the attacks performed by scripts in Kali.

B List of abbreviations

NN	Neural network
ML	Machine learning
IDS	Intrusion detection systems
SCADA	Supervisory control and data cquisition
MLP	Multiplayer perceptron
PA	Pattern associator
LAN	Local area network
PCAP	Packet capture
ASCII	American Standard Code for Information Interchange
rr	Recognition rate

C Summary of send documents (21 july)

The document "Application of machine learning technique for Arcsight" contains a short description of how machine learning techniques can be used to create new firewall rules for Arcsight. Machine learning is a technique that can learn from data. In this case the machine learning technique learns a model that can distinguish between malicious and normal network traffic. This is done by looking at a data set of normal traffic and malicious traffic where you know what data is malicious and what data is normal (reinforced learning). The model that is generated by the machine learning algorithm can be visualised and interpreted by humans to be used to create more complex and better firewall rules. This results in lower false positive rates,lower false negative rates, higher true positive rates and higher true negative rates. It also gives the ability to measure, and show, how well the rules that you have generated or currently have work (by for example, creating ROC curves or calculating the false positive rates).

The document "Master Final Thesis draft" is my master thesis project so far, it is not expected to be read entirely but parts can be read of something is unclear or are found interesting by the reader. This document also contains chapters on SCADA systems because the initial plan was to use it in the SCADA system environment.

D Application of machine learning technique for ArcSight

D.1 Introduction

This is a separate document from the actual thesis. In the actual thesis a machine learning method is described that creates a model that can discriminate between malicious and normal traffic. This machine learning method is trained on a specific data set that contains normal traffic and malicouse traffic. In this document it is explained how the resulting model of the machine learning method can be visualised and used to create firewall rules that can be used in ArcSight. This document is more practical and contains less technical details. After the introduction the relevant parts of the machine learning method and visualisation technique will be described in Appendix A.2. In Appendix A.3 it is explained where the firewall rules can be inserted into ArcSight. Finally some remarks are given in Appendix A.4

D.2 The machine learning and visualisation method

Machine learning concerns the construction and study of systems that can learn from a data set. In this case the data set consists of 2 types of network traffic: malicious network traffic and normal network traffic. Network traffic is not a continuous flow of data but consists of packets being send one after another. These packets can be visualised with Wireshark, see figure 3.

The data set is labeled, so it is known for all the network traffic in the data set whether it is either malicious network traffic or normal network traffic. The malicious traffic is generated by performing attacks with Metasploit on a machine with Metasploitable installed. We want the machine learning algorithm to learn a model that distinguishes between malicious or normal traffic when unlabeled network traffic (network traffic of which it is not known whether it is malicious or normal) is handed to the model. The machine learning algorithm learning a model is called the training phase, and is phase 1 in figure 15. Some examples of machine learning algorithms that can create such model are "random forests","neural networks", "multiple instance diverse density", or decision trees". In phase 2 called "the test phase" the models performance can be tested. There are many techniques for testing the performance, some example of measurements of the model performance are the false positive rate, true positive rate and ROC curves. If the performance is not as desired the model created in the training phase should be further optimized. In phase 3 the model is visualised and interpreted by a human that can create advanced firewall rules for ArcSight.

