Smart Network-based Intrusion Detection System (SNIDS) for Advanced Metering Infrastructure

A dissertation submitted to The University of Manchester for the degree of Master of Science in the Faculty of Engineering and Physical Sciences

2017

By

Philokypros Ioulianou

School of Computer Science

Table of Contents

Word Count: 17587

List of Figures

List of Tables

List of Abbreviations

SG	Smart Grid
SNIDS	Smart Network-based Intrusion Detection System
AMI	Advanced Metering Infrastructure
IDS	Intrusion Detection System
SM	Smart Meter
CPU	Central Processing Unit
RAM	Random Access Memory
HAN	Home Area Network
NAN	Neighbourhood Area Network
WAN	Wide Area Network
KB	Kilobytes
MB	Megabytes
MIM	Man-In-The-Middle
DoS	Denial of Service
U2R	User to Root
R2L	Remote to Local
KDD99	KDD CUP 1999 dataset
SVM	Support Vector Machine
DT	Decision Tree
RBF	Radial Basis Functions
RF	Random Forest
OOB	Out-of-Bag
MAC	Medium Access Control layer
PHY	Physical layer
DR	Detection Rate
PCA	Principal Component Analysis
LFS	Linear Forward Selection
GUI	Graphical User Interface
IDE	Integrated Development Environment
API	Application Programming Interface

TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
JVM	Java Virtual Machine

Abstract

Smart meters have been deployed around the globe at a fast pace. Technologically improved meters will have many benefits for the consumers. However, these devices are vulnerable to different types of attacks. Many researchers suggest using Intrusion Detection Systems (IDSes) for detecting attacks. An Intrusion Detection System (IDS) employs a classifier to identify known and unknown attacks. Finding the proper classifier for an IDS is a challenge because no single classifier can detect all types of attacks accurately.

In this work, the problem of detecting malicious attacks in Advanced Metering Infrastructure (AMI) is studied. Focus is on the attacks affecting the availability and the integrity of the system. Smart Network-based Intrusion Detection System (SNIDS), a hybrid and distributed network-based IDS, is developed to detect attacks. In the literature, very few studies attempt to use multiple classifiers in IDS. For this reason, a combination of Support Vector Machine (SVM) and Random Forest (RF) classifiers is used in NAN-let software component of SNIDS. SNIDS was implemented in Java using Eclipse Integrated Development Environment (IDE). A Graphical User Interface (GUI) was also developed for running SNIDS.

The system developed in this project was tested to ensure that it classifies attacks correctly. Also, its detection performance was evaluated. The results showed that two combined classifiers achieve higher accuracy than single classifiers. More importantly, detection rate and speed are improved when feature selection methods are used.

Declaration

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual Property Statement

i. The author of this dissertation (including any appendices and/or schedules to this dissertation) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this dissertation, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has entered into. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the dissertation, for example graphs and tables ("Reproductions"), which may be described in this dissertation, may not be owned by the author and may be owned by third parties. Such intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialization of this dissertation, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/display.aspx?DocID=24420), in any relevant Dissertation restriction declarations deposited in the University Library, and The University Library's regulations (see http://www.library.manchester.ac.uk/about/regulations/_files/Library-regulations.pdf).

Acknowledgments

I would like to express my deepest appreciation to my supervisor, Dr Ning Zhang, for giving me endless support and valuable advice for solving any issues. Thank you for trusting me and encouraging my research as well as for helping me throughout this year.

Also, I would like to express my special thanks to my parents, brothers and friends for providing me with courage and support throughout this year and for helping me to study the MSc at the University of Manchester. Without them, this accomplishment would be impossible.

Chapter 1 – Introduction

1.1. Motivation

Nowadays, governments around the world are replacing the electricity and gas meters with new smart meters. These smart meters will help consumers to understand better their energy usage, have more accurate bills as well as save money on their bills. By 2020, the European Union (EU) will replace 80% of traditional meters with smart meters (Energy, 2014). In the UK, the number of smart meters installed in homes will be about 85% (Clark, 2016). Smart meters, sensors and special tools for supporting the management of power transmission and distribution will be the main parts of the AMI.

However, the AMI network introduces many security risks. These should be promptly addressed before attackers exploit the device's vulnerabilities creating chaos among countries. Lately, security experts "have raised concerns about the meters' security" (Clark, 2016). In some cases, smart meters were hacked (Krebs, 2012). Moreover, attacks can cause damage to smart meters and to the three network levels of AMI: Home Area Network (HAN), Neighbourhood Area Network (NAN) and Wide Area Network (WAN). For example, a distributed Denial of Service (DoS) attack can take place if multiple smart meters are compromised in a NAN. This can be done by injecting malware into the smart meter's software and then, use the compromised device to take control of other meters. Consequently, the security of smart meters must be examined again so that proper solutions are developed to protect the critical infrastructure.

Many methods such as encryption algorithms, firewalls, authorization and authentication mechanisms and even security protocols are used to protect AMI network. However, Cleveland (2008) states that encryption and authentication mechanisms alone will not be enough for AMI security protection. Therefore, monitoring solutions are vital for AMI and must be used. Regarding firewalls, even though they offer a kind of defence, they do not fully protect the network. Thus, they are recommended to be used with Intrusion Detection System (IDS). An IDS is "a

computer system (possibly a combination of software and hardware) that attempts to perform intrusion detection" (Kerschbaum, Spafford and Zamboni, 2000).

1.2. Aim and Objectives

The aim of this project is to develop a solution that will protect AMI from cyber-attacks by detecting threats. The solution is a hybrid and distributed network-based IDS. The proposed IDS will classify network traffic into "normal" and "malicious" using different machine learning algorithms. In the literature, very few studies attempt to use multiple classifiers in IDS. For this reason, in this work a combination of Support Vector Machine (SVM) and Random Forest (RF) classifiers is used. It is expected that using a combination of classifiers will produce better results than single classifiers.

Achieving the above goal required the definition of multiple objectives. The objectives of the project are as follows:

- Study the literature for existing work on IDSes protecting AMI. This includes identifying the requirements and constraints of IDSes.
- Study the literature for classifiers used in IDSes. Several classifiers will be compared and the best ones will be chosen. Accuracy and speed are the criteria for choosing the classifiers. One will be selected to protect HAN, and two classifiers will be combined to protect NAN.
- Design a new IDS with the proper software components. Software components are designed based on smart meter's constraints, such as low memory.
- Implement an IDS as a software tool. IDS will have two software components. These software components will use classifiers to detect normal and anomaly network packets in different networks. The classifiers to be used will be selected from literature based on their performance.
- Test and evaluate the IDS to make sure it works as expected.

1.3. Thesis Structure

The chapters of the thesis are the following:

Chapter 2 - The scope of this chapter is to define and explain the most important concepts of the thesis. Specifically, definitions of Smart Grid (SG), AMI and IDS are given. The security issues in SG are also explained. Moreover, algorithms that IDS uses for attack classification are discussed.

Chapter 3 - The aim of this chapter is to describe in detail the design of Smart Network-based Intrusion Detection System (SNIDS). In the first section, the system architecture is explained along with the functionalities of the HAN-let and NAN-let. These software components make use of datasets which are available to researchers for training and testing an IDS. In section 3.3, KDD99 and NSL-KDD datasets are described in detail. Feature selection techniques are also introduced for improving the performance of SNIDS.

Chapter 4 - This chapter describes the implementation of SNIDS. It starts with a discussion about the development environments and programming languages. After that, the actual implementation is described in detail. An analysis of the functionalities of SNIDS is also provided. Then, the implementation of Graphical User Interface (GUI) of SNIDS is explained.

Chapter 5 - This chapter presents the evaluation results of SNIDS. Testing methodology and performance metrics are explained in the first place. SNIDS' performance is tested by running the software components and exporting the results of metrics. Then, the evaluation of SNIDS' components is shown. In the last section, SNIDS is compared with similar IDSes from literature in terms of Accuracy, Detection Rate (DR) and False Positive (FP) rate.

Chapter 6 - This chapter summarizes the results of the research carried out in this project. Furthermore, further improvements are suggested.

Chapter 2 – Background and Literature Review

2.1. Chapter Overview

The topics to be discussed in this work requires some background knowledge. The scope of this chapter is to define and explain the most important concepts of the thesis. Firstly, the SG concept and the AMI will be described to understand the difference between the two. Then, the security issues in SG will be explained so that the reader can understand the dangers that exists in these networks. Then, the IDS along with the different classifications will be presented. Various algorithms that researchers are implementing in IDSes are also discussed.

2.2. Smart Grids

2.2.1. The Smart Grid vision

The current electric power grid throughout the world is mainly an old-fashioned system which cannot satisfy the today's needs for big amount of electricity. The world is changing. As a result, new challenges appear for electricity transmission and distribution. Low reliability of the current electric grid due to many power outages, high maintenance costs due to old machines, and lack of security measures such as firewalls are some of the most important problems of the current grid. Thus, the current system will be transformed into an intelligent system, called "Smart Grid", which will help manage electricity supply and demand efficiently. The new SG will be composed of Smart Meters (SMs), sensors, and special tools that will support the management of power transmission and distribution.

Consumers will be benefited from this new SG. Firstly, the costs of energy bills will be reduced. Consumers will make a more efficient use of energy because SMs will show their energy usage. In addition, the security level of the whole grid will be improved. IDSes will provide a real-time picture of the network status. Finally, network providers will have better grid management and will act fast to solve any customer's issues. The two-way communication between utility provider and consumer will make this possible.

Regarding the definitions of the term “Smart Grid”, there aren’t any globally accepted definitions yet. However, in the UK the most common definition is from the Energy Network Association (2014) which defines the Smart Grid as “everything from generation through to home automation with a SM being an important element, with every piece of networks equipment, communications technology and processes in between contributing to an efficient and smart grid.”

The SG network is predicted to interconnect various home appliances with the SM. This will help to establish a flexible, reliable, cost-effective and environmentally friendly power system. It will be able to manage network components in a way so that alternative energy sources are used in the electricity network (Jenkins, 2010).

2.2.2. Advance Metering Infrastructure (AMI)

The SG of the future will be based on a sensor network called Advanced Metering Infrastructure (AMI). The AMI is a system mainly composed of SMs, in-home displays, AMI communication network and the utility’s backhaul system. The most important feature is that it enables two-way communication between customers and utility provider. This makes it possible for automated data collection from meters, real-time system monitoring and other demand-response functionalities (Bennett and Highfill, 2008).

Moreover, the AMI consists of two main components; SMs and communication networks. SM is a low-cost network embedded device deployed in houses and in industrial environments responsible to metering the power usage. SMs have the following functionalities:

- Observing and tracking the demand and supply
- Recording power cut events in the internal flash memory
- Sending the usage data to the AMI provider
- Sending and collecting control messages such as remote disconnect
- Providing real-time billing services to the user thanks to their internal clock

Apart from the obvious advantages of SMs, there are some drawbacks too. Firstly, due to their limited functionalities, they have very low Central Processing Unit (CPU) power and Random Access Memory (RAM). These devices must be cost-effective and that's why they have small processing power. Secondly, SMs are in public and private places. Therefore, they are more vulnerable to physical attacks such as energy theft and device modification. Consequently, measures should be taken to protect the devices from these attacks.

The communication infrastructure of SG has a hierarchical structure in which various devices using different technologies exchange critical information. Particularly, AMI consists of three main communication networks i.e., Home Area Network (HAN), Neighbourhood Area Network (NAN), and Wide Area Network (WAN). HAN is basically the customer's smart devices that form the home network which is connected to the SM. All the house appliances are connected to the SM which records in real-time the devices' power usage. It works as an interface for HAN as it allows the communication between AMI (NAN) and HAN. Usually, all the devices in HAN, similarly in NAN, are using wireless technology for communication.

Regarding NAN, it provides bi-directional communication between the SM in HAN and the data concentrators. Data concentrators are recording the energy usage and the pricing for each household. Moreover, NAN is the most important part of the network as it collects the data from many HANs and forwards them to AMI headend (WAN) for further processing. The need for communication and the fact that many devices are distributed in a large geographical area lead to the use of wireless technologies for data transmission in NAN (Meng *et al.*, 2014).

About WAN, it has the central systems of utility provider. It is the largest one because it connects the utility control centres with NAN's data concentrators that are in large geographical areas. Although the role of WAN is important for self-healing and situation awareness control of the network, this dissertation focuses on improving the security of NAN and HAN. Figure 2.1 below shows the communication networks of AMI. It can be distinguished that the scale of each network differs. For example, HAN is limited to some square meters while NAN and WAN cover some square kilometres.

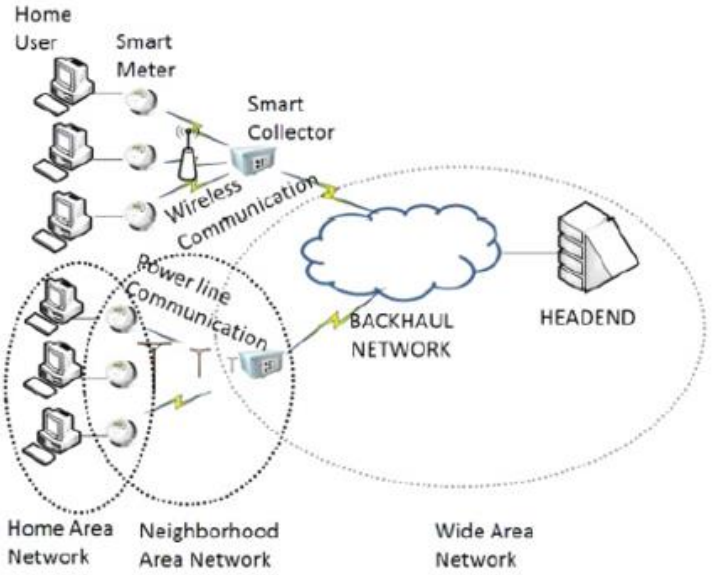


Figure 2.1: *HAN, NAN and WAN (Source: Ali and Al-Shaer,2015)*

Every network has some intelligent devices that are vital for the proper functionality of AMI. For example, SMs are in HAN, data concentrator/collector is in NAN and AMI headend is in WAN. All these three types of devices have continuous data flow but they have some differences. As can be seen from Table 2.1, the SM which is in HAN receives very small amount of data because house appliances send a few KB of information to SM. Smart devices are required to process small amount of data and should be low-cost. Thus, SMs have a few KB in memory and low processing power just enough for recording data, sending the information and receiving messages. Moreover, due to infrequent requests the transfer speed is very slow.

Regarding data collector, it is a device responsible for receiving the data from different HANs and forwarding them to AMI headend (WAN). It has a memory of some MB because it must process a large amount of data received from thousands of SMs. Transferring lot of data from SMs requires higher transfer speed. That's why data transfer speed in NAN should be higher than in HAN.

The most powerful device among the three networks is the AMI headend. This device is in WAN. The WAN covers a large geographical area. As it is expected, the amount of data received from data collectors is huge. Consequently, the AMI headend

must have several servers with large memory for fast processing of the information. Speed should also be high enough for handling the data as fast as possible.

Smart meter (HAN)	**Data concentrator/collector (NAN)**	**AMI headend (WAN)**
Very small amount of data in HAN from house appliances.	Big amount of data collected from ten to thousands of SMs.	Huge amount of data received from millions of SMs.
Very limited resources (CPU, memory in KB).	Bigger need in resource (memory in MB).	Resources must be very high like in a server.
Very slow data transfer from SM due to infrequent requests.	Data transfer speed is high because aggregates data from SMs.	Data transfer speed must be high to handle huge amount of data.

Table 2.1: *Differences of AMI devices*

2.2.3. Security issues in Smart Grid

SG is based on AMI networks which use technologies that can be targets for malicious activities. Currently, vulnerabilities such as malicious software injection in a SM (Goodspeed *et al.*, 2009) or the intentional disconnection of the devices for creating a blackout (Bennett and Highfill, 2008) have already been discovered. Another class of threats in the SG is the wireless technology that is used by the devices for communication forming a mesh topology. Although this technology is cheap and convenient to use, attackers can damage the nodes or the whole network. Some attacks that may happen include DoS attacks such as signal jamming and resource exhaustion, harming the routing protocols, performing unauthorised network functionalities, replay attacks, and spoofing of SMs. Apart from attacking the nodes or the network, hackers can exploit the communication link that nodes use to share information. For instance, data packets can be captured by sniffers allowing attackers to perform Man-In-The-Middle (MIM) attacks or get unauthorised access to important information. Last but not least, the integrity of the information stored in SMs is another target of attackers. As Bennett and Highfill (2008) support, the smart devices must report the correct readings to the utility when needed. Still, this is may not always be possible as consumers have physical access to the devices and may attempt to change the measurements.

The attacks above can be distributed among the three network levels of an AMI network. For example, attacks in a HAN target SMs and they are usually hardware or software modifications. These include replacing the device with a cloned one to manipulate billing information. Another attack is to reprogram the device's firmware to control the information exchanged with the control centre. In addition to these attacks, DoS attack is possible by connecting too many smart devices of the house in one SM. Signal jamming can also be used to prevent the meter's communication.

Malicious attacks can happen in NAN as well. An attacker can sniff NAN's traffic aiming at sending tampered messages to the headend or capturing a neighbourhood's SM data for future attacks. The ultimate attack is to compromise the data concentrator for wide-range attacks. An example of this type of attack is the spread of a malicious software to SMs in a mesh topology causing wide-scale problems.

