Machine Learning on Anomaly-based Network Intrusion Detection Systems

Thiago Roscia Cerdeiro de Lima
trcl10@inf.ufpr.br

Universidade Federal do Paraná
Professor: Luiz Eduardo S. Oliveira

August 8, 2013

This paper presents a summary, which aims to cover the main ideas available on the literature of the past few years regarding network intrusion detection. The goal is to explain the general architecture of Network Intrusion Detection Systems (NIDS) and how different machine learning algorithms are applied. The focus of this review will be the anomaly-based systems, which present a vast field of study.

1 Introduction

The Internet represents today not only a tool used daily, but a whole environment where people put their personal files, companies do bank transactions and governments transfer valuable information. Just as it is possible to connect from a home computer to a web server, it is possible to do the opposite. Any machine connected to the internet is theoretically subject to the access of an outsider. Operating Systems and routers of today are equipped with basic tools to prevent unauthorized access to the computer. Still, there are people dedicated on bypassing such protections. Some of these attackers are successful on copying files, installing malware and even obtaining control over the system without being noticed by the owner.

For that matter, intrusion detection systems (IDSs) were created. They aim to acknowledge unusual behavior within a network. In literature, many papers have been published exploring machine learning algorithms for this matter.

There are several types of attacks and different communication protocols, but the problem of detecting an intrusion can be simplified as a binary system behavior classification problem, where the two possible outcomes would be normal or intrusive. There are two well-defined kinds of approach:

1. *Signature-based*, also known as misuse-based. This is the standard and currently the only approach in commercial use. It has little adaptability to new threats.
2. *Anomaly-based*. Still a field of research, it has a high false positive rate, but good adaptability.

This paper will later focus on the concepts and design of Machine Learning algorithms and Anomaly-based Network Intrusion Detection Systems (A-NIDS).

2 Misuse-based IDS

Signature-based or misuse-based approach has been the most successful for intrusion detection so far. The idea is to compare a network traffic with known patterns of behavior during certain attacks. The major problem is the inability to detect new attacks.

The first systems used only the pattern-recognition approach. This technique has a database of signatures for each known threat. A malicious event is detected when the current state of the network matches a signature. As it tries to match all known signatures and the amount of flowing data grows every year, the computational cost became too large.

A second approach is the implication rules method. It gives a set of rules that describes known events which may lead to infer that an intrusion has occured. Nevertheless, misuse IDSs generally need a human expert to acquire traffic models, who needs to create a new set of rules each time a new attack emerges.

That is the motivation for using data mining techniques on misuse systems. They provide a way to increase the automation on building and adjusting the model. They can adapt the models as they access network traffic containing new threats or by detecting different versions of a known attack. Still, it is impossible to identify compeltely new malicious behavior.

3 Anomaly-based IDS

The anomaly detection technique analyzes current network behavior to check whether or not it corresponds to a normal behavior. The greatest benefit of this approach is the ability to succesfully spot new attacks. The downside is the high rate of false positives. Instead of requiring a new model to adapt, anomaly-based systems require attack-free network traffic data. Data mining has also been used in anomaly-based IDSs, alongside with machine learning and statistical methods.

The different techniques can be grouped into three categories, according to the processing involved: statistical, knowledge-based or machine learning algorithms.

3.1 Statistical

A probability model of certain behavior is created from the captured network traffic activity, such as traffic rate, number of packets and number of different IP adresses.

It does not require prior knowledge as it has the ability to learn the normal behavior from attack-free observations. Another upside of statistical techniques is the possibility to acuratly identify malicious activities ocurring over long periods of time. On the other hand, not all possible behaviors can be modelled and the balance between false positive and false negative highly depends on the correct settings of parameters.

3.2 Knowlege-based

Also called expert systems, knowledge-based systems are implemented by creating a set of classification rules to classify the data. The model is generally created by a human expert. This kind of system does not point out new harmless activities as malicious, therefore granting a reduced number of false positive. For that reason, the set of rules needs to be specific enough and, even though it is possible to reach some level of automatization using finite state machines, it requires high-quality knowledge and significant time to be created.

3.3 Machine Learning

Machine learning methods require a set of labeled data to train the expected behavior model. The main characteristic of this technique is the ability to adapt its classification rules as new data is received. This grants little need for human intervention. The downside of machine learning is the high computational cost. The most popular algorithms used in A-NIDS are:

- Bayesian Networks - creates a network of probability relationships between features.
- Markov Models - compares the probability