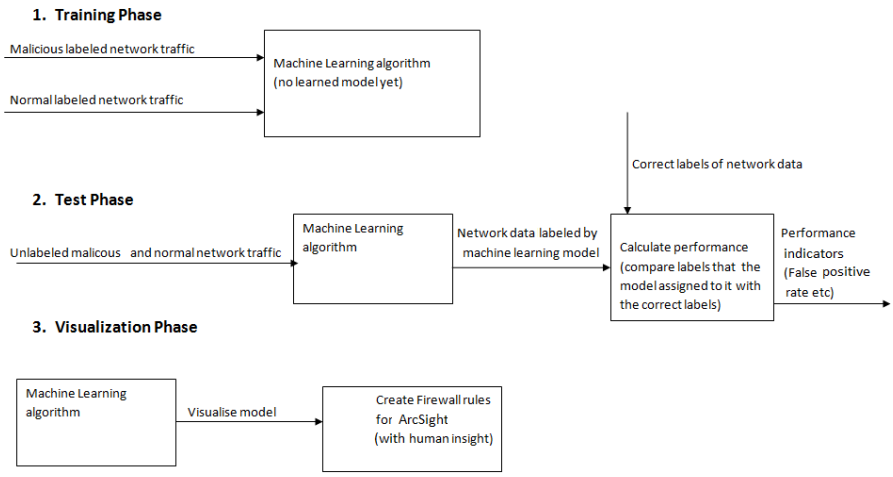

Figure 15: 3 Phases to create new firewall rules

Traditionally network security analysts looked at patterns in connections from certain IP addresses with histories of intrusive behavior. However, intrusions have become more complex. For example, intrusions can be low and slow which means that an attack consists of intrusive behavior over hours, days or weeks and they can have more than one network source. Machine learning can be used to help the data analyst by doing not only simple but also complex pattern recognition. An example of simple pattern recognition is looking at the IP source address or the number of connections on a certain port. A complex pattern can be derived from a visualised model constructed with the method explained above.

The tree is created with WEKA which can be downloaded here: www.cs.waikato.ac.nz/ml/weka . In the attachment is a file called dataset1.arff, this is a sample data set that can be loaded into weka. After one finished downloading weka one can re-create the tree above in the following steps:

1. open Weka, click explorer
2. Click "open file" and select dataset1.arff (in attachment)
3. Click the Classify tab
4. Click choose, and under trees select RandomTree
5. Click on the bold letters "Random Tree" that just appeared next to "choose"
6. Set MaxDepth to 3
7. Click Start

8. In the white area below where you just click start some text saying something like " trees.RandomTree" appeared, right click it and click visualise tree.

Network data is not a constant flow of data but consists of network packets. For each packet or for a group of packets one could determine whether the packet or the group of packets is malicious or not. For simplicity in the example we determine per packet whether it is maliciouse or not. In the tree every node and leaf is numbered, and behind the number of the node and the ":" is a number indicating which byte is used by that node. The frequency of that byte determines what path through the tree should be followed. This is a very simple tree and this tree is not usable yet but gives an idea of the technique. However this decision tree did gave a recognition rate of 91.5% on the test set. If one has a packet where the "*" character occurs more than 0.5 times, the unit separator occurs less than 1.5 times, the NULL character occurs more than 128,5 times and the destination port is smaller than the packet is classified as malicious. This can be seen by starting at the top of the tree, and following the decisions downward to the right leaf.

Using machine learning algorithms to detect intrusions has several advantages; it can detect zero-day malware because it can determine the statistical likelihood that a program is malware based on previous examples.

D.3 The resulting firewall rules in ArcSight

From the model shown in 8 firewall rules can be created. In order to explain how to create these rules, first examples of currently existing firewall rules in ArcSight are shown below.

1. Rule Name: High Number of Connections Rule Desc: This rule detects firewall accept events for MSSQL, Terminal Services, and TFTP connections (default destination ports: MSSQL=1433, Terminal Services=2289, TFTP=69). The rule triggers when ten events from the same device occur within 2 minutes. Rule Conditions:

   event1 : (Type = Base AND Category Behavior = /Access AND Category Device Group = /Firewall AND Category Object = /Host/Application/Service AND Category Outcome = /Success AND (Target Port = 1433 OR Target Port = 3389 OR Target Port = 69) AND NotInActiveList("Event-based Rule Exclusions"))

2. Rule name: Possible Internal Network Sweep Rule Description: This rule detects a single host trying to communicate with at least ten other hosts on the same target port within the network, within a minute. This rule, combined with a spike in target port activity by the same host, results in the worm outbreak detected rule being triggered. Rule Conditions: 10 matches in 1 minute of events. The 10 events that have different target address but same target port and source address.