About WAN, similar attacks may be executed. Yet, because of the utility provider's systems, the attacks usually target at stealing metering data so that they can create user's profiles which can be sold later. As can be seen, the attacks are distributed in each network level and should be prevented using the proper measures.

Cleveland (2008) discusses the security requirements of an AMI network. The main concerns are the confidentiality, integrity, availability, and non-repudiation. Although all the requirements are important for AMI, this project will focus on ensuring the availability of the network. Keeping the systems and AMI network online for every single day is a challenge. The reason for that is the intentional or unintentional failures that devices may suffer. Some examples of failures are problems with the software or the hardware of the device, communication problems such as network traffic, and even bandwidth issues can make the devices unavailable. From customer's perspective, availability means that electricity will always be available and nobody can shut down the power intentionally or accidentally. However, from the provider's view availability means having access to critical information at any time. For example, devices that handle important data report values to utility centre in small time periods and should be online every single minute. Energy provider need access to the device's data to block any attempt of attack and to keep the network online.

Solutions exist for preventing many of the above-mentioned attacks but more research is needed in this topic. Taking into consideration the four security requirements, researchers have divided security solutions into three aspects: encryption and authentication algorithms, privacy protection, and intrusion detection (Meng *et al.*, 2014). For securing SG, encryption and authentication algorithms are needed for preventing data theft while data is transmitted from node to node. Even though cryptographic algorithms exist for protecting data in computer networks, the requirements in smart grids are different. Therefore, new algorithms should be designed. Encryption algorithms will be used for encrypting data so that someone will need a key to decrypt them. Similarly, authentication algorithm will be used to authenticate and authorize SMs to the utility centre. The limited capabilities of the smart devices such as low memory and low CPU speed, should be considered for designing these algorithms with low complexity.

Another aspect of securing SG is the protection of customer's data. Smart meters in AMI network will collect detailed information about the electricity usage in specific time periods. The data collected are sensitive information and must be protected by anonymizing them. If an attacker has access to raw data, anything could happen. For that reason, privacy protection should be considered for securing smart grid.

Finally, in AMI networks it is important to have an IDS. This system will enable the detection of malicious traffic in the network and it will warn the administrator. In the current work, an IDS system will be implemented for detecting malicious attacks on SMs and the AMI network. The IDS can be centralized or decentralized but the second type is usually preferred like in *da Silva et al.* (2005). Beigi-Mohammadi *et al.* (2014) claim that the centralized approach does not work due to scalability issues as well as computational problems that central server may face. In a decentralized IDS, the already deployed devices help in processing the big amount of data. No single point of failure exists. Additionally, IDS will detect attacks at each network level which makes the detection procedure faster and more effective.

2.3. Intrusion Detection System (IDS) and its Classifications

2.3.1. Introduction to IDS

The main challenge in securing AMI from malicious activities is to develop a monitoring system that will fulfil the needs and the limitations of the AMI network. The most common monitoring system that is used in networks is called “Intrusion Detection System (IDS)”. An IDS is “A computer system (possibly a combination of software and hardware) that attempts to perform intrusion detection” (Kerschbaum, Spafford and Zamboni, 2000). In other words, it is a system that detects if there is an ongoing attack or not and triggers an alert to the user when an intruder attempts to launch an attack.

The first line of defence are methods such as encryption algorithms, firewalls, authorization and authentication mechanisms as well as security protocols. However, Cleveland (2008) states that encryption and authentication mechanisms alone will not be enough for AMI security protections. Using monitoring solutions are vital for protecting AMI. Regarding firewalls, even though they offer a kind of defence, they do not fully protect the network. That’s why IDS is recommended to be used with firewalls. IDS is vital in a network in case security mechanisms such as encryption and authentication are breached (Koshal and Bag, 2012).

An IDS is commonly composed of: 1) sensors to track network activities, 2) a centralized management server to process the data received from sensors, 3) a database to save the information generated from IDS, 4) an interface for administrators to check the status of the system, receive warnings and make the proper configurations to the system.

Regarding the types of an IDS, Cleveland (2008) states that two types exist: host-based IDS and network-based IDS. The former is used for analysing and checking the file integrity of a system, the audit logs and the system calls, while the latter is used for analysing network traffic and data packets. The advantage of host-based IDS is that it provides a better view of the behaviour of each program that runs on the host. Regarding network-based IDS, only one IDS is needed for all the hosts on a network. Still, the latter needs to process lots of network traffic. The drawbacks of the host-

based IDS are that for every host an IDS is needed, and if an attacker gains access to a machine, the IDS can be tampered with (Kosamkar and Chaudhari, 2013). Apart from these types, according to the detection mechanisms IDSes can be classified into three types: signature-based, anomaly-based and specification-based (Berthier *et al.*, 2010).

2.3.2. Signature-based detection

Signature-based or misuse detection is based on the existence of specific patterns for intrusions, enabling the system to report any activity that matches with any pre-existing pattern. Patterns are basically known attacks and are also called signatures, the root of the term signature-based detection. These systems can detect well-known attacks very accurately and this is the reason for being installed widely in industry. However, attackers invent new complex attacks and this detection method will fail to report unknown attacks as well as variations of them. According to Wu and Banzhaf (2010), detecting unknown attacks requires creating a local database and updating it regularly. This database can be updated manually, which takes a lot of time, or automatically by using intelligent algorithms. An example of signature-based IDS is the rule-based approach where set of rules are created to identify attacks and if an activity matches with the rules, the action specified for this rule is executed.

2.3.3. Anomaly-based detection

Anomaly-based detection tries to recognise malicious behaviour. It needs the previous creation of profiles for defining the normal behaviour of users, hosts or networks. Therefore, the data is collected and stored in the database during a normal operation. The anomaly IDS uses various statistical measurements to separate abnormal behaviour from normal. During the detection phase, false alarms are more likely to happen as user's behaviour might not be consistent and the system will find it difficult to detect the attack. Thus, keeping the profiles updated is important. However, these systems can detect unknown attacks in contrast with signature-based systems. For this reason, they are mostly preferred by researchers (Arumugam *et al.*, 2010; Haddadi *et al.*, 2010).

2.3.4. Specification-based detection

This model is similar to anomaly-based detection approach in detecting intrusions. In specification-based IDS, the normal behaviour is defined by taking into account the functionalities and the security policies of the system. A profile with normal behaviour is created. Any kind of operation that happens outside the specifications is considered suspicious and is a possible intrusion. The benefits of the system are the detection of unknown attacks accurately as well as it is cost-efficient to create it. However, a new specification IDS had to be designed for each protocol. This is because it is difficult to generalize for too many protocols. Also, it is hard to ensure that all the specifications of the system are correct and they protect the system from the specified threat.

2.4. Algorithms for the Intrusion Detection System (IDS)

An IDS is a combination of software and hardware which means that an intelligent algorithm is needed for detecting intrusions. In the follow subsection, the recent studies about the various approaches are presented.

2.4.1. Overview of Current Algorithms

Several algorithms have been used recently by researchers for the intrusion detection task. Nguyen and Choi (2008) survey the latest studies regarding the best algorithms for classification. They focus on four main attacking classes: DoS, PROBE (information gathering), U2R (User to Root), and R2L (Remote to Local) using the KDD99 dataset. The authors select ten classifier algorithms that belong to Bayesian, decision trees, rule-based models, function and lazy functions categories. They conclude that JRIP detects best DoS and Probe, decision table is good for U2R detection and OneR should be used for R2L. Because the first model is not fast enough for using it in real-time detection, they propose a second model that employs C4.5 (J48 in WEKA) to detect DoS attacks in a real-time environment.

Using the same dataset, Wu and Yen (2009) use machine learning and data mining methods to improve the efficiency of IDSes. Specifically, they compare C4.5 and SVM algorithms by classifying the 4 attacks types that are contained in the dataset. The

results indicate that C4.5 has better accuracy and detection rate in DoS attacks than SVM algorithm. Similarly, So-In *et al.* (2014) study several classification schemes such as decision tree, Neural Networks, Ripper Rule, Naives Bayes, k-Nearest-Neighbour and SVM using both KDD99 dataset and HTTP BOTNET attacks. They evaluate the algorithms using k-fold cross-validation and other metrics. As Vanschoren *et al.* (2014) explain, K-fold cross-validation is used for evaluating models that make predictions. It takes the original dataset and divides it into k random subsets of the same size. Then, one subset is randomly selected as testing set to evaluate the model and the rest k-1 subsets are used for training the model. This process is repeated k times (folds). Each k subset is used only once as test data. At the end, the results from the k folds are combined to generate the average error. Results using this method showed that for the case of classifying normal and attack class, the C4.5 (or J48) has the highest accuracy.

Mehmood and Rais (2016) in a recent study of machine learning algorithms SVM, Naives Bayes, J48 and decision table, they indicate that J48 has the highest accuracy among the other algorithms. This is due to the redundant features that exist in the KDD99 dataset. Thus, they recommend using a combination of algorithms for detecting attacks to improve the overall performance. Kalyani and Lakshmi (2012) studying the latest dataset NSL-KDD with the Naïve Bayes, C4.5, OneR, PART and RBF network algorithms, conclude that C4.5 and PART are those with the best performance. In another study, Chauhan *et al.* (2013) use 10-fold cross-validation for evaluation with the NSL-KDD dataset and they show that using RF is an algorithm with very high accuracy.

In a very interesting study, Choudhury and Bhowal (2015) compared the performance of nine classifiers in WEKA using both NSL-KDD dataset and 10-fold cross-validation. The results show that Random Forest and BayesNet are the best algorithms for the IDS. They also compare ensembles which improve the efficiency of the single classifiers and they conclude that the best ensemble is Boosting. In a similar study of classifications algorithms, Giray and A.G. Polat (2013) used several datasets to conclude that the best classifier is decision trees.

As can be seen, several studies attempt to solve the problem of selecting the most suitable algorithm for using in IDS for AMI. Still, most papers conclude that not a single algorithm can detect all the attacks with high accuracy. They recommend that a combination of different algorithms should be employed so that the overall system performance is enhanced. In this work, C4.5 (J48), SVM and Random Forest classifiers will be studied. These algorithms are chosen from the literature above. They have the best performance in several studies regarding intrusion detection.

2.4.2. C4.5

A Decision Tree (DT) is one of the most used classification algorithms in data mining. Its operation is based on the divide and conquer idea where the training dataset is recursively partitioned according to its attributes and it terminates when the stopping conditions are fulfilled. A DT has nodes, edges, and leaves. Each node has its own dataset and this defines the best attribute to divide the dataset into its categories. Moreover, a node has multiple edges that state the values or a range of values of the chosen attributes on the node. Based on the values of the edges, the dataset of each node is partitioned into several subsets. Then, a child node is made for each data subset and the dividing procedure is repeated. This continues in the node until the stopping conditions are met which means all the datasets are the same or the attributes cannot be further divided. At this point, the DT algorithm stops the process and the node gets the name of the class label of the dataset. The labelled node is now called a leaf. Following this recursive procedure, the result of the DT is the creation of a tree structure.

The first DT model was created by Kohavi and Quinlan (1999) and the most recent implementation of his model is C4.5. The most important key issue of the algorithm is to select the most appropriate attribute that best partitions the dataset into corresponding classes. The C4.5 model uses the information entropy theory to create decision trees from training datasets. That means, the attribute with the highest gain is selected for the partition procedure. The formula used to calculate the gain is the following:

Gain (D, T) = $Entropy\ (D) - \sum_{j=1}^{m} f_D(T_j)\ x\ Entropy(D_{T_j})$,

where Gain (D,T) is the gain of dataset D after splitting attribute T; Entropy (D) is the information entropy of dataset D; m is the number of different values of attribute T in D; T is the proportion of items possessing T_j as the value for T in D; T_j is the jth possible value of T; and D_{Tj} is a subset of D containing all items where the value of T is Tj.

Here, the entropy is obtained as follows:

Entropy (S) =$\sum_{i=1}^{n} f_D(i) x \log_2 f_D(i)$,

where n is the number of different values of the attribute in D and $f_D(i)$ is the proportion of the value "i" in the set D. After creating the tree, the algorithm calculates classification error for each node and prunes the tree accordingly.

The detailed steps of the algorithms are as follow (Kohavi and Quinlan, 1999):

Input: Training dataset D.

Output: A decision tree.

1) Create the root node R.
2) If D belongs to the same category K, then return R as a leaf node, and label it as a class K.
3) If no attributes exist or the rest data of D is below a threshold, then return R as a leaf node, and label it as the most frequent category.
4) For each candidate attribute, calculate its information gain.
5) If test attribute is the testing attribute of R, then test attribute is the attribute with the highest information gain.
6) If the test attribute is continuous, then calculate its threshold for division.
7) For each new leaf node created by node R: Calculate the classification error rate of each node, and then prune the tree.

2.4.3. Support Vector Machine (SVM)

SVM is one of the best classification algorithms according to Yang and Li (2006). It is a reliable algorithm which can achieve high accuracy on predicting the class of unseen data. It was first introduced by Vapnik (2000) who developed the principle of structural risk minimization which SVM is based on. This principle aims in finding a hypothesis h for which one can be sure that the lowest error is observed while other methods are using the empirical risk principle which tries to improve the performance of the training set. The learning process of SVM starts with mapping the values of the training data into high dimensional feature space with the use of kernels. After this step, it computes a hyper plane that separates the data points with a maximum margin. The kernel transforms a problem that is linearly non-separable to a separable one. SVM can be used with many kernel functions such as polynomial, RBF, linear and sigmoid. The user can select a kernel function to use in the training phase and SVM chooses support vectors along with this function.

Suppose an input of M data points {(x1,y1), (x2,y2), (x3,y3), … ,(xM, yM)}, where xi∈Rd and y∈{+1,-1}. A hyperplane can be defined by (w, b) where “w” is a weight factor and “b” is bias. The decision function that is used when a new object “z” is classified is the following:

$$\mathrm{f(z)} = sgn\ (\mathrm{w} * \mathrm{z} + \mathrm{b}) = \sum_{j}^{M} a_j y_j \left(x_j * z\right) + b$$

When the f(z) has a positive result, the object “z” is accepted as normal else the object “z” is treated as outlier when f(z) is negative.

In this project, SVM is used in the IDS because it has many benefits. First of all, SVM is a fast algorithm for detecting intrusions and that is significant for building a real-time IDS which can take decisions quickly. Scalability is another advantage of SVM. That means that the complexity of the algorithm is not dependent on the size of the feature space. Consequently, it can learn a large dataset and scale better than neural networks. Moreover, it is less prone to overfitting because the number of

parameters used depend on the margin that separates the data points and not on the number of features (Sung, 1998).

2.4.4. Random Forest

Random Forest (RF) is an ensemble method of un-pruned classification or regression trees (Breiman, 2001). Ensemble is a divide-and-conquer methodology applied to enhance the performance. The idea behind the ensemble is that several "weak learners" can gather together and form a "strong learner". Therefore, the algorithm achieves the highest accuracy among the other data mining algorithms, mainly with large datasets. RF algorithm creates several classification trees using a tree classification algorithm which constructs each tree by using a separate bootstrap sample from the training dataset. Classifying a new object is done when the forest is formed. The new object is put below of each tree. Then, each tree votes for the class of the object so that the forest will select later the class with the most votes. The algorithm below is for both regression and classification (Liaw and Wiener, 2002; Svetnik *et al.*, 2003):

1) Select m_{tree} samples with replacement (bootstrap) from the training of m samples.
2) For each selected sample, grow a regression or classification tree with the following alterations: At each node, randomly select n_p predictor variables from all the predictors and choose the predictor with the best split from those variables. The tree will grow to the maximum size and will not prune back.
3) Classification of new data is based on the majority vote of the m_{tree} trees. The average from these trees is used for regression.

In order to evaluate the test error rate, RF algorithm calculates the Out-of-Bag (OOB) error during the training phase. Thus, there is no need for cross-validation or a test set to get an unbiased test error. In other words, the fact that each tree is created based on bootstrap sample means that the one-third of cases, called OOB cases, are out of the bootstrap samples and are not used in training. (Zhang and Zulkernine, 2006)

The main parameters used for configuring the RF are: the number of trees (m_{tree}), the number of predictors randomly selected as candidates for splitting at each node (n_p) and the minimum node size. During the development of the forest, attributes are chosen arbitrary from all the attributes of the training dataset. The most important variable to configure is the number of variables/predictors to be used in dividing the nodes in each tree (n_p). Setting this variable to a proper large number it will boost the performance of the algorithm. Apart from this parameter, the minimum node size is another criterion which the algorithm uses to decide if it will split the node or not. If the remaining nodes are below this minimum number, there will be no splitting. As a result, this parameter affects the size of the grown trees. Thus, for classification the default value of minimum node size is 1, making sure trees have reached the maximum size (Svetnik *et al.*, 2003).

2.4.5. Hybrid approach - Ensembles

A hybrid approach or ensembles is the combination of various learning or decision-making models to boost the performance of the IDS. This approach exploits the different characteristics of each model. Consequently, accuracy and the whole generalization of the IDS are increased (Peddabachigari *et al.*, 2007).

One simple and famous hybrid method that Govindarajan and Chandrasekaran (2011) describe is Voting, which combines the results of various models. In this model, the results from the various models are combined by calculating the prediction possibilities of each method so that final predictions are decided from these probabilities. Another method is Bagging (bootstrap aggregation) and boosting (Pan and Tang, 2014) which creates several samples from the training data by bootstrapping (randomly selecting samples with replacement) and a classifier is built for each sample. After this step, the classifiers generate results which are combined by calculating the average or by using majority voting. This method uses C4.5 (J48) algorithm as a base classifier which helps avoiding over fitting and enhancing accuracy.

A third ensemble algorithm which is described by Choudhury and Bhowal (2015, May) is AdaBoost which means adaptive boosting. This model has a base classifier

which is created from the training data. It has also a second classifier which runs behind the first one so that it focuses on the instances that were wrongly observed from the first classifier. This procedure of adding extra classifiers continues until a specific limit in number of models or accuracy is reached. AdaBoost uses C4.5 (J48) algorithm as a base classifier, and it helps in improving the accuracy of any single algorithm.

2.5. IDS in AMI

Given the structure of the AMI, the main characteristics that an IDS should have are the following: 1) be a powerful system which means to have high accuracy in identifying known and unknown attacks, 2) run without causing any problems to the current system activities, 3) have the minimum overhead on the SG infrastructure, 4) prevent attacks at each network level so that the availability of the network remains unaffected. Bearing these in mind, many researchers have proposed several approaches in designing an IDS for AMI. The first approach is to use the existing IDS that have been employed in other kinds of networks which are usually centralized. Still, that traditional approach does not take into account the constraints mentioned before. Attacks within the AMI network, such as malicious attacks against the routing protocol of the mesh network, Medium Access Control (MAC) or Physical (PHY) layer attacks, and application layer attacks between peer to peer AMI nodes will remain undetected if a centralized IDS is used. A new decentralized IDS approach is the most suitable solution for AMI because the data processing and reporting is distributed among the nodes and no single point of failure exists.

Many researchers have attempted to address the issue of intrusion detection in the SG. Jokar, Nicanfar and Leung (2011) designed a specification-based IDS for HAN. Specifically, their IDS aims to protect the PHY and MAC layers of Zigbee devices which are in the HAN. Zhang *et al.* (2011) proposed the Smart Grid Distributed Intrusion Detection System (SGDIDS) which protects the SG from DoS attacks. Their system makes use of machine learning algorithms that are executed in SMs, data collectors and control centre so that any suspicious behaviour is detected. Similarly, Beigi-

Mohammadi *et al.* (2014) presented a distributed IDS that protects NAN from Wormhole attack. They employed an analytical approach to develop the IDS which is embedded in SMs and they evaluate IDS with OPNET software. The latest work from Sedjelmaci and Senouci (2016) is the development of a lightweight IDS for SG which follows both centralized and decentralized approach. Their IDs is a hybrid one as it combines rule-based and anomaly-based detection by using a machine learning algorithm. The results from the simulations show that their system is secured enough and more efficient than other works.

2.6. Chapter Summary

The current chapter explained the basics of SG and introduced some important terms that are going to be used in this project. Understanding the concept of AMI and its security problems is the main goal of this chapter. Gaining the vital information about IDSes will help the reader understand the next chapters.

Chapter 3 – Smart Network-based Intrusion Detection System (SNIDS) Design

3.1. Chapter Overview

The aim of this chapter is to describe in detail the design of Smart Network-based Intrusion Detection System (SNIDS). In the first section, the system architecture is explained along with the functionalities of the HAN-let and NAN-let. These software components make use of datasets which are available to researchers for training and testing an IDS. In section 3.3, KDD99 and NSL-KDD datasets are described in detail. Feature selection techniques are also introduced for improving the performance of SNIDS.

3.2. Smart Network-based Intrusion Detection System (SNIDS) Design

3.2.1. Assumptions

SNIDS is an IDS aiming at protecting AMI network by detecting and stopping malicious packets from entering the network. This will be achieved by running J48 classifier on HAN-let, and a combination of SVM and RF classifiers on NAN-let. Both HAN-let and NAN-let are software components. They can be embedded in smart meters or routers. Specifically, HAN-let will run J48 classifier on a smart meter in a HAN. J48 is a fast and light DT classifier, suitable for being integrated into smart sensors such as Zigbee. Regarding NAN-let, it will run Vote classifier which combines SVM and RF classifiers. Combining the results from all the classifiers improves the overall Detection Rate (DR). However, SVM and RF classifiers need more CPU and memory than J48. Therefore, NAN-let will be integrated into routers which have more processing power.

Many factors play important role in the decision of how the system should be designed. Some of them are:

- Limited computational resources in smart meters. A computational intensive classifier cannot be embedded into a smart meter because it needs large

processing power and memory. Smart meters have low CPU power and only a few KBs of RAM.

- Different types of networks in AMI. Networks such as HAN, NAN and WAN provide a hierarchical structure in AMI and each one connects different parts of the whole AMI network. As a result, an IDS should protect all three different network levels.
- Many attack types exist in AMI. Attack types vary from data breach of a smart meter to physical modification of a device. In this work, IDS should focus on DoS, Probing, U2R, and R2L attacks so that the DR for these attacks will be higher than other types of attacks.
- Ability to classify network packets correctly and in real-time. Large amount of network packets are exchanged in the AMI because smart meters in each network layer communicate between them and report to the control centre (AMI headend). Thus, a solution should consider the generated traffic from the devices so that malicious messages are stopped.

Bearing in mind the above factors, assumptions were made to simplify the design and functionality of the IDS:

- HAN-let and NAN-let should be able to run in existing devices. HAN-let could be integrated into normal smart sensors. Thus, HAN-let should have low memory requirements. Similarly, NAN-let runs heavier algorithms. Therefore, it should be executed in NAN routers which have more computational power than sensors. Both HAN-let and NAN-let software components will be developed in this work.
- HAN-let will be employed at HAN while NAN-let at NAN. For this reason, the communication between them is assumed to be encrypted and each component will have to authenticate itself using public key cryptography. In this way, security events and network health will be monitored.
- Use of widely used dataset such as NSL-KDD for training and testing. This dataset contains the four main attack types which SNIDS is assumed to detect.

- Supervised learning algorithms will be used to train SNIDS to classify packets. HAN-let is assumed to run different algorithm from NAN-let. This will protect every node in the network and increase the DR.
- Devices such as data collectors in NAN, and servers in WAN are assumed to be always available. Availability in all three network levels is important. In case of attack, the message must arrive at the control centre to notify administrator.

With the above assumptions, the architecture of the IDS will be simpler and the whole system will be easily deployed in AMI network.

3.2.2. System Architecture

3.2.2.1. Design Requirements for SNIDS

Having the proper system architecture is necessary for fulfilling the requirements of an AMI. Additionally, it will improve the overall efficiency of the IDS. In this project, HAN and NAN are the two networks to be protected by the IDS. This is because the former has the smart meter which can be targeted for stealing the energy usage data. According to Meng, Ma and Chen (2014) NAN can be attractive due to the large amount of data held by data concentrators. Regarding WAN, the designed IDS can be extended to cover this network but this is not in the scope of this work.

SNIDS should have several features that will improve the attack detection speed and will enable better decision making. First of all, a decentralized IDS should be used to distribute the processing of data in HAN-let and NAN-let. This means that decisions will be taken at each network level by the software components. The devices running these algorithms will classify the traffic and alert the network administrator in case of an attack.

In addition, a distributed approach is the most appropriate option for the proposed IDS. Millions of smart meters are already deployed in each house. Smart meters will be embedded with the HAN-let software without extra overhead. Additionally, routers in NANs will run the NAN-let component. In this way, the whole network will be monitored by existing devices.

3.2.2.2. SNIDS Network Topology

Network topology is presented in Figure 3.1. Beginning from the left side, each household forms a HAN which contains a smart meter running the HAN-let software for detecting known attacks. In the middle, NANs are shown with data concentrators located inside them. NAN-let software will be executed by central NAN routers which have the capability of running machine learning classifiers. Routers or NAN devices will be connected to WAN where the AMI headend and the backhaul network are located. The AMI headend contains the management server that stores packets information in the database for training the algorithms. There is a WAN-let in WAN which has the same capabilities as the NAN-let but it is not implemented in this work.

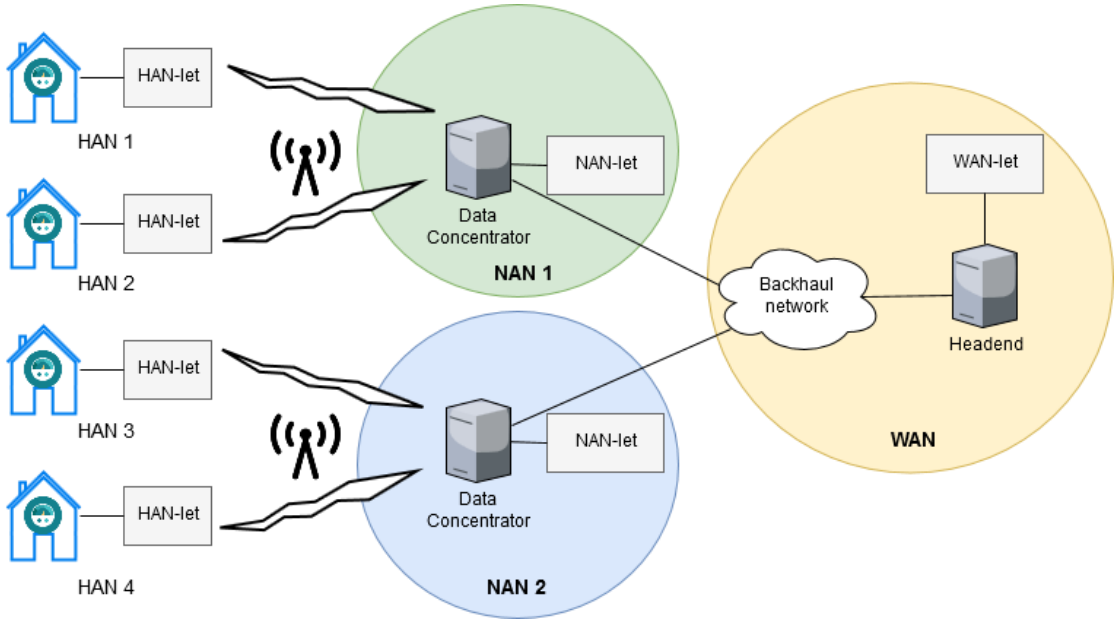


Figure 3.1: *SNIDS network topology*

As can be seen, HAN-lets (embedded in smart meters) and NAN-lets (embedded in routers) are distributed among the different networks of AMI. Monitoring all devices (smart meters, AMI headend) is strongly recommended when an IDS is designed as it makes the system less vulnerable to attacks (Tong *et al.*, 2016). Regarding data transfer, messages between HAN-lets and NAN-lets will be delivered with the use of wireless technologies like WiFi or LTE. The components will also have an encrypted communication channel where only authenticated devices will be allowed to exchange critical network data. In this way, when an attack is launched, an encrypted notification will be sent to WAN so that network administrator is notified.

3.2.3. System Components

3.2.3.1. HAN-let

One of the most important software components of the system is HAN-let. This software component can be installed in a normal sensor node. For this reason, it is developed to work with low computational power devices. HAN-let is deployed in SNIDS to protect smart meters and the whole HAN from external attacks. The algorithm to be used in HAN-let is the decision tree algorithm C4.5 which generates a tree that helps to classify a packet to normal or malicious. In this way, every packet will be classified according to its characteristics and in case a malicious one is found, the sensor will block the packet and notify NAN-let for further action. A decision tree is the most suitable classifier to be used in a HAN because it is fast and computationally inexpensive.

3.2.3.2. NAN-let

Another software component that plays an important role in SNIDS is NAN-let. This component is like HAN-let but it uses different classifiers. NAN-lets will be deployed in NANs to protect them from attacks targeting the data collector which holds vital information. Additionally, it will help HAN-lets by providing new updated data on attacks which will be generated from the prediction capabilities of NAN-let. This will be achieved by running the SVM and RF algorithms. New predicted attacks will be added in the central database in WAN. Both HAN-lets and NAN-lets will have access to the database in WAN. The former can create rules to stop attacks while the latter will predict new attacks based on these data. Due to heavy processing, NAN-let can be embedded only in devices with large CPU and memory. In case of an attack in NAN, the nodes will stop the attack and notify the AMI headend which will notify the system administrator. The same pattern is followed when an attack happens in HAN which means HAN-lets will stop the attack and they will notify both NAN-lets and the control centre. In case of a communication failure, alternative paths will be used to send the data to the specified device.

3.3. Dataset and Pre-processing Methodology

3.3.1. KDD99 and NSL-KDD Datasets

Selecting the proper dataset to evaluate an IDS is an important task. Among several available datasets, KDD99 is the one that has been used widely by researchers for benchmarking IDS. The dataset was prepared by Stolfo *et al.* (2000) and in the training set there are almost 4.900.000 records with 41 attributes (excluding 'class' attribute). Each record is categorized as normal or attack. The attack records can belong to one of the four following attack types:

1) *DoS attack* in which an attacker attempts to reserve the computational and memory resources of the system so that it becomes busy and doesn't respond to users' requests.
2) *U2R attack* in which the attacker gains access to a normal user account and then, tries to exploit the system so that it becomes super user.
3) *R2L attack* in which the hacker sends packets to a machine without having an account on that machine. The goal is to exploit the machine and become a user of the machine.
4) *Probing Attack* is when the attacker scans the machine or the network to find vulnerabilities. The goal is to compromise the machine by exploiting these vulnerabilities.

The testing data is generated differently from the training data, and it contains attack types that do not exist in the training data. Thus, the attack detection procedure is close to reality. Table 3.1 shows the number of records contained in each of the four attack categories. For more details see Appendix B.

Normal (number of records)	**Probing (number of records)**	**DoS (number of records)**	**R2L (number of records)**	**U2R (number of records)**
Normal (97.277)	Nmap (231)	Land (21)	Spy (2)	Buffer_overflow (30)
	Portsweep (1.040)	Pod (264)	Phf (4)	Rootkit (10)
	Ipsweep (1.247)	Teardrop (979)	Multihop (7)	Loadmodule (9)
	Satan (1.589)	Back (2.203)	ftp_write (8)	Perl (3)

		Neptune (107.201)	Imap (12)	
		Smurf (280.790)	Warezmaster (20)	
			Guess_passwd (53)	
			Warezclient (1.020)	

Table 3.1: *Types of attacks in KDD99 and NSL-KDD. See Appendix B for more details.*

Although KDD99 dataset has been used widely, a new dataset NSL-KDD (NSL-KDD Dataset, 2015) has appeared to replace it. The KDD99 dataset has some problems which have led to the creation of the new improved version. The two most serious problems are: 1) it contains many redundant records in both training and testing sets. This makes the classifier biased to the records that appear frequently, and 2) the number of existing records is very large and thus, researchers usually use only the 10% of the set (Meng, 2011).

The above issues are solved by NSL-KDD which contains a reasonable number of records and it doesn't have duplicate records. As a result, researchers can run experiments on the complete dataset, and the classifier will not be biased to specific records. The original NSL-KDD dataset is divided into training and testing sets with 125.973 and 25.544 records, respectively. There is also a training set that contains 20% of the original training set called KDDTrain+_20Percent (Tavallaee *et al.*, 2009). The types of attacks included in NSL-KDD are the same as the original KDD99 dataset.

In this work, the NSL-KDD is used to evaluate SNIDS using the 20% of the original set (KDDTrain+_20Percent.arff) for training. For testing, the KDDTest+.arff file is chosen which contains some attack types that are not included in the training set.

3.3.2. Feature Selection

Achieving the ideal IDS solution is a major challenge that has not been accomplished yet. An ideal IDS would have 100% DR, low FP and zero training time. However, due to large datasets with a big number of records and features, the training time remains high. As a result, measures have been taken to reduce required

computational time. One of the most common methods for improving the training time and DR of SNIDS is the feature selection method. This technique is based on selecting the best subset of attributes, called features, which are relevant and play an important role in the decision-making process. Although the features are reduced, the goal is to keep accuracy in acceptable levels.

There are three categories of feature selection algorithms: filter, wrapper and embedded methods. Starting with filter methods (Hall, 1999), they try to assess and select the features from the original dataset without using any learning algorithm. Evaluation criteria such as information gain, inconsistency, Principal Component Analysis (PCA) and others, are used for ranking and selecting the best features (Hall and Smith, 1999). The fact that this method is fast, has low complexity, it avoids overfitting and can scale to high dimensional datasets are some of the reasons of using it to reduce the number of features of a dataset (Lan and Vucetic, 2009).