3. Data Monitor Name: Covert Channel DM Desc: This data monitor displays event information indicating that there is a covert channel. Port 53 is a well-known port for DNS, but DNS activity is generally UDP. Such activity can be correlated with covert channels. DM uses a filter that looks for Destination Port = 53 and Transport Protocol = TCP.

If one looks at the visualised model one could use common sense and create more of these rules. These rules can be much more advanced and complex. These complexer rules will lead to less false positives and less false negatives. This makes a better firewall. How good these rules are can be shown and proven with statistics (phase 2 in 15). Without the machine learning certain complex rules will never be thought off because creating these rules is a very complex and difficult task.

D.4 Remarks

There are some remarks that need to be kept in mind when using this technique in practice:

1. The data set containing normal and malicious packets on which machine learning methods are applied and on which the model is trained is very important. The machine learning method will learn to differentiate between these malicious and normal packets. In this project malicious packets are generated by performing attacks with Metasploit on a machine with Metasploitable installed. Normal data is just dumps of capture of normal network traffic generated by browsing the Internet, downloading some files, reading mail and playing a video game online. So the model will learn to differentiate between these 2 different data sets.

2. The rules have to be derived from the visualised model. Human interpretation is needed in this process.

3. The model that is created should be tested, this is done by calculating false positive and false negative rates on a test set. This gives an indication how good the model is.

E Visualised experiments

E.0.1 Experiment 0

	mal	norm
classified as mal	1497	379
classified as norm	407	11037

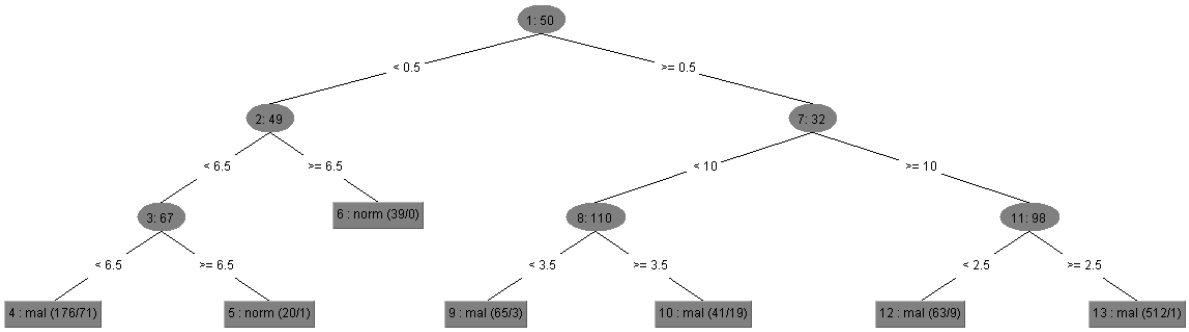

Figure 16: Resulting tree of Experiment 0

E.0.2 Experiment 1

	mal	norm
classified as mal	1835	15
classified as norm	19	11395

The tree of experiment 1 is omitted because it is too large.

E.0.3 Experiment 2

	mal	norm
classified as mal	1396	472
classified as norm	431	11012

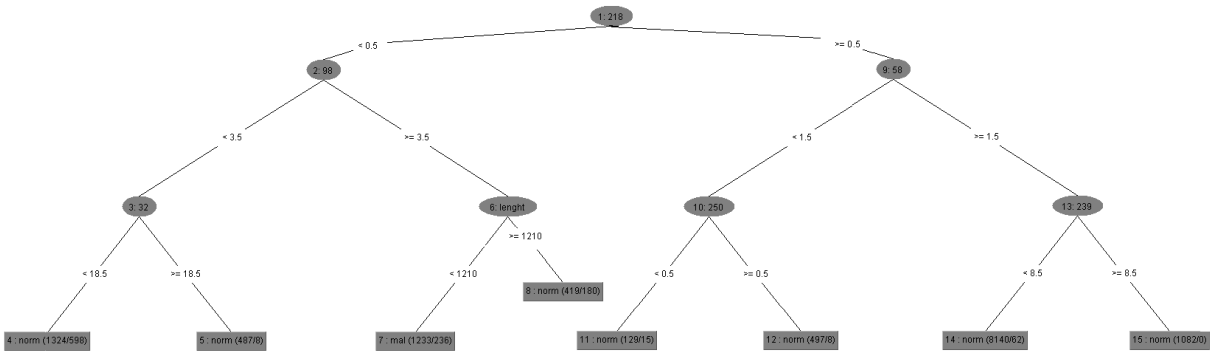

Figure 17: Resulting tree of experiment 2

E.0.4 experiment 3

	mal	**norm**
classified as mal	1537	4331
classified as norm	415	11028

The tree of experiment 3 is omitted because it is too large.