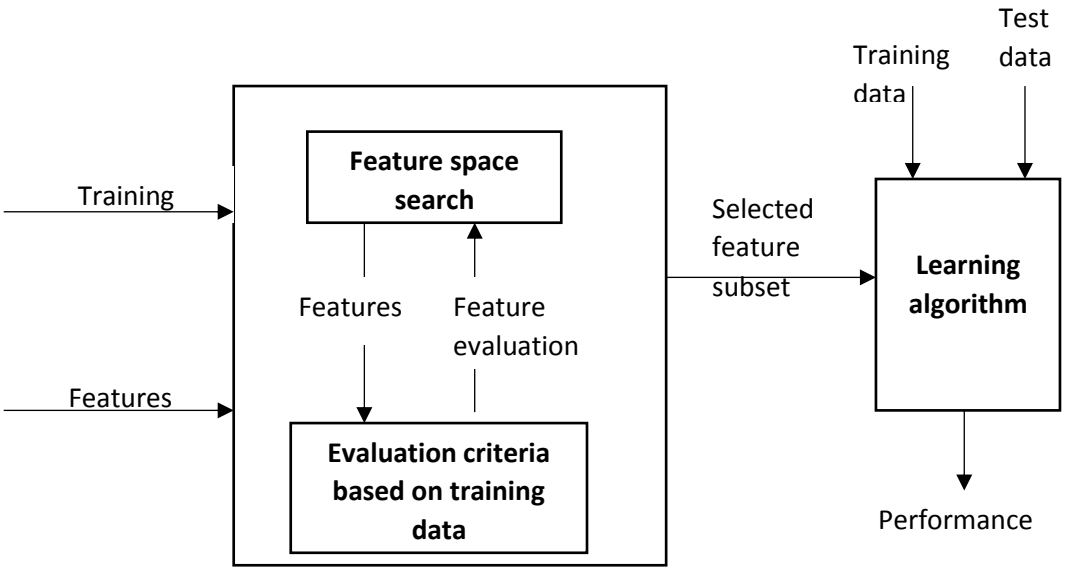


Figure 3.2*: *Filter feature selection procedure

Another category is the wrapper methods (John, Kohavi and Pfleger, 1994) which use a learning algorithm in order to search all feature subsets and find the best one. These methods select the feature subset that is most useful according to the algorithm used. Thus, the performance of the specified algorithm is considered when choosing the attributes. Any algorithm from DTs to SVM can be used for wrapper methods (Hall and Smith, 1999). Comparing wrapper with filter methods, the former perform better

in selecting attributes than the latter. This is because of taking into account dependencies between features. However, the high risk of overfitting and the long processing time are some of the drawbacks of wrappers.

Regarding embedded methods, the attribute selection process is done during the training phase. In other words, these methods try to find the best subset of attributes while they build the model. DTs algorithms such as C4.5 or ID3 belong to this category. They select the best attributes in each node so that the classification is made based on them. After that, they continue dividing the dataset until the stopping condition is reached (Boutemedjet, Bouguila and Ziou, 2009). Embedded methods are less computational intensive from wrapper methods, and they build trees quickly. However, they are specific to the learning algorithm that is used to train the model.

Having these in mind, the feature selection methods to be used in SNIDS are filter and wrapper methods. These methods will be used to extract a feature subset so that training time is reduced and DR is maintained at acceptable levels. In order to select the most appropriate features, the WEKA library will be used. It provides several options for searching the attributes and evaluating them.

"Attribute evaluators" will be used to evaluate the attributes. These methods include: 1) CfsSubsetEval, introduced by Hall and Smith (1998), which assesses each feature subset by taking into account the predictive power of each attribute together with the degree of redundancy between them. The selected attributes have strong relationship with the class and have low interrelationship (Hall and Smith, 1998), 2) Filteredsubseteval which runs a random subset evaluator on data that have been filtered with a random filter, 3) WrapperSubsetEval which uses a learning algorithm to evaluate feature sets. It uses cross-validation (Vanschoren *et al.*, 2014) to calculate the accuracy of the learning algorithm for a set of features (Kohavi and John, 1997).

Search methods are used to search the feature space so that the best subset is selected. After selecting the subset, the attribute subset evaluator will measure the quality of the subset. The most common search methods are BestFirst, GreedyStepwise and Linear Forward Selection (LFS). Starting with BestFirst, it explores for feature subsets by "greedy hill-climbing augmented with a backtracking facility"

(Witten *et al.*, 2016). It can start with an empty set of features and search forward, or start with a full set and search backward or can start at any point going in both directions. The second method is GreedyStepwise which performs a greedy forward or backward search through the space of attribute subsets (Witten *et al.*, 2016). The last technique is LFS which is an extension of BestFirst. There are the fixed-set and the fixed-width types of algorithm. The former selects a fixed number of k features while in the latter the k number increases in each step. The algorithm uses the initial order of attributes to select the top k of them or it ranks them using the same attribute evaluator used in the search step. Also, the direction of selecting the features can be forward, or floating forward. The latter means that backward search steps are optional.

3.4. Chapter Summary

This chapter described the SNIDS architecture and its architectural components. The dataset to be used is also explained along with the attacks that are going to be detected. Lastly, the feature selection techniques are introduced to the reader. These methods will be used after the initial results so that DR of SNIDS is further improved.

Chapter 4 – SNIDS Implementation

4.1. Chapter Overview

This chapter describes the implementation of SNIDS. It starts with a discussion about the development environments and programming languages. After that, the actual implementation is described in detail. An analysis of the functionalities of SNIDS is also provided. Then, the implementation of Graphical User Interface (GUI) of SNIDS is explained.

4.2. Development Environments and Programming Languages

4.2.1. Programming Languages

An important aspect of the project is the implementation of SNIDS. The software will be used by researchers or any user that is interested in testing SNIDS. Creating an easy-to-use software is one of the main requirements. In order to build the software, a programming language should be used. The criteria for choosing the programming languages are: 1) being Object-oriented and, 2) having the ability of building a GUI easily. Python, C# and JAVA fulfil the two criteria but JAVA will be used.

JAVA has many advantages. First of all, it is an Object-oriented language which is significant for this work. This means that objects can be created to manage easily the classifiers and classes. Another advantage is that a software developed in JAVA can be executed in any system independent of the system's architecture. Code reusability is another benefit. SNIDS could have future modifications for adding extra functionalities or improving its performance. Many libraries are developed in JAVA. In the case of SNIDS, the WEKA library (Witten *et al.*, 2016) is implemented in JAVA. This was the most important advantage of JAVA over the other candidates.

4.2.2. Development Environments

Nowadays, the use of an Integrated Development Environment (IDE) makes software development an easy task. IDE provides all the tools to write code, test and package the developed software. The most popular IDEs are Microsoft Visual Studio, NetBeans, Eclipse and IntelliJ IDEA. Most of them can be used in any operating system except for Microsoft Visual Studio. Visual Studio can be installed only on Windows. Apart from that, it supports the development of Windows applications only. As a result, Microsoft Visual Studio is not considered as a choice for building SNIDS.

The selection of IDE for developing SNIDS is a task of selecting one of the rest IDEs (NetBeans, Eclipse, IntelliJ IDEA). All three IDEs support JAVA and they provide an interface for building a GUI. Therefore, any of them can be used for this task. Eclipse (Foundation, 2017) is chosen as it has more advantages than the others. The most important benefit is that it allows you to easily install any plugin required for the software. In other words, it provides you with a set of tools for building any component of the software such as a GUI. In addition, it assists the programming by having an internal editor for writing the source code. The editor can help the programmer by pointing out coding errors and providing coding hints while typing. The programmer can customize the editor or the environment with a few clicks. Eclipse has also an integrated revision control system which helps in keeping all the changes of source code (Tony De Vita, 2014).

4.3. SNIDS Implementation

4.3.1. Basic Functionalities and Configuration

The goal of developing SNIDS is to provide an easy-to-use interface demonstrating HAN-let and NAN-let functionalities. The basic features of SNIDS are the following:

- Classify data into “normal” and “anomaly” accordingly. This is done by using C4.5 (called J48) algorithm in HAN-let. The reason for choosing C4.5 for HAN-let is due to its fast processing and its high DR. This is suitable in a real-time environment where attack detection must be fast. Regarding NAN-let, the Vote meta-classifier will be used to combine SVM and RF classifiers. Combining

means that the result of both classifiers will be considered to produce the class label. The selection of SVM and RF classifiers is based on their performance from the literature (Chauhan *et al.*, 2013; Choudhury and Bhowal, 2015; Yang and Li, 2006). These classifiers have high DR on NSL-KDD dataset (NSL-KDD Dataset, 2015).

- Evaluate HAN-let and NAN-let using a testing dataset or k-fold cross-validation. For the former, a dataset should be selected by the user before running the program. NSL-KDD is the recommended dataset for IDS. In case of k-fold cross-validation (Vanschoren *et al.*, 2014), the default value of folds is 10. However, the number of folds can be changed by the user.
- Import datasets with extension .arff, .xarff, .csv, C4.5 .data and .name files. The files with .arff extension are recognised by the WEKA library (Witten *et al.*, 2016). These files contain header information which is followed by data information. SNIDS can accept all the file extensions supported by WEKA (weka, 2015). In case of file loading problem, an exception is raised.

The configuration used for developing SNIDS was a laptop computer carrying 8 GB RAM and an Intel i7 3rd generation CPU. The operating system was Microsoft Windows and the programming environment was Eclipse IDE for Java EE Developers (Eclipse, 2017).

4.3.2. HAN-lets and NAN-lets

Implementing the software components requires coding in JAVA the functionalities mentioned before. The first step in implementing HAN-let and NAN-let was to import the WEKA library (Witten *et al.*, 2016) into the Eclipse project. According to Özgür and Erdem (2016), WEKA is one of the most widely used tools for comparing classifiers. This is because it contains the implementation for many classification algorithms used in IDS research. For the current project, the WEKA library is used to implement the J48, SVM, RF and Vote classifiers. Using this common library allows SNIDS to be extended and improved in the future by other researchers of the field.

During implementation, the WEKA API documentation was followed (Grepcode.com, 2014). The WEKA API contains all the information about the functions supported by the "weka.jar" library. Regarding HAN-let, the goal was to configure the J48 algorithm and run it by calling the proper function. This was achieved by creating a new J48 object, setting any optional parameters, and then calling the function for training the classifier. The next step was the testing phase. This can be done by using a testing dataset or cross-validation. For using the former, a testing set from NSL-KDD dataset must be imported. For using cross-validation, the number of folds must be set. The default value is 10. The HAN-let's source code is in the *trainHANEval(String file)* function inside "MethodsIDS.java" class. The "file" parameter is a file created by the classifier containing the predictions. This option can be enabled by the user.

Similarly, NAN-let logic is implemented in *trainNANEval(String file)* function inside "MethodsIDS.java" class. The development of NAN-let differs from HAN-let. The former uses Vote meta-classifier while the latter uses the J48 classifier. Vote combines SVM and RF classifiers. Deploying the SVM algorithm required importing the LibSVM library (Chang and Lin, 2011) into the project. The library contains various formulations of SVM that can be used in a software. Parameters for SVM and RF can be found in "NAN-let options" of SNIDS menu.

Regarding the coding of NAN-let, Java objects for SVM, RF and Vote classifiers were firstly created. Then, each classifier was configured using its default options. The Vote classifier was configured to combine the two classifiers. After coding classifiers' configurations, the function for training them was called. Regarding evaluation, this can be done using a testing dataset or cross-validation. Again, testing dataset is the default option and 10-fold cross-validation is the alternative method. NAN-let can also generate a file with predictions when requested by the user.

4.3.3. Combining classifiers

As discussed earlier, NAN-let implements the Vote meta-classifier (Kuncheva, 2004; Kittler *et al.*, 1998). This classifier is included in WEKA library (Witten *et al.*, 2016) which is used in this work. Vote can combine any classifier using different combination rules. The available combination rules are “Average of Probabilities”, “Product of Probabilities”, “Majority Voting”, “Minimum Probability”, “Maximum Probability” and “Median”. Any of these rules can be selected in SNIDS options. However, the default rule used in SNIDS is “Average of Probabilities”. This rule calculates the average of the probability distributions for every classifier used. According to the source code (Grepcode.com, 2014), this is done by creating an array with probabilities of classifiers trained both within and outside Vote classifier. Then, it divides each probability by the number of models used. In NAN-let, Vote uses three models: one SVM and two RF classifiers. The reason for using two RF classifiers is because of better results obtained with this configuration. Consequently, RF has two votes instead of one. That means the classification result will be more affected by the result of RF than by SVM.

4.3.4. User Interface

One of the requirements for building SNIDS was to have a user-friendly interface. A GUI was developed for this purpose. It required to install the WindowBuilder plugin for Eclipse (Anon., n.d.). According to Anon. (n.d.), WindowBuilder “is composed of SWT Designer and Swing Designer and makes it very easy to create Java GUI applications without spending a lot of time writing code”. Using this plugin someone can design an interface with different components. Components such as JLabel, JRadio button, JButton, JPanel, JMenu and JCheckbox were used in GUI design.

The main screen of SNIDS is shown in Figure 4.1. This is the welcome screen with a message introducing the tool to the user. Moreover, it provides some instructions on how to use the software. The focus here is the left side where someone can run the HAN-let or the NAN-let. This can be done as follows: firstly, the user must choose the HAN-let or NAN-let radio button. Then, user must import training dataset from “*File->Import train data*” (Figure 4.1). In case of using test dataset, user can “*Import test data*”. Otherwise, user should select cross-validation from “*Options->General*

Options" (Figure 4.2). After importing datasets, the last step is to click the "*Start*" button. Pressing the button will run the HAN-let (J48 classifier) or NAN-let (SVM and RF classifiers combined with Vote). For terminating the program, user must click the "*Exit*" button.

As can be seen, GUI makes SNIDS' execution procedure a simple task. Users can classify data easily using HAN-let or NAN-let software component. In this way, the difference in performance and DR of these components can be seen easily.

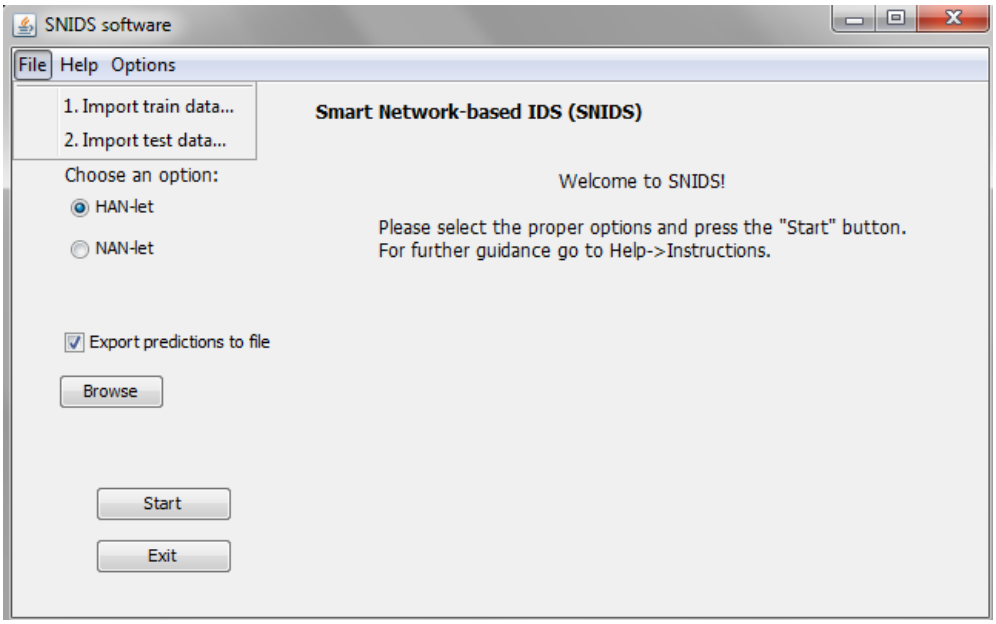


Figure 4.1: *SNIDS main screen and File menu.*

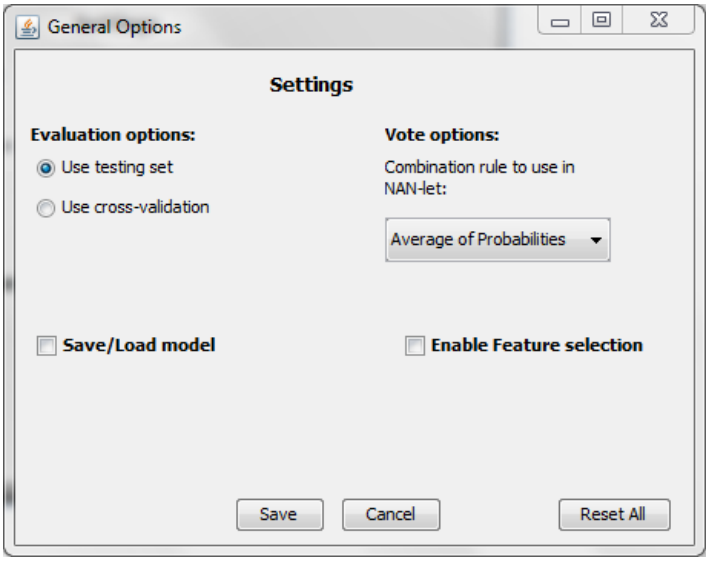


Figure 4.2: *SNIDS General Options*

4.3.5. Additional Functionalities

Apart from the main features, some extra functionalities are implemented for better user experience. Below these features are described:

- *Export classifier's predictions into a .csv file*

  Both HAN-let and NAN-let support exporting the predictions of their classifiers into a .csv file. This feature helps the user to see the instances and the prediction made by the classifier for each instance. The generated file has 5 columns: #inst (instance number), actual (actual class value), predicted (predicted class value), error (shows "+" when prediction failed to match the actual class), prediction (probability that instance belongs to the predicted class). Using the developed GUI, user can select the path and the name of the file to be created. The file is created after the evaluation phase is finished. In case of low disk memory, an exception will appear.