E.1 Experiment 4

	mal	**norm**
classified as mal	1535	333
classified as norm	212	11231

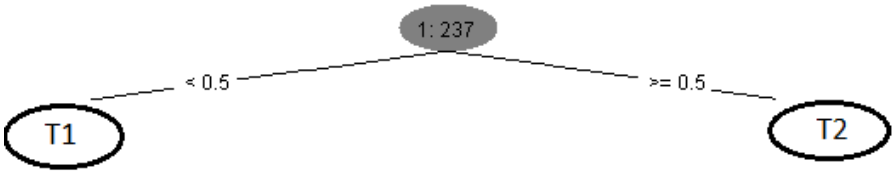

Figure 18: Part 0 of the resulting tree of experiment 4

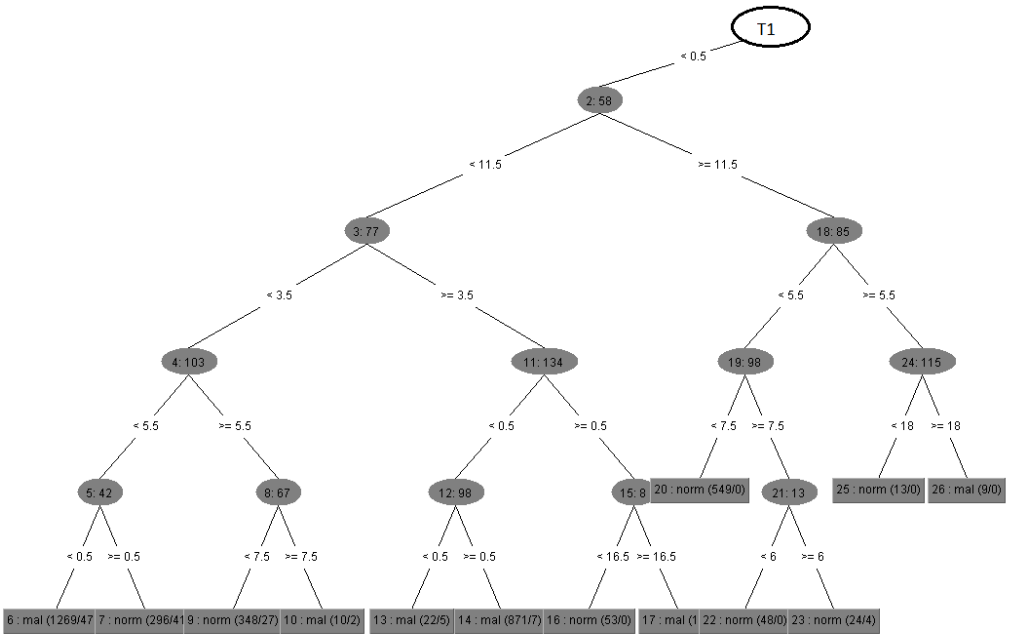

Figure 19: Part 1 of the resulting tree of experiment 4

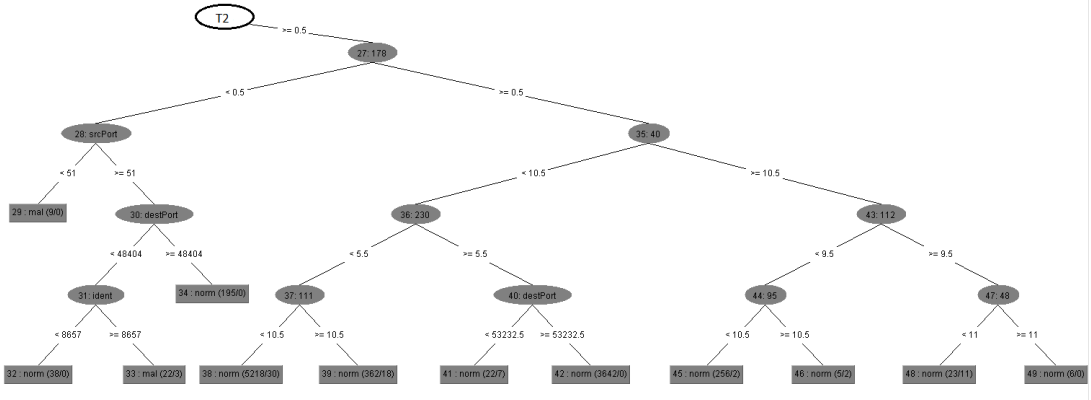

Figure 20: Part 2 of the resulting tree of experiment 4

Confusion matrix:

	mal	norm
classified as mal	0	648
classified as norm	3878	11231

E.1.1 experiment 6

Neural network:

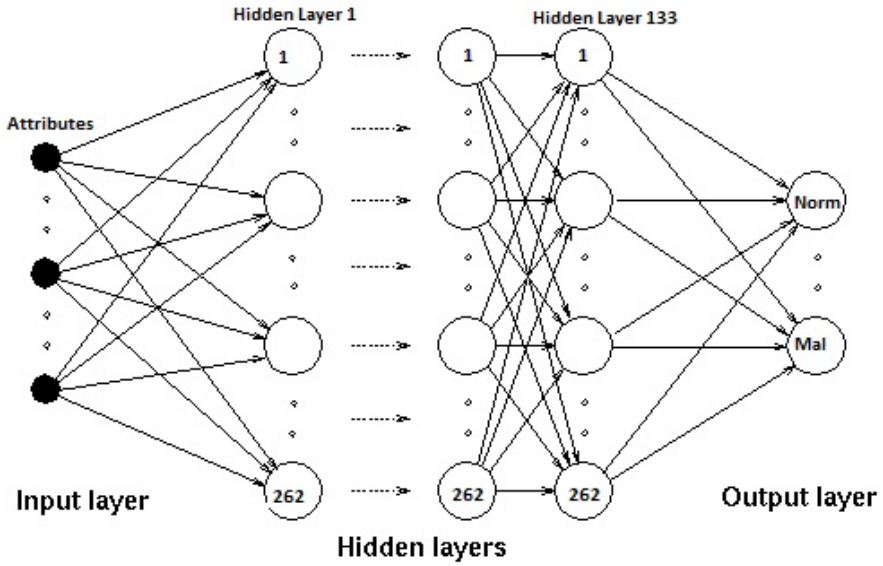

Figure 21: Neural network of experiment 6

The confusion matrix:

	mal	norm
classified as mal	590	58
classified as norm	3	3875

E.2 Experiment 7

Confusion matrix:

	mal	norm
classified as mal	1554	340
classified as norm	211	11253

Resulting Tree:

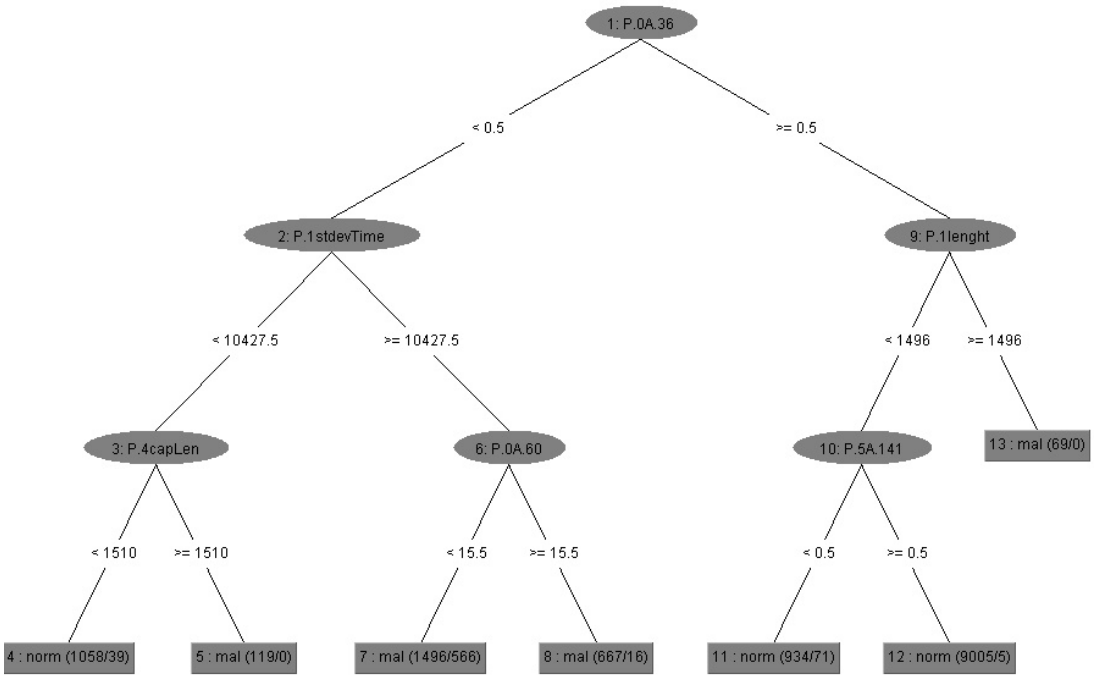

Figure 22: The resulting tree of experiment 7. The number after the P indicates which of the 8 packets, then comes the attribute

E.3 Experiment 8

Confusion matrix:

	mal	norm
classified as mal	1864	12
classified as norm	14	11430

References

[C. Saeed] Chen Saeed, Thomas M., Abu-Nimeh. (2011) Lessons from stuxnet. Computer Volume:44 Issue: 4.

[R. Williams] Ronald D. Williams, Vinay M. Igure, Sean A. Laughter. (2006) Security issues in SCADA networks. Elsevier, Computers and security.

[M. Fossi] M. Fossi, D. Turner,T. Adams, J. Blackbird, S.Entwise, B.Graveland, D.McKinney, J. Mulcahy, C. Wueest, K Haley, E Johnson. (2009) Symantec, Internet Security Threat Report 2009. Symantec.

[A. Nicholson] A. Nicholson, S. Webber, S. Dyer, T. Patel, H. Janicke. (2012) SCADA security in the light of Cyber Warfare, Computers and security.

[Infosecurity] InfoSecurity-Magazine. (2009) MI5 hires teenage hackers in fight against cyberterrorism.

[V. Jyothsna] V. Jyothsna, V. Rama Prasad, K. Munivara Prasad. (2011) A Review of Anomaly based Intrusion Detection Systems. International Journal of Computer Applications.

[O. Linda] Ondrej Linda, Todd Vollmer, Milos Manic. (2009) Neural network Based Intrusion Detection System for Critical Infrastructures, International Joint Conference on Neural networks.

[E. pleijsier] Erik Pleijsier. (2009) Towards Anomaly Detection in SCADA Networks using Connection Patterns, International Joint Conference on Neural networks.