- *Save/Load a model from file*

  SNIDS supports saving a trained model into a file. This means that users can save a trained HAN-let or NAN-let software component into a file. Moreover, the saved model can be loaded into SNIDS. Importing the model saves time as it skips the training phase. After importing the model, user can test the model by using a test dataset or cross-validation. SNIDS only accepts the models exported for HAN-let and NAN-let. The extension for exported/imported files is ".model".

- *Customize classifiers by selecting various parameters*

  Users can customize classifiers by choosing different options. This is possible thanks to the GUI which provides a simple interface to select algorithm's parameters. All the classifiers (J48, SVM, RF, and Vote) have different parameters which can be adjusted by the user. For example, users can select the option to visualize the decision tree of J48 in HAN-let. Options

should be chosen before running the algorithms. If no changes happen, the algorithms use the default options which are described in WEKA documentation (WEKA, n.d.).

- *Feature selection*

  This option allows both HAN-let and NAN-let to select the most appropriate features before classification. Filter and wrapper methods (Hall, 1999; John, Kohavi and Pfleger, 1994) are available for selecting the features. Filter method can be selected if user wants fast attribute selection by just passing the training data through a filter. Wrapper method takes more time to select the attributes. The search method used in filter method is BestFirst and the attribute evaluator is CfsSubsetEval. For the wrapper evaluator, the base classifier is J48 algorithm and the search method is again BestFirst. Regarding search methods, all the methods available in WEKA produce the same results. So, BestFirst is selected randomly from the available methods. Feature selection is implemented in SNIDS so that the dimensionality of the datasets is reduced. The result is a reduction in classifier's running time, and improved DR. In other words, SNIDS can detect intrusions faster by using only the most significant attributes. Moreover, it keeps the detection percentage at the same levels as with using all the attributes.

All in all, most of the above functionalities are implemented in SNIDS to improve the efficiency. SNIDS gives the option to users to customize the classifiers and see the actual predictions. Although GUI is designed for better user interaction, feature selection is the most important functionality. This is because it improves classifier's detection ability and reduces training time. This makes it suitable for working in real-time environments. The source code of SNIDS can be found in philok93 (2017).

4.4. Chapter Summary

As already mentioned, developing SNIDS is a vital part of this project. Therefore, the details about implementation are described in this chapter. Information about the development of HAN-let and NAN-let software components is provided. Basic and extra functionalities of SNIDS are also explained in detail. Furthermore, the designed GUI is demonstrated. Users can use the GUI for executing software components easily.

Chapter 5 – Testing and Evaluation

5.1. Chapter Overview

This chapter presents the evaluation results of SNIDS. Testing methodology and performance metrics are explained in the first place. SNIDS' performance is tested by running the software components and exporting the results of metrics. Then, the evaluation of SNIDS' components is shown. In the last section, SNIDS is compared with similar IDSes from literature in terms of Accuracy, Detection Rate (DR) and False Positive (FP) rate.

5.2. SNIDS Testing Methodology

In this project, SNIDS is tested to verify that it works properly. During testing phase, any unseen bugs can be found. It can be also verified that the software classifies data into normal and anomaly. Another purpose of testing, is to identify any possible bottlenecks.

There are two testing methodologies to follow: functional and non-functional. Unit testing is a functional testing method in which each module of the software is tested to ensure that it operates correctly. In SNIDS, test cases were created for HAN-let and NAN-let to test their functionality. JUnit (Junit-About, 2017) was used to test SNIDS' performance and workflow. It is an external plugin for Eclipse.

Apart from unit testing, evaluation of IDS performance is important. Performance evaluation is a non-functional method of testing. SNIDS' detection performance is checked in terms of DR. In the field of IDS, high DR and low FP are desired. This means that most of the actual attacks are detected. Additional performance metrics such as memory consumption and time needed to test the classifiers will be used. Having a fast tool for classifying network packets is one of the expectations of SNIDS.

5.3. Performance Metrics

Performance evaluation is based on four parameters; True Positive (TP), True Negative (TN), False Negative (FN) and False Positive (FP). The definitions of them are

as follows: 1) TP shows the number of instances that are predicted as an attack correctly, 2) TN represents the number of instances that are correctly predicted as normal, 3) FN shows the number of attacking packets which have been incorrectly classified as normal packets. In other words, it shows the wrong prediction, 4) FP indicates the number of normal packets which have been incorrectly treated as attacking packets. These four parameters can be presented in a confusion matrix. Confusion matrix is a table showing the actual and predicted classifications applied by a classifier (Provost and Kohavi, 1998). Table 5.1 presents an example of confusion matrix showing the connection between the 4 parameters (TP, TN, FP, and FN).

Actual	Predicted	
	Normal	**Attack**
Normal	TN	FP
Attack	FN	TP

Table 5.1: *Confusion matrix*

Four metrics are used for SNIDS' evaluation. These metrics are based on the above parameters. The four metrics which were selected from the list of *Özgür and Erdem* (2016) and *Wu and Banzhaf* (2010) are the following:

- **True Positive (TP) rate or Detection rate (DR) (%):** $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$, indicates the rate of correctly detected attacking packets. High DR means better detection performance.
- **False Positive (FP) rate or False alarm (%):** $\frac{\mathrm{FP}}{\mathrm{TN+FP}}$, shows the rate of normal packets which have been incorrectly classified as attacking packets. Low FP rate is desired for SNIDS.
- **Accuracy (%):** $\frac{TP+TN}{TN+TP+FN+FP}$, shows the percentage of correct predictions from the total number of predictions. High accuracy is desired for SNIDS.
- **Testing time (sec):** the time taken by the classifier to evaluate the dataset. Fast data classification is required by an IDS.

5.4. SNIDS Evaluation Results

5.4.1. Evaluation Procedure

The evaluation of SNIDS is significant because it will show the effectiveness of the software components. In order to evaluate the system, a dataset must be used. As discussed in previous chapters, the NSL-KDD dataset (NSL-KDD Dataset, 2015) will be used for training and testing SNIDS. This dataset is structured as an ARFF file so that SNIDS can read it. Except for using NSL-KDD test set, 10-fold cross-validation (Vanschoren *et al.*, 2014) will be used for testing. This means that the NSL-KDD training dataset will be divided into 10 random subsets. Then, one subset will be used for evaluation while the rest 9 will be used for training. This process is repeated 10 times.

The evaluation procedure is as follows: firstly, the components are trained using KDDTrain+_20Percent dataset. Then, components are tested using both test dataset and 10-fold cross-validation. Moreover, feature selection techniques are used in testing phase. As mentioned in chapter 3, filter and wrapper methods (Hall, 1999; John, Kohavi and Pfleger, 1994) are the two techniques to be checked. After each experiment, the results are extracted. This procedure is applied for both HAN-let and NAN-let.

Regarding feature selection, the attribute evaluator CfsSubsetEval and the wrapper evaluator WrapperSubsetEval from WEKA API (Witten *et al.*, 2016) are used. As discussed in the previous chapter, filter method uses CfsSubsetEval as evaluator and BestFirst as search method. The result from running the filter on KDDTrain+_20Percent dataset is to select the features 4,5,6,12,26,29,30,37. So, 8 features out of the 41 are chosen. The WrapperSubsetEval method is configured to run with J48 classifier and BestFirst as search method. After running it, the features 1,3,5,7,15,23,26,34,38 (9 in total) are selected from the dataset. Both results exclude the ‘class’ column. With regard to the performance of these methods, wrapper method needed more processing time than filter method. This is due to the use of J48 from wrapper method to evaluate features.

5.4.2. HAN-let Evaluation

This section presents the evaluation of HAN-let. There are 6 test cases. In three test cases, a testing dataset is used while the rest test cases use 10-fold cross-validation. Also, in four test cases a feature selection method is applied. The filter method in feature selection uses 8 features while the wrapper uses 9 features. The results are shown below:

Test #	Using Test set	Using 10-fold Cross-validation	Feature selection		Accuracy (% of correct/ % of incorrect)
			Filter method (8 attr.)	Wrapper method (9 attr.)	
1	YES				81.05/18.95
2	YES		YES		75.59/24.41
3	YES			YES	84.12/15.88
4		YES			99.56/0.44
5		YES	YES		98.61/1.39
6		YES		YES	99.56/0.44

Test # - Method	DR (%)	FP rate (%)	Testing time (sec)
1- Testset	68.9	2.8	0.01
2- Testset	59.7	3.3	0.02
3- Testset	74.6	3.3	0.05
4- Cross Val.	99.6	0.4	11.90
5- Cross Val.	97.6	0.5	7.02
6- Cross Val.	99.6	0.4	12.85

Table 5.2: *HAN-let evaluation results*

According to the tables above, when using a test set the highest accuracy achieved is 84.12% in test #3. In that test case, the DR is the highest but the FP rate is by 0.5% higher than test #1. Regarding testing time, in all testset cases the time is under 1 second so it's negligible. In cases where 10-fold cross-validation is used, tests #4 and #6 have the highest accuracy with a percentage of 99.56%. They also have the same DR and FP rate, 99.6% and 0.4% respectively. This is also presented in the graph below with all the test cases. Consequently, HAN-let can be used with 9 attributes instead of 41 as it will improve the performance and save memory.

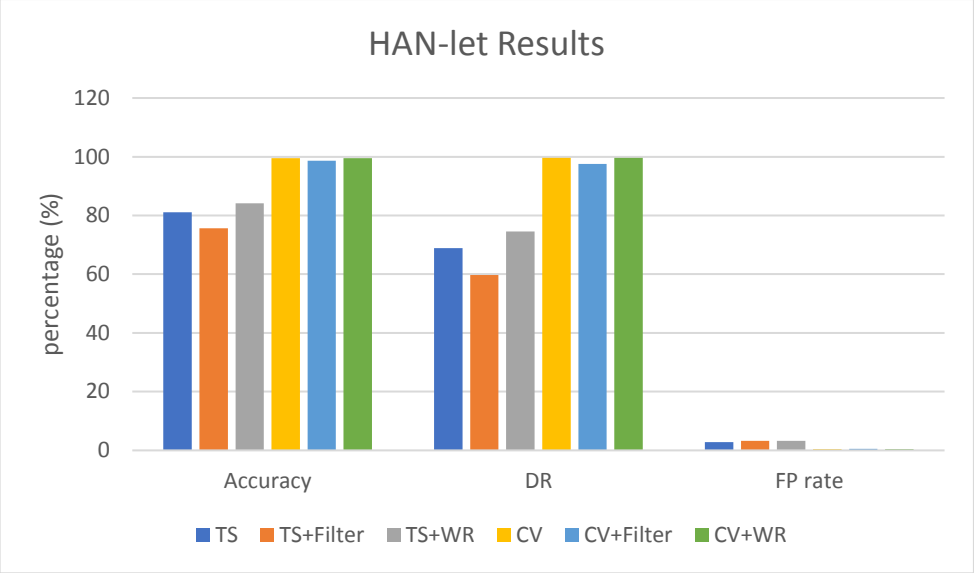


Figure 5.1: *HAN-let evaluation results in graph*

5.4.3. NAN-let Evaluation

This section presents the results obtained from NAN-let evaluation. It follows the same manner with HAN-let evaluation. There are again six test cases; three using test dataset and three using 10-fold cross-validation. Moreover, feature selection is applied in some cases as can be seen below:

Test #	Using Test set	Using 10-fold Cross-validation	Feature selection		Accuracy (% of correct/ % of incorrect)
			Filter method	Wrapper method	
1	YES				76.53/23.47
2	YES		YES		75.20/24.80
3	YES			YES	82.54/17.46
4		YES			99.65/ 0.35
5		YES	YES		98.30/1.70
6		YES		YES	99.71/ 0.29

Test # - Method	DR (%)	FP (%)	Testing time (sec)
1 - Testset	60.8	2.7	95.36
2 - Testset	58.9	3.2	53.62
3 - Testset	71.5	2.9	22.39
4 - Cross Val.	99.4	0.1	4095.38
5 - Cross Val.	98.9	2.3	1978.90
6 - Cross Val.	99.6	0.2	889.33

Table 5.3: *NAN-let evaluation results*

As can be clearly seen, the tests #3 and #6 have the best results. Specifically, test #3 which uses test dataset with 9 attributes has 82.54% accuracy and 71.5% DR. The FP rate is at similar levels with the other test cases. About test #6, the accuracy is 99.71%, the DR is 99.6% and FP rate is 0.2%. These values are slightly higher than test #4. The most important metric here is the testing time. The testing time of test #6, the model with 9 features, was 4.6 times faster than the time needed in test #4. This means that test #6 is by far faster and has slightly better DR than test #4. The reason for that is the use of 9 attributes instead of 41 in test #6. Using less attributes reduces testing time. Also, selecting the most relevant attributes with feature selection seems to produce slightly better DR.

Generally, testing time in tests #4, #5 and #6 is bigger than the first three cases because cross-validation is used. This method is repeated 10 times (10 folds). The classifiers employed in NAN-let play an important role. SVM and RF classifiers need more time to classify data than J48 (HAN-let). That's why the two software components are used in different networks. HAN-let will be employed in HAN for detecting malicious packets quickly while NAN-let will be employed in NAN for classifying large number of packets more accurately.

Looking at the first three tests (#1, #2 and #3), test #2 has the worst DR from all while tests #1 and #3 have a big difference in accuracy and testing time. Test #3 is faster and more accurate than test #1. Again, the use of 9 features instead of 41 helps in reducing processing time and improving DR. The graph below shows the results of NAN-let. Clearly, tests #3 and #6 have the best performance. As a result, NAN-let should be used with 9 attributes to have higher accuracy and reduce processing time.

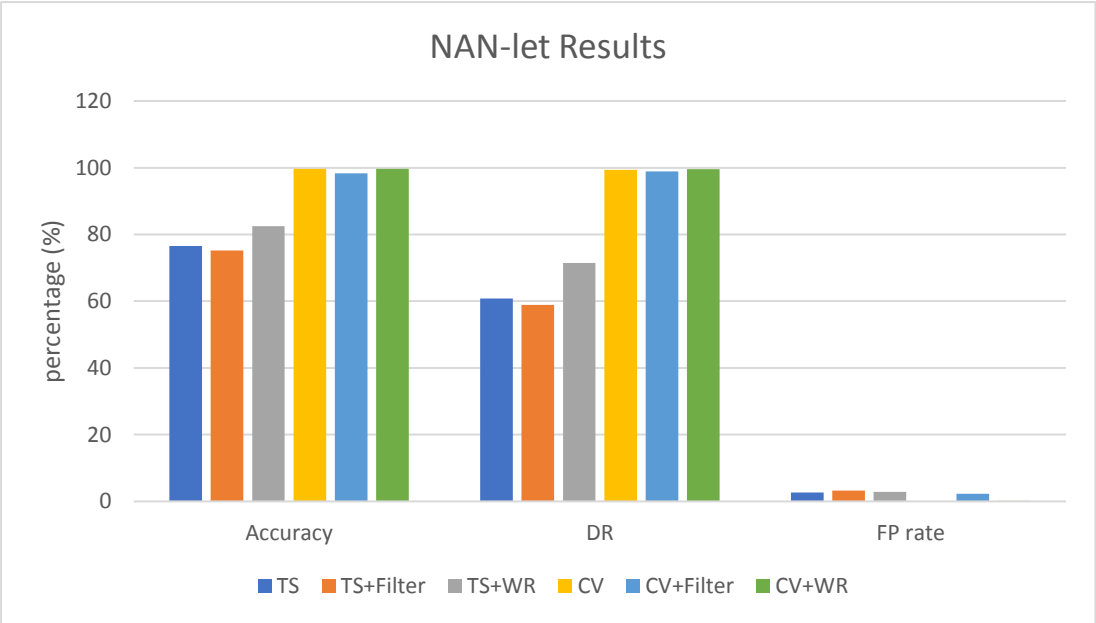


Figure 5.2: *NAN-let evaluation results in graph*

5.4.4. Classifiers Comparison

In this section, classifiers used in software components will be compared. The classifiers to be considered are the following: J48 classifier used in HAN-let, SVM single classifier, RF single classifier, "Vote classifier combining RF with SVM", and "Vote classifier combining two RF classifiers with one SVM classifier". Starting from J48 algorithm, it is the decision tree classifier used in HAN-let to stop attacks in a HAN. SVM single classifier is used only for comparison purposes. Similarly, the RF single classifier. The "Vote classifier combining RF and SVM" was the first attempt in combining classifiers to improve DR. However, the "Vote classifier combining two RF classifiers and one SVM classifier" was finally used in NAN-let. NAN-let's Vote classifier is expected to have better accuracy than single classifiers. Classifiers were tested using both test set and 10-fold cross-validation. Tables 5.4 and 5.5 below show the results:

Case #	Classifier	Cross Val.- Best Accuracy (% correct/ % incorrect)	DR (%)	FP (%)
1	**J48 (Used in HAN-let)**	99.56/ 0.44	99.6	0.4
2	**SVM**	97.40/ 2.60	94.9	0.4
3	**RF**	99.77/ 0.23	99.7	0.1
4	**Vote with RF and SVM**	98.09/ 1.91	96.5	0.5
5	**Vote with 2xRF and one SVM (Used in NAN-let)**	99.71/ 0.29	99.6	0.2

Table 5.4: *Classifiers results using cross-validation*

Case #	Classifier	Test set- Best Accuracy (% correct/ % incorrect)	DR (%)	FP (%)
1	**J48 (Used in HAN-let)**	84.12/15.88 (using wrapper method)	74.6	3.3
2	**SVM**	78.56/21.44 (using wrapper method)	64.4	2.8
3	**RF**	80.54/19.46 (using wrapper method)	67.9	2.8
4	**Vote with RF and SVM**	78.60/21.40 (using wrapper method)	64.4	2.7
5	**Vote with 2xRF and one SVM (Used in NAN-let)**	82.54/17.46 (using wrapper method)	71.5	2.9

Table 5.5: *Classifiers results using test dataset*

Looking at the results, it is clear that J48 classifier used in HAN-let has one of the best performances. It has 99.6% and 74.6% DR when using cross-validation and test dataset respectively. However, the FP rate remains at high levels in comparison with the other classifiers. This means that in a HAN it will detect many attacks with a small chance of wrong prediction.

According to the tables, combining two classifiers using the Vote classifier (case #5) seems to improve DR. When cross-validation is used, “Vote classifier combining 2xRF classifiers and one SVM classifier” has better accuracy than SVM. On the other hand, RF (case #3) is slightly better than the classifier of case #5. The difference in DR and FP rate is only 0.1% which is negligible. The outcomes from using the test set

indicate that NAN-let's Vote classifier outperforms RF (case #3), SVM (case #2) and "Vote with RF and SVM" (case #4). The values of two out of three metrics show that NAN-let's Vote classifier has the best performance. Only FP rate is by 0.2% higher than the lowest percentage. Having this in mind, it can be argued that combining classifiers can help in improving the accuracy of the IDS. That's why NAN-let component uses Vote classifier to combine RF and SVM classifiers. Yet, when cross-validation is used RF has slightly better performance than NAN-let's Vote classifier. The graphs below illustrate the results of classifiers for the cases of using cross-validation and test dataset. It can be distinguished that the performance of the classifiers used by both HAN-let and NAN-let components is the best.

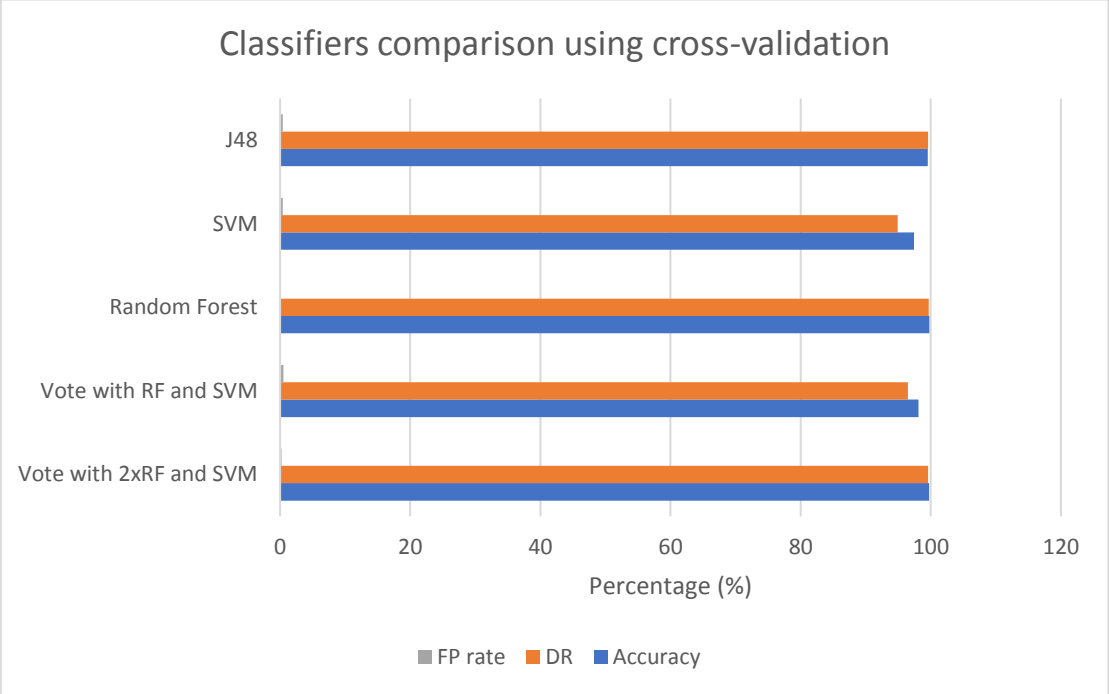


Figure 5.3: *Classifiers comparison using cross-validation*

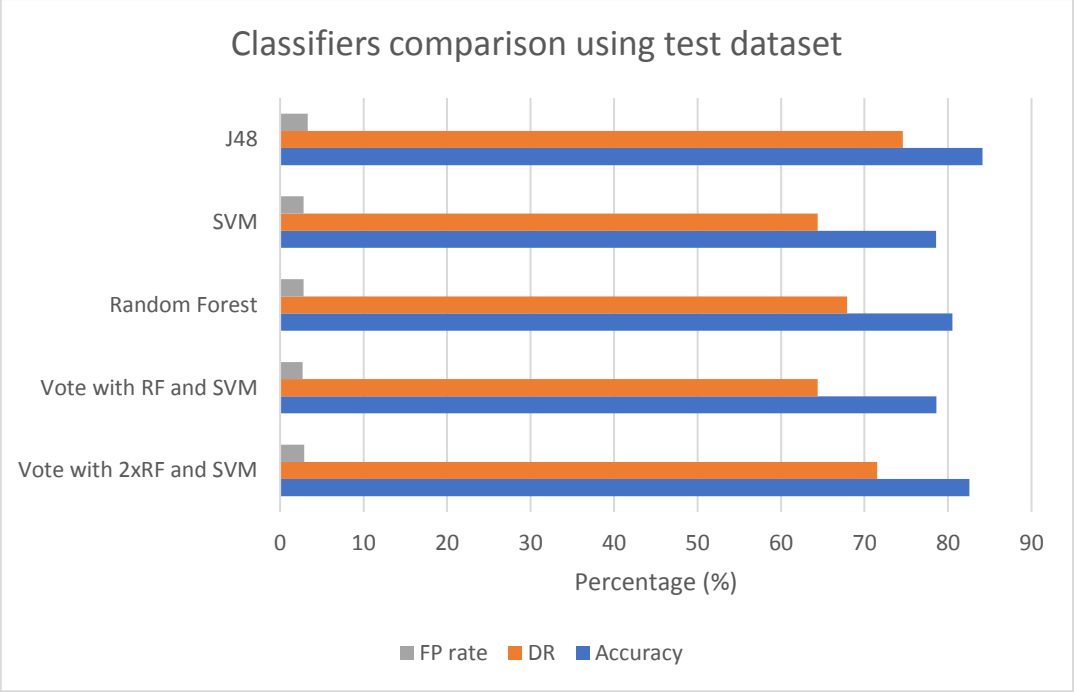


Figure 5.4: *Classifiers comparison using test dataset*

5.4.5. Memory performance

Evaluating the memory performance of SNIDS is part of the evaluation procedure. Both HAN-let and NAN-let components should have low memory requirements. This is because they will be employed in devices with low processing and memory capabilities. For measuring memory consumption, Java VisualVM tool (Java Documentation, 2016) is used. This software provides detailed information about Java processes running on a Java Virtual Machine (JVM). Using this tool, SNIDS' memory consumption was recorded during running time. The screenshot below shows the memory consumption of HAN-let. The column "Live Bytes" indicates the memory used by the component. According to the results, HAN-let needs approximately 7 MB of memory. That memory usage is measured during the execution of the classifier. It also calculates the memory used by the objects created by Java libraries and the software component.

Class Name - Live Allocated Objects	Live Bytes	Live Bytes	Live Objects	Generations
double[]		3.766.14... (52,9%)	8.810 (13,5%)	8
weka.core.**DenseInstance**		1.676.99... (23,6%)	52.406 (80,4%)	7
int[]		827.912 B (11,6%)	1.565 (2,4%)	3
weka.core.**Instance[]**		666.328 B (9,4%)	117 (0,2%)	1
byte[]		102.920 B (1,4%)	10 (0%)	2
weka.classifiers.evaluation.**NominalPrediction**		26.720 B (0,4%)	668 (1%)	1
weka.classifiers.trees.j48.**Distribution**		10.560 B (0,1%)	330 (0,5%)	1
weka.classifiers.trees.j48.**C45Split**		10.440 B (0,1%)	145 (0,2%)	1
double[][]		8.376 B (0,1%)	343 (0,5%)	1
weka.core.**WekaEnumeration**		7.320 B (0,1%)	305 (0,5%)	1
weka.core.**Instances**		4.400 B (0,1%)	110 (0,2%)	1
java.util.**ArrayList**		3.216 B (0%)	134 (0,2%)	3
java.lang.**Class[]**		1.720 B (0%)	59 (0,1%)	5
weka.classifiers.trees.j48.**C45PruneableClassifierTree**		1.296 B (0%)	27 (0%)	1
weka.classifiers.trees.j48.**C45Split[]**		736 B (0%)	4 (0%)	1
weka.core.**Attribute**		640 B (0%)	16 (0%)	3
weka.classifiers.trees.j48.**NoSplit**		600 B (0%)	25 (0%)	1
weka.core.**Instances[]**		408 B (0%)	17 (0%)	1
java.lang.**String**		408 B (0%)	17 (0%)	3
java.lang.**String[]**		344 B (0%)	14 (0%)	1
java.lang.**Integer**		304 B (0%)	19 (0%)	2
java.lang.**Object[]**		224 B (0%)	9 (0%)	2

Figure 5.5: *HAN-let memory consumption*

Regarding NAN-let, its memory consumption is calculated to be 22 MB approximately. As can be seen, there is a big difference in memory requirements for the two components. This is due to the classifiers used. NAN-let uses RF and SVM classifiers which need more memory for calculations than J48 classifier of HAN-let.

Class Name - Live Allocated Objects	Live Bytes	Live Bytes	Live Objects	Generations
float[]		15.722.3... (71,5%)	156 (0,2%)	1
double[]		3.763.55... (17,1%)	5.271 (6,5%)	2
libsvm.**svm_node**		921.120 B (4,2%)	38.380 (47,1%)	1
weka.core.**DenseInstance**		678.208 B (3,1%)	21.194 (26%)	1
libsvm.**svm_node[]**		202.368 B (0,9%)	2.648 (3,3%)	1
weka.core.**Queue$QueueNode**		127.464 B (0,6%)	5.311 (6,5%)	1
weka.core.**Instances**		106.480 B (0,5%)	2.662 (3,3%)	1
libsvm.**svm_node[][]**		100.784 B (0,5%)	1 (0%)	1
libsvm.**Cache$head_t[]**		100.784 B (0,5%)	1 (0%)	1
byte[]		100.696 B (0,5%)	3 (0%)	2
libsvm.**Cache$head_t**		85.728 B (0,4%)	2.679 (3,3%)	1
java.util.**ArrayList**		63.624 B (0,3%)	2.651 (3,3%)	1
float[][]		7.944 B (0%)	331 (0,4%)	1
int[]		1.040 B (0%)	7 (0%)	2
weka.core.**Capabilities$Capability[]**		672 B (0%)	7 (0%)	2
java.lang.**Class[]**		568 B (0%)	19 (0%)	2
weka.core.**Attribute**		440 B (0%)	11 (0%)	1
java.lang.**StringBuilder**		312 B (0%)	13 (0%)	1
java.lang.**StringBuffer**		216 B (0%)	9 (0%)	1
java.lang.**String[]**		168 B (0%)	6 (0%)	1
java.text.**DecimalFormat**		144 B (0%)	1 (0%)	1
org.bounce.net.**DefaultAuthenticator**		104 B (0%)	1 (0%)	1

Figure 5.6: *NAN-let memory consumption*

5.5. Comparison with Existing Systems

In this section, SNIDS will be compared to other IDSes obtained from the literature. The reason for this comparison is to get an idea of how SNIDS' performance compares with similar system. The metrics to be used for comparison are Accuracy, DR and FP

rate. However, not all studies provide all the metrics. IDSes along with the results can be seen below:

#	Source	Classifier	Accuracy	DR	FP
1	Panda et al. (2012)	END+ Nested Dichotomies +Random Forest	-	99.5%	0.1%
2	Chae et al. (2013)	J48 with feature selection	99.794% / 99.763% - using 22 features / using 42 features	-	
3	Shrivas and Mishra (2016)	Ensemble C4.5 and CART	99.67%		
4	Kosamkar and Chaudhari (2013)	SVM with CFS (Correlation Feature Selection)	98.30%	98.62%	1.01%
5	*HAN-let (SNIDS)*	*J48 with wrapper method (feature selection)*	*99.56%*	*99.6%*	*0.4%*
6	*NAN-let (SNIDS)*	*Vote with two RF and one SVM*	*99.71%*	*99.6%*	*0.2%*

Table 5.6: *Comparison of IDSes from various studies*

According to Table 5.6, there are four studies that propose four different IDSes using different classifiers. Starting from the first study, it uses a combination of RF with Nested Dichotomies. The values for DR and FP rate are 99.5% and 0.1% respectively. This means that NAN-let's classifiers are by 0.1% better in DR but FP rate is by 0.1% higher. *Chae et al.* (2013) use J48 with feature selection. They achieve an accuracy of 99.79% using 2 features. If it is compared to SNIDS, HAN-let with J48 classifier achieves 99.56% accuracy using only 8 features. The difference is 0.23%. A study from *Shrivas and Mishra* (2012) shows that an ensemble of C4.5 and CART has an accuracy of 99.67% (no other metrics available). This percentage is by 0.4% lower compared to NAN-let's Vote accuracy. The last study to compare is using SVM classifier with CFS. *Kosamkar and Chaudhari* (2013) achieved 98.30% accuracy, 98.62% DR and 1.01% FP rate. Compared to NAN-let's results, the value of accuracy is by 1.4% lower while DR is almost 1% less than NAN-let's Vote DR. Regarding FP rate, NAN-let's classifiers

achieve far better false alarm rate with 0.2% instead of 1.01% of SVM using CFS. As can be seen, classifiers used in SNIDS' components (both HAN-let and NAN-let) have great performance in comparison with many recent studies. Although a small number of already proposed methods may achieve slightly better results, SNIDS seems to be better in many cases.

5.6. Chapter Summary

In this chapter, the results of SNIDS' evaluation are presented. Firstly, the evaluation procedure is explained. Then, the results of different metrics are extracted for both HAN-let and NAN-let. A comparison among the classifiers used in SNIDS' software components is also presented. Conclusively, combining multiple classifiers produces better results than single classifiers. Apart from detection performance, SNIDS memory performance is evaluated and discussed. In the last section, SNIDS' performance is compared to other IDSes from the literature. Results show that detection performance of SNIDS' components is sometimes better in comparison with existing IDSes.

Chapter 6 – Conclusions and Future work

This chapter summarizes the results of the research carried out in this project. Furthermore, further improvements are suggested.

6.1. Conclusions

The project's aim was to create a new IDS, called SNIDS, for protecting the AMI. SNIDS consists of HAN-let and NAN-let software components. HAN-let is for HAN protection and NAN-let for NAN protection. High DR and low FP rate is desired from an IDS. Achieving these in SNIDS required studying the literature for existing IDSes and selecting the most appropriate classifiers. According to many studies, J48, SVM and RF classifiers had better performance than others. These classifiers were used in HAN-let and NAN-let. Regarding implementation, it was coded in Java using the WEKA library. Moreover, a GUI was developed for better user interaction. After finishing the implementation, SNIDS was evaluated using the NSL-KDD dataset.

According to the results, NAN-let's Vote classifier has better accuracy, DR and FP rate than existing IDSes. This means that combining classifiers is more efficient than using single classifiers. Additionally, results indicate that using a feature selection method is recommended. Specifically, having only 9 attributes has better performance than 8 attributes. This is due to the difference in the evaluation procedure of wrapper and filter methods. As regards NAN-let's speed, it's not as fast as HAN-let. HAN-let uses J48 classifier and it has fast processing time with high DR. NAN-let uses two classifiers. Thus, it needs more time to classify data.

There were many challenges during the development of SNIDS. Firstly, the combination of classifiers in NAN-let was a difficult decision. There are many ways to combine classifiers. Still, Vote method was preferred for its simplicity. Using a different method could produce different results. Secondly, building a system for smart devices is a challenge. Smart devices have limited capabilities and that requires developing HAN-let and NAN-let as efficient as possible. Thirdly, evaluation of the system is based only on the one available dataset. NSL-KDD is the latest version of the

old KDD99 dataset. Having more than one datasets available would show the performance of SNIDS in different conditions. Also, creating a dataset with the latest attacks would give a better idea of the real system's performance. Another challenge is the evaluation with other IDSes. Many studies from the literature do not provide the same metrics. This makes it difficult to compare with other systems.

6.2. Future work

In this project, an IDS system consisting of HAN-let and NAN-let software components was developed. However, further improvements on the tool can be done. For future work, the author suggests the following:

- Test the software components using real sensor devices. Researchers could experiment with real smart devices in a HAN or NAN environment. The software from SNIDS could be uploaded into a sensor. Then, its real detection performance can be tested.