[K. Rieck] Konrad Rieck, Philipp Trinius, Carsten Willems, Thorsten Holz. (2011) Automatic analysis of malware behavior using machine learning, Journal of computer security IOS Press.

[Kali] Offensive security. (2014) Kali Linux, Website: www.Kali.org, date of download: 1 June 2014.

[Cyber arms] Cyberarms wordpress. (2013) worldwide map of internet connected SCADA Systems.

[Wireshark] Wireshark. (2014) Wireshark. Website: www.Wireshark.org, date of download: 1 June 2014.

[Symantec] Symantec connect. (2010) KingView Heap Based Buffer Overflow Vulnerability. Website: www.securityfocus.com/bid/45727/exploit, Date of download: 1 june 2014.

[Metasploit] Rapid 7. (2014) Metasploit. Website: www.Metasploit.com, date of download: 1 June 2014.

[Army magazine] Army magazine. (2014) Building Teams of Cyber Warriors.

[J. McHugh] J. McHugh. (2000) Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory, ACM Transactions on Information and System Security.

[J. Jonathan] J. Jonathan, Davisa, K. Andrew, Clark. (2011) Data Preprocessing for Anomaly Based Network Intrusion Detection: A Review, Compuer and Security.

[KDDcup] Knowledge discovery and datamining. (1999) KDD Cup 1999: Computer network intrusion detection. Website: www.sigkdd.org/kdd-cup-1999-computer-network-intrusion-detection, date of download: 1 May 2014.

[P. Ahmed] P. Ahmed, Amrita. (2013) A Hybrid-Based Feature Selection Approach for IDS, NetCom 2013

[S. Kumar] A. Arya, S. Kumar. (2014) Information theoretic feature extraction to reduce dimensionality of Genetic Network Programming based intrusion detection model. ICIT 2014.

[U. Subramanian] U. Subramanian, Hang See Ong. (2014)Analysis of the Effect of Clustering the Training Data in Naive Bayes Classifier for Anomaly Network Intrusion Detection. Journal of Advances in Computer Networks.

[DARPA] MIT. (1999) DARPA Intrusion Detection Data Sets. Website: www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data, date of download: 1 may 2014.

[K. Rajitha] K. Rajitha, Dr.D. Vijaya Lakshmi, R. Udaya, Sreekanth. (2014) Mining Technique For DARPA 99 Intrusion Detection Data Set. International Conference on Computer and Communication Technologies 2K14.

[R. Perdisci] Roberto Perdisci. (2008) McPAD : A Multiple Classier System for Accurate Payload-based Anomaly Detection. Elsevier Science.

[LookupTables] lookup tables. Website: www.lookupTables.com, date of download: 1 may 2014.

[Rapid7] Metasploitable 2.0. Website: www.information.rapid7.com/Metasploitable-download.html, date of download: 1 june 2014.

[vulns] Vulnerabilities in Metasploitable 2.0. Website: www.hackertarget.com/sample/nexpose-Metasploitable-test.pdf, date of download: 1 may 2014.

[tcpDump] tcpdump. Website: www.tcpdump.org/manpages/tcpdump.1.html, date of download: 1 may 2014.

[Winpcap] WinPcap a pcap library for JAVA. Website: www.winpcap.org/, date of download: 15 may 2014.

[I. S. Dhillon]] I. S. Dhillon, S. Mallela, and R. Kumar. (2003) A divisive information-theoreticfeature clustering algorithm for text classification, Journal of MachineLearning Research, 3:12651287, 2003. Website: www.dl.acm.org/citation.cfm?id=944973, date of download: 15 june 2014.

[S. Warren] S. Warren, McCulloch and Walter Pitts. (1943) A logical calculus of the ideas immanent in nervous activity, Bulletin of mathematical biophysics 5.

[C. Mihaescu] Dr. Ing. Cristian Mihaescu. Flat Clustering K-means algorithm. Website: www-users.cs.umn.edu/ han/dmclass/cluster_survey_10_02_00.pdf, date of download: 1 may 2014.