- Use different datasets for evaluation. As it is known, in the field of IDS there are not many datasets available. As a result, systems are tested using only the KDD99 and NSL-KDD datasets. Creating a new dataset containing real network attacks will help in better evaluation of future IDSes.

- Improve the DR and FP percentages of SNIDS' classifiers. This could be achieved by combining classifiers using a different method than the Vote classifier. Moreover, using different classifiers could have better results. For example, HAN-let could use a different lightweight classifier for improving detection.

- Find the optimal parameters for each classifier. Both HAN-let and NAN-let software components should be configured using the optimal values for their classifiers. Changing the default values for each classifier could improve their performance.

- Minimize memory consumption of software components. Using a different programming language and better programming techniques, the memory consumption could be minimized. This would enable more smart devices with very low memory to run SNIDS' software components.

- Improve the implementation of GUI. Implementing all the parameters of each classifier is not efficient. GUI should have the most important parameters for each of the classifiers. Also, a better interface design could make the GUI more user-friendly.

References

Ali, M. Q. & Al-Shaer, E. (2015). Randomization-Based Intrusion Detection System for Advanced Metering Infrastructure*. *ACM Transactions on Information and System Security (TISSEC),* 18(2), 7.

Anon, (n.d.). WindowBuilder. [online] Available at: https://eclipse.org/windowbuilder/ [Accessed 11 Jul. 2017].

Arumugam, M., Thangaraj, P., Sivakumar, P., & Pradeepkumar, P. (2010). Implementation of two class classifiers for hybrid intrusion detection. In *Communication and Computational Intelligence (INCOCCI), 2010 International Conference on* (pp. 486-490). IEEE.

Beigi-Mohammadi, N., Misic, J., Khazaei, H., & Misic, V. B. (2014). An intrusion detection system for smart grid neighbourhood area network. In *Communications (ICC), 2014 IEEE International Conference on* (pp. 4125-4130). IEEE.

Bennett, C., & Highfill, D. (2008). Networking AMI smart meters. In *Energy 2030 Conference, 2008. ENERGY 2008. IEEE* (pp. 1-8). IEEE.

Berthier, R., Sanders, W. H., & Khurana, H. (2010). Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. *In Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on* (pp. 350-355). IEEE.

Boutemedjet, S., Bouguila, N., & Ziou, D. (2009). A hybrid feature extraction selection approach for high-dimensional non-Gaussian data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(8), 1429-1443.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Chae, H. S., Jo, B. O., Choi, S. H., & Park, T. K. (2013). Feature selection for intrusion detection using NSL-KDD. *Recent Advances in Computer Science*, 184-187.

Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3), 27.

Chauhan, H., Kumar, V., Pundir, S. and Pilli, E.S. (2013). A comparative study of classification techniques for intrusion detection. In *Computational and Business Intelligence (ISCBI), 2013 International Symposium on* (pp. 40-43). IEEE.

Choudhury, S. and Bhowal, A. (2015). Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection. In *Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015 International Conference on* (pp. 89-95). IEEE.

Clark, P. (2016). MPs warned of sabotage threat from smart meter hackers. *Financial Times*. [online] Available at: https://www.ft.com/content/325f66b8-8177-11e6-bc52-0c7211ef3198 [Accessed 25 Apr. 2017].

Cleveland, F. M. (2008). Cyber security issues for advanced metering infrastructure (AMI). In *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE* (pp. 1-5). IEEE.

da Silva, A. P. R., Martins, M. H., Rocha, B. P., Loureiro, A. A., Ruiz, L. B., & Wong, H. C. (2005). Decentralized intrusion detection in wireless sensor networks. In *Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks* (pp. 16-23). ACM.

Detections List File. (1999). *MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation*. [online] Available at: https://www.ll.mit.edu/ideval/docs/detections_1999.html [Accessed 10 Jul. 2017].

Eclipse (2017). *Eclipse IDE for Java EE Developers | Packages*. [Online] Available at: http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/oxygenr [Accessed 30 Jun. 2017].

Energy (2014). Smart grids and meters - Energy - European Commission. [online] Available at: https://ec.europa.eu/energy/en/topics/markets-and-consumers/smart-grids-and-meters [Accessed 19 Apr. 2017].

Energy Network Association (2014). Electricity - Smart networks overview. [2015-08-16]. http://www.energynetworks.org/electricity/smart-grid-portal/overview.html

Foundation, E. (2017). *Eclipse IDE*. [Online], Available at: https://www.eclipse.org/downloads/

Giray, S. M., & Polat, A. G. (2013). Evaluation and comparison of classification techniques for network intrusion detection. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on* (pp. 335-342). IEEE.

Goodspeed, T., Highfill, D. R., & Singletary, B. A. (2009). Low-level Design Vulnerabilities in Wireless Control Systems Hardware. *Proceedings of the SCADA Security Scientific Symposium 2009* (S4). (p. 3-1–3-26).

Govindarajan, M., & Chandrasekaran, R. M. (2011). Intrusion detection using neural based hybrid classification methods. *Computer networks*, *55*(8), 1662-1671.

Grepcode.com (2014). *GC: weka-dev-3.7.12.jar - GrepCode Java Project Source*. [Online] Available at: http://grepcode.com/snapshot/repo1.maven.org/maven2/nz.ac.waikato.cms.weka/weka-dev/3.7.12/ [Accessed 26 Jun. 2017].

Haddadi, F., Khanchi, S., Shetabi, M., & Derhami, V. (2010). Intrusion detection and attack classification using feed-forward neural network. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on* (pp. 262-266). IEEE.

Hall, M. A. (1999). Correlation-based feature selection for machine learning.

Hall, M. A., & Smith, L. A. (1998). Practical feature subset selection for machine learning.

Hall, M. A., & Smith, L. A. (1999). Feature Selection for Machine Learning: Comparing a Correlation-Based Filter Approach to the Wrapper. In *FLAIRS conference* (Vol. 1999, pp. 235-239).

Intrusion Detection Attacks Database (1999). *MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation*. [online] Available at: https://www.ll.mit.edu/ideval/docs/attackDB.html [Accessed 10 Jul. 2017].

Java Documentation (2016). *Java VisualVM*. [online] Available at: https://docs.oracle.com/javase/8/docs/technotes/guides/visualvm/index.html [Accessed 24 Jul. 2017].

Jenkins, N. (2010). *Distributed generation*. The Institution of Engineering and Technology.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Machine learning: proceedings of the eleventh international conference* (pp. 121-129).

Jokar, P., Nicanfar, H., & Leung, V. C. (2011). Specification-based intrusion detection for home area networks in smart grids. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on* (pp. 208-213). IEEE.

JUnit-About (2017). *JUnit - About*. [ONLINE] Available at: http://junit.org/junit4/. [Accessed 27 June 2017].

Kalyani, G., & Lakshmi, A. J. (2012). Performance assessment of different classification techniques for intrusion detection. *IOSR Journal of Computer Engineering (IOSRJCE)*, *7*(5), 25-29.

Kendall, K. K. R. (1999). *A database of computer attacks for the evaluation of intrusion detection systems* (Doctoral dissertation, Massachusetts Institute of Technology).

Kerschbaum, F., Spafford, E. H., & Zamboni, D. (2000). Using embedded sensors for detecting network attacks. In *Proceedings of the 1st ACM Workshop on Intrusion Detection Systems*.

Kittler, J., Hatef, M., Duin, R. P., & Matas, J. (1998). On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, *20*(3), 226-239.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), 273-324.

Kohavi, R., & Quinlan, R. (1999). C5. 1.3 Decision Tree Discovery.

Kosamkar, V., & Chaudhari, S. S. (2013). Improved Intrusion Detection System using C4. 5 Decision Tree and Support Vector Machine (Doctoral dissertation, Mumbai University).

Koshal, J., & Bag, M. (2012). Cascading of C4. 5 decision tree and support vector machine for rule based intrusion detection system. *International Journal of Computer Network and Information Security*, *4*(8), 8.

Krebs, B. (2012) FBI: Smart Meter Hacks Likely to Spread. Available from: http://krebsonsecurity.com/2012/04/fbi-smart-meter-hacks-likely-to-spread/ [Accessed 25 Apr 2017].

Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.

Lan, L., & Vucetic, S. (2009). A multi-task feature selection filter for microarray classification. In *Bioinformatics and Biomedicine, 2009. BIBM'09. IEEE International Conference on* (pp. 160-165). IEEE.

Liaw, A., & Wiener, M. (2002). Classification and regression by random Forest. *R news*, *2*(3), 18-22.

Mehmood, T. and Rais, H.B.M. (2016). Machine learning algorithms in context of intrusion detection. In *Computer and Information Sciences (ICCOINS), 2016 3rd International Conference on* (pp. 369-373). IEEE.

Meng, W., Ma, R., & Chen, H. H. (2014). Smart grid neighbourhood area networks: a survey. *IEEE Network*, 28(1), 24-32.

Meng, Y. X. (2011). The practice on using machine learning for network anomaly intrusion detection. In *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on* (Vol. 2, pp. 576-581). IEEE.

Nguyen, H.A. and Choi, D. (2008). Application of data mining to network intrusion detection: classifier selection model. In *Asia-Pacific Network Operations and Management Symposium* (pp. 399-408). Springer Berlin Heidelberg.

NSL-KDD Dataset. (2015). *GitHub-defcom17/NSL_KDD*. [online] Available at: https://github.com/defcom17/NSL_KDD [Accessed 16 Jun. 2017].

Olusola, A. A., Oladele, A. S., & Abosede, D. O. (2010). Analysis of KDD'99 intrusion detection dataset for selection of relevance features. In *Proceedings of the World Congress on Engineering and Computer Science* (Vol. 1, pp. 20-22).

Özgür, A., & Erdem, H. (2016). A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ PrePrints*, 4, e1954v1.

Pan, Y., & Tang, Z. (2014). Ensemble methods in bank direct marketing. In *Service Systems and Service Management (ICSSSM), 2014 11th International Conference on* (pp. 1-5). IEEE.

Panda, M., Abraham, A., & Patra, M. R. (2012). A hybrid intelligent approach for network intrusion detection. *Procedia Engineering*, *30*, 1-9.

Peddabachigari, S., Abraham, A., Grosan, C. and Thomas, J. (2007). Modelling intrusion detection system using hybrid intelligent systems. *Journal of network and computer applications*, *30*(1), pp.114-132.

philok93 (2017). philok93/SNIDS: First release of SNIDS. [online] Available at: https://doi.org/10.5281/zenodo.848834 [Accessed 25 Aug. 2017].

Provost, F., & Kohavi, R. (1998). Guest editors' introduction: On applied research in machine learning. *Machine learning*, *30*(2), 127-132.

Sedjelmaci, H., & Senouci, S. M. (2016). Smart grid Security: A new approach to detect intruders in a smart grid Neighborhood Area Network. In *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on* (pp. 6-11). IEEE.

Shrivas, A. K., & Mishra, P. K. (2016). Intrusion Detection System for Classification of Attacks with Cross Validation. *Probe*, *2*(209), U2R.

So-In, C., Mongkonchai, N., Aimtongkham, P., Wijitsopon, K. and Rujirakul, K. (2014). An evaluation of data mining classification models for network intrusion detection. In *Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on* (pp. 90-94). IEEE.

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the JAM project. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings* (Vol. 2, pp. 130-144). IEEE.

Stolfo, S., Fan, W., & Lee, W. (1999) KDD-CUP-99 Task Description. *http://KDD. ics. uci. edu/databases/kddcup99/task, html*.

Sung, A. H. (1998). Ranking importance of input parameters of neural networks. *Expert Systems with Applications*, *15*(3), 405-411.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. (2003). Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*, *43*(6), 1947-1958.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on* (pp. 1-6). IEEE.

Tong, W., Lu, L., Li, Z., Lin, J., & Jin, X. (2016). A Survey on Intrusion Detection System for Advanced Metering Infrastructure. In *Instrumentation & Mea*

Tony De Vita, Jr. (2014). 'The Advantage of Using Eclipse IDE for JAVA Programming', *HubPages*, 31 July 2014. Available at: https://hubpages.com/technology/The-Advantage-of-Using-Eclipse-IDE-for-JAVA-Programming (Accessed: 20 June 2017)

Vanschoren, J., Van Rijn, J. N., Bischl, B., & Torgo, L. (2014). OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, *15*(2), 49-60.

Vapnik, V. N. (2000). The nature of statistical learning theory, ser. Statistics for engineering and information science. *New York: Springer*, *21*, 1003-1008.

weka (2015). *weka- ARFF (stable version)*. [Online] Available at: https://weka.wikispaces.com/ARFF+(stable+version)

WEKA (n.d.) WEKA API. [Online], Available at: http://weka.sourceforge.net/doc.stable-3-8/ [Accessed June 2017].

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wu, S. X., & Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, *10*(1), 1-35.

Wu, S.Y. and Yen, E. (2009). Data mining-based intrusion detectors. *Expert Systems with Applications*, *36*(3), pp.5605-5612.

Yang, Q., & Li, F. (2006). Support vector machine for intrusion detection based on LSI feature selection. *In Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on* (Vol. 1, pp. 4113-4117). IEEE.

Zhang, J., & Zulkernine, M. (2006). A hybrid network intrusion detection technique using random forests. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on* (pp. 8-pp). IEEE.

Zhang, Y., Wang, L., Sun, W., Green II, R. C., & Alam, M. (2011). Distributed intrusion detection system in a multi-layer network architecture of smart grids. *IEEE Transactions on Smart Grid*, *2*(4), 796-808.

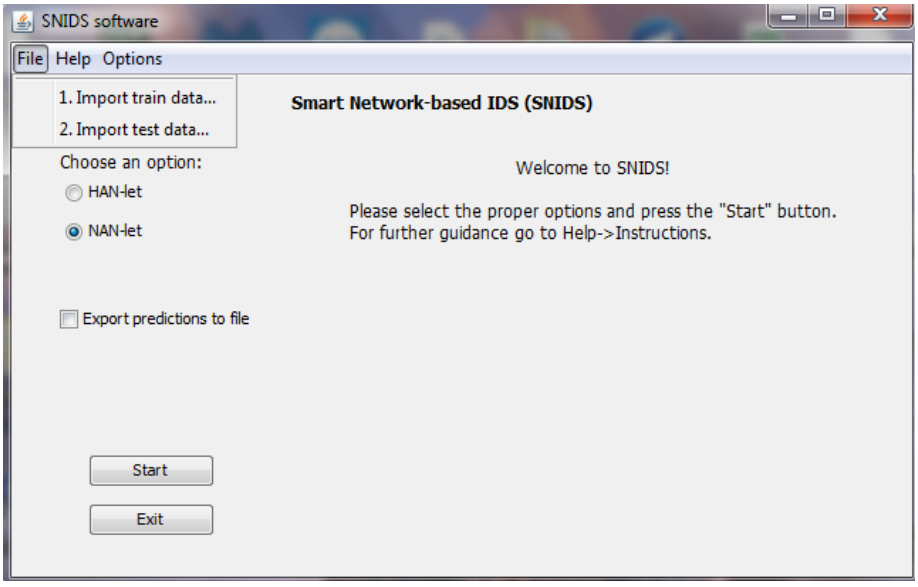
Appendix A

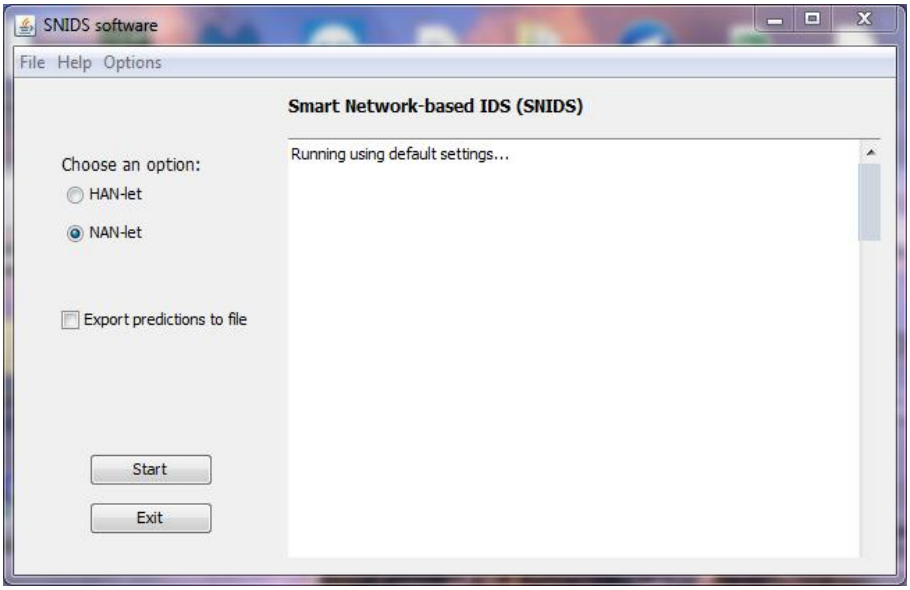
SNIDS Use Case

The execution procedure of SNIDS is described. Also, some screenshots from SNIDS software are shown.

Execution procedure:

User is assumed that wants to run the NAN-let component of SNIDS, using the default options and evaluate it with a testing dataset. Firstly, user chooses the NAN-let radio button. Then, user must import the training dataset from "*File->Import train data*" (Figure A.1.1). The next step is to choose the test method. However, test dataset is the default evaluation method so user just needs to import the test dataset. This can be done by going to "*File->Import test data*". After importing datasets, the last step is to click the "*Start*" button. Pressing the button will execute NAN-let (SVM and RF combined with Vote). The screen in Figure A.1.2 will be shown which means the program is running using default options. After a couple of minutes, the results will be displayed (Figure A.1.3). If user wants to execute the same experiment for HAN-let, it needs to select the "HAN-let" radio button and press the "Start" button. For terminating the program, user must click the "*Exit*" button.





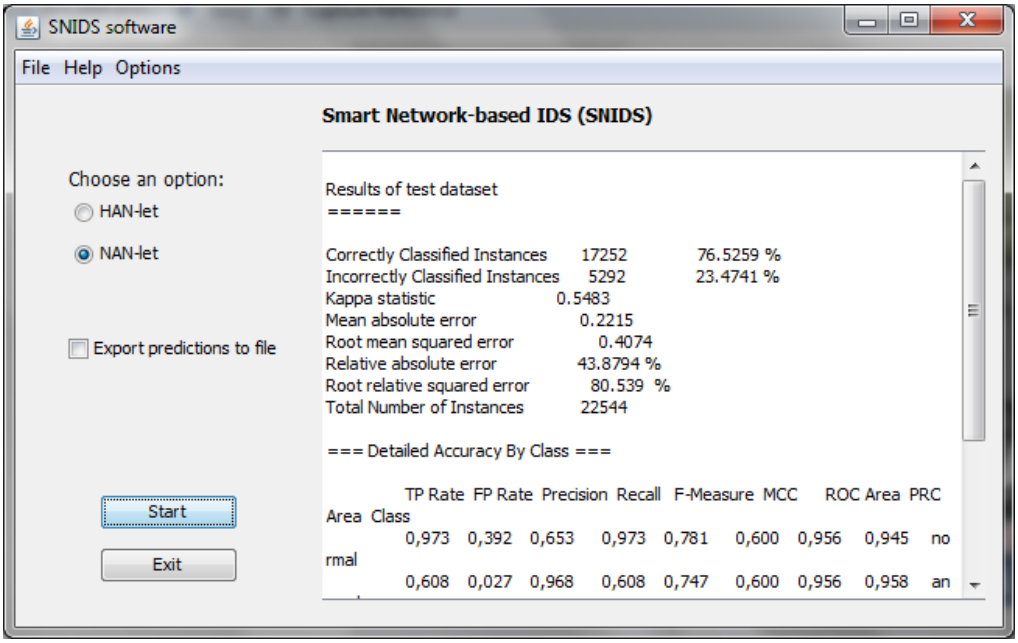
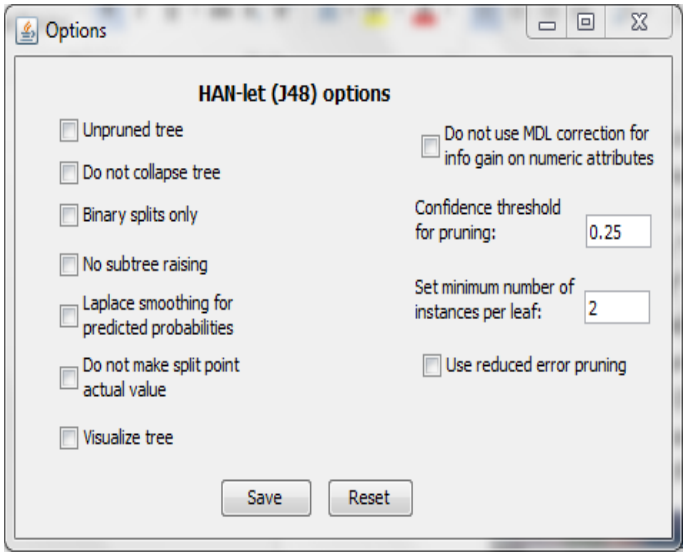
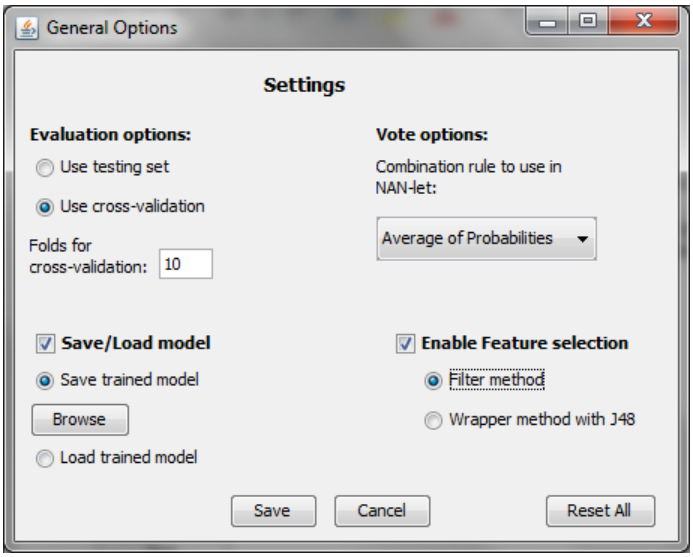
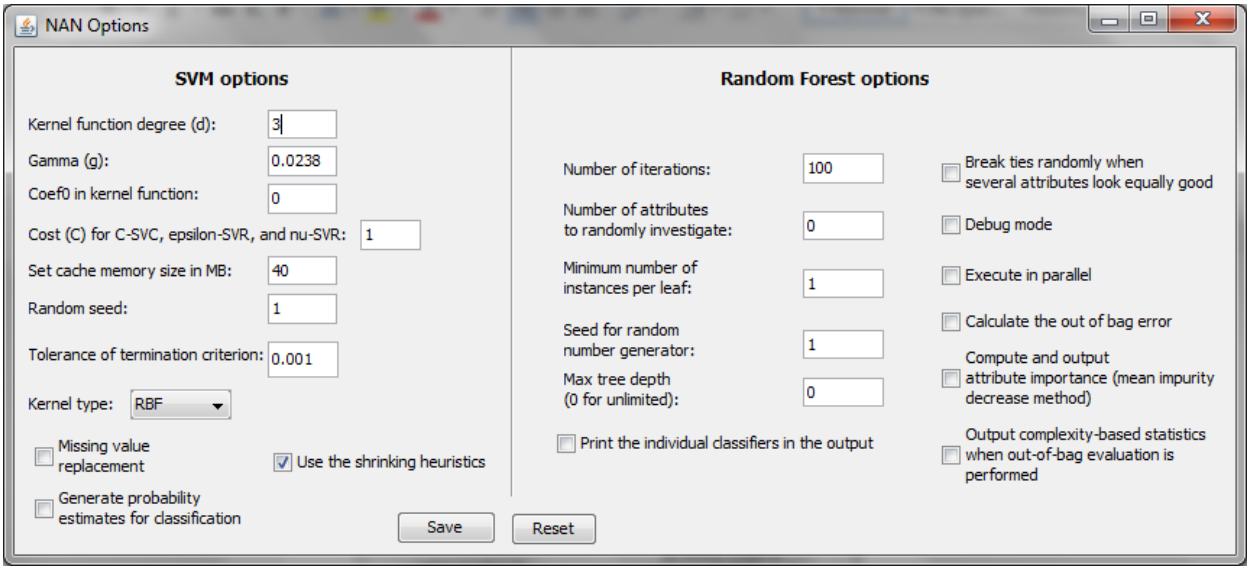


***Figure A.1**: 1) Main screen of SNIDS (Top row), 2) Running NAN-let (Middle row) and 3) Results of NAN-let (Bottom row)*

Below are the screenshots of HAN-let options (left side), General options (right side) and NAN-let options (second row).







Source code of SNIDS can be found in philok93 (2017).

SNIDS Options

The parameters to be selected by the user are explained below. The options for each classifier are obtained from WEKA library (Witten *et al.*, 2016).

HAN-let Options:

Option	Description
Unpruned tree	Whether pruning is performed.
Do not collapse tree	Whether parts are removed that do not reduce training error.
Confidence threshold for pruning	The confidence factor used for pruning (smaller values incur more pruning).
Use reduced error pruning	Whether reduced-error pruning is used instead of C.4.5 pruning.
Number of folds (for error pruning)	Determines the amount of data used for reduced-error pruning. One-fold is used for pruning, the rest for growing the tree.
Binary splits only	Whether to use binary splits on nominal attributes when building the trees.
No subtree raising	Whether to consider the subtree raising operation when pruning.
Laplace smoothing for predicted probabilities	Whether counts at leaves are smoothed based on Laplace.
Do not use MDL correction for info gain on numeric attributes	Whether MDL correction is used when finding splits on numeric attributes.
Do not make split point actual value	If true, the split point is not relocated to an actual data value. This can yield substantial speed-ups for large datasets with numeric attributes.
Set minimum number of instances per leaf	The minimum number of instances per leaf.
Seed for random data shuffling	The seed used for randomizing the data when reduced-error pruning is used.
Visualize tree	Visualizes the decision tree created by the classifier. A new popup window will appear showing the tree.

NAN-let (SVM) options:

Option	Description
Kernel function degree (d)	The degree of the kernel.
Gamma (g)	The gamma to use, if 0 then 1/max_index is used.

Coef0 in kernel function	The coefficient to use.
Cost (C) for C-SVC, epsilon-SVR, and nu-SVR	The cost parameter C for C-SVC, epsilon-SVR and nu-SVR.
Set cache memory size in MB	The cache size in MB.
Random seed	The random number seed to be used.
Tolerance of termination criterion	The tolerance of the termination criterion.
Kernel type	The type of kernel to use.
Missing value replacement	Whether to turn off automatic replacement of missing values. WARNING: set to true only if the data does not contain missing values.
Use the shrinking heuristics	Whether to use the shrinking heuristic.
Generate probability estimates for classification	Whether to generate probability estimates instead of -1/+1 for classification problems.

NAN-let (Random Forest) options:

Option	**Description**
Number of iterations	The number of iterations to be performed.
Number of attributes to randomly investigate	Sets the number of randomly chosen attributes. If 0, int(log_2(#predictors) + 1) is used.
Minimum number of instances per leaf	Sets the minimum number of instances per leaf.
Seed for random number generator	The random number seed to be used.
Max tree depth (0 for unlimited)	The maximum depth of the tree, 0 for unlimited.
Print the individual classifiers in the output	Print the individual classifiers in the output.
Break ties randomly when several attributes look equally good	Break ties randomly when several attributes look equally good.
Debug mode	If set to true, classifier may output additional info to the console.
Execute in parallel	Whether to use multiple slots (threads) for constructing the ensemble.
Calculate the out of bag error	Whether the out-of-bag error is calculated.
Compute and output attribute importance (mean impurity decrease method)	Compute attribute importance via mean impurity decrease

Output complexity-based statistics when out-of-bag evaluation is performed	Whether to output complexity-based statistics when out-of-bag evaluation is performed.

Appendix B

KDD99 Dataset

Below, the attack types of KDD99 Dataset are explained. The information was taken from Olusola *et al.* (2010) and Stolfo *et al.* (1999).

#	Feature name	Description	Type
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. TCP, UDP, etc.	discrete
3	service	network service on the destination, e.g., http, telnet, etc.	discrete
4	src_bytes	number of data bytes from source to destination	continuous
5	dst_bytes	number of data bytes from destination to source	continuous
6	flag	normal or error status of the connection	discrete
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of “wrong” fragments	continuous
9	urgent	number of urgent packets	continuous
10	hot	number of “hot” indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	number of “compromised” conditions	continuous
14	root_shell	1 if root shell is obtained; 0 otherwise	discrete
15	su_attempted	1 if “su root” command attempted; 0 otherwise	discrete
16	num_root	number of “root” accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the “hot” list; 0 otherwise	discrete

22	is_guest_login	1 if the login is a “guest” login; 0 otherwise	discrete
23	count	number of connections to the same host as the current connection in the past two seconds	continuous
24	serror_rate	% of connections that have “SYN” errors	continuous
25	rerror_rate	% of connections that have “REJ” errors	continuous
26	same_srv_rate	% of connections to the same service	continuous
27	diff_srv_rate	% of connections to different services	continuous
28	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
29	srv_serror_rate	% of connections that have “SYN” errors	continuous
30	srv_rerror_rate	% of connections that have “REJ” errors	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous
32	dst_host_count	count of connections having the same destination host	continuous
33	dst_host_srv_count	count of connections having the same destination host and using the same service	continuous
34	dst_host_same_srv_rate	% of connections having the same destination host and using the same service	continuous
35	dst_host_diff_srv_rate	% of different services on the current host	continuous
36	dst_host_same_src_port_rate	% of connections to the current host having the same src port	continuous
37	dst_host_srv_diff_host_rate	% of connections to the same service coming from different hosts	continuous
38	dst_host_serror_rate	% of connections to the current host that have an S0 error	continuous
39	dst_host_srv_serror_rate	% of connections to the current host and specified service that have an	continuous

		S0 error	
40	dst_host_rerror_rate	% of connections to the current host that have an RST error	continuous
41	dst_host_srv_rerror_rate	% of connections to the current host and specified service that have an RST error	continuous

Description of attacks

The table below summarizes the attacks into 4 categories: DOS, R2L, U2R and Probing attacks. The description for each attack type is derived from Kendall (1999), MIT Lincoln Laboratory (1999), and Detections List File (1999).

DOS attacks	Description
Land	The Land attack occurs when an attacker sends a spoofed SYN packet in which the source address is the same as the destination address
Neptune	A Neptune is a DoS attack to which every TCP/IP implementation is vulnerable (to some degree). Each half-open TCP connection made to a machine causes the 'tcpd' server to add a record to the data structure that stores information describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially-open connections. The half-open connections data structure on the victim server system will eventually fill and the system will be unable to accept any new incoming connections until the table is emptied out.
Back	In this denial of service attack against the Apache web server, an attacker submits requests with URL's containing many frontslashes. As the server tries to process these requests it will slow down and becomes unable to process other requests.
Pod (Ping of Death)	The Ping of Death is a DoS attack that affects many older operating systems. It has been widely reported that some systems will react in an unpredictable fashion when receiving oversized IP packets.
Smurf	In the "smurf" attack, attackers use ICMP echo request packets directed to IP broadcast addresses from remote locations to create a denial-of-service attack.
Teardrop	The teardrop exploit is a denial of service attack that exploits a flaw in the implementation of older TCP/IP stacks. Some implementations of the IP fragmentation re-assembly code on these platforms does not properly handle overlapping IP fragments.

U2R attacks	Description
buffer_overflow	Buffer overflows occur when a program copies too much data into a static buffer without checking to make sure that the data will fit.

Loadmodule	The Loadmodule attack is a User to Root attack against SunOS 4.1 systems that use the xnews window system. The loadmodule program within SunOS 4.1.x is used by the xnews window system server to load two dynamically loadable kernel drivers into the currently running system and to create special devices in the /dev directory to use those modules. Because of a bug in the way the loadmodule program sanitizes its environment, unauthorized users can gain root access on the local machine
Perl	The Perl attack is a User to Root attack that exploits a bug in some Perl implementations. Suidperl is a version of Perl that supports saved set-user-ID and set-group-ID scripts. In early versions of suidperl the interpreter does not properly relinquish its root privileges when changing its effective user and group IDs. On a system that has the suidperl, or sperl, program installed and supports saved set-user-ID and saved set-group-ID, anyone with access to an account on the system can gain root access
Rootkit	Multi-day scenario where a user installs one or more components of a rootkit.

R2L attacks	**Description**
Imap	The Imap attack exploits a buffer overflow in the Imap server of Redhat Linux 4.2 that allows remote attackers to execute arbitrary instructions with root privileges.
ftp_write	The Ftp-write attack is a Remote to Local User attack that takes advantage of a common anonymous ftp misconfiguration. The anonymous ftp root directory and its subdirectories should not be owned by the ftp account or be in the same group as the ftp account. If any of these directories are owned by ftp or are in the same group as the ftp account and are not write protected, an intruder will be able to add files (such as an rhosts file) and eventually gain local access to the system.
guess_passwd	Try to guess password via telnet for an account.
multihop	Multi-day scenario in which a user first breaks into one machine.
phf	The Phf attack abuses a badly written CGI script to execute commands with the privilege level of the http server. Any CGI program which relies on the CGI function escape_shell_cmd() to prevent exploitation of shell-based library calls may be vulnerable to attack.
spy	Multi-day scenario in which a user breaks into a machine with the purpose of finding important information where the user tries to avoid detection. Uses several different exploit methods to gain access.

warezclient	Users downloading illegal software which was previously posted via anonymous FTP by the warezmaster.
warezmaster	Anonymous FTP upload of Warez (usually illegal copies of copyrighted software) onto FTP server.

Probing attacks	**Description**
Nmap	Network mapping using the nmap tool. Mode of exploring network will vary options include SYN.
Portsweep	Surveillance sweep through many ports to determine which services are supported on a single host.
Ipsweep	An Ipsweep attack is a surveillance sweep to determine which hosts are listening on a network. This information is useful to an attacker in staging attacks and searching for vulnerable machines. (mit)
Satan	Network probing tool which looks for well-known weaknesses. It operates at three different levels.