[M. Eldracher] M. Eldracher. (1992) Classification of Non-Linear-Separable Real-World-Problems Using deltaRule, Perceptrons, and Topologically Distributed Encoding. Applied Computing: Technological Challenges of the 1990s.

[K. Hornik] Kur Hornik. (1989) Multilayer feedforward networks are universal approximators, Neural networks Vol. 2.

[M. W. Gardner] M. W. Gardner, S. R. Dorling. (1998) Artificial neural networks (the multilayer perceptron) - A review of applications in the atmospheric sciences, Atmospheric environment Vol.32 Elsevier.

[F. Piekniewski] F. Piekniewski and Leszek Rybicki. (2004) Visualizing and Analyzing Multidimensional Output from MLP Networks via Barycentric Projections, Departament of Mathematics and Computer Science.

[M. Tom] M. Tom. Mitchell. (1997) Machine Learning, McGraw-Hill Science/Engineering/Math.

[A. Blumer] A. Blumer. (1987) Occam's Razor, Information processing letters volume 24.

[G. Lovet] Guillaume Lovet. (2006) Dirty Money on the Wires: The Business Models of Cyber Criminals. Virus Bulletin Conference 2006.

[M. Clement] M. Clement. (2008) The Cost Impact of Spam Filters: Measuring the Effect of Information System Technologies in Organizations. University of Hamburg Institute for Marketing and Media. Website: www.d-nb.info/1006920315/34, date of download: 15 april 2014.

[C. Kanich] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage. Spamalytics: An empirical analysis of spam marketing conversion. In Proceedings of the 15th ACM Conference on Computer and Communications Security.

[M. Fossi] M. Fossi, D. Turner, T. Adams, J. Blackbird, S.Entwise, B.Graveland, D.McKinney, J. Mulcahy, C. Wueest. , K Haley, E Johnson. (2010) Symantec, Internet Security Threat Report 2009. Methodology, 2008 Symantec.

[S. Mukkamala] S. Mukkamala. (2002) Intrusion detection using neural networks and Support Vector machines, New Mexico Institute of Mining and Technology.

[L. Breiman] Leo Breiman. (2001) Random Forests, Machine learning Springer.

[M. Eeten] M. van Eeten, J.M. Bauer, S. Tabatabaie. (2009) Damages from internet security incidents. A framework and toolkit for assessing the economic costs of security breaches. TU Delft.

[B. Cashell] B. Cashell, William D. Jackson, M. Jickling, and B. Webel. (2004) The Economic Impact of Cyber-Attacks, Government and Finance Division.

[McAfee] McAfee, center for strategic and international studies. (2013) The Impact of Cybercrime and cyber espionage.

[T. Dietterich] Thomas G. Dietterich. (2000) An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, Machine Learning, springer.

[O. Maron] Oded Maron, Tomas Lozano-Perez. (1998) A Framework for Multiple-Instance Learning, AI Lab , M.I.T.
Website: www.lis.csail.mit.edu/pubs/tlp/maron98framework.pdf, date of download: 15 june 2014.

[M.v.j.] Ministerie van Veiligheid en Justitie. (2010) Nationaal Trendrapport Cybercrime en Digitale Veiligheid.

[T. Jaochims] Thorsten Jaochims, Dortmunt. (1997) Text categorization with support vector machines: learning with many relevant features, University of Dortmund.

[E. Mizutani] Eiji Mizutani and Stuart E. Dreyfus. (2001) On complexity analysis of supervised MLP-learning for algorithmic comparisons, IEEE.

[I. Weon] Ill-Young Weon, Doo-Heon Song, Sung-Bum Ko, Chang-Hoon Lee. (2005) Multiple Instance Learning Problem Approach Model to Anomaly Network Intrusion Detection, International Journal of Information Processing Systems Vol.1, No.1.

[A. Simmonds] Andrew Simmonds, Peter Sandilands, Louis van Ekert. (2004) Lecture Notes in Computer Science Volume 3285, 2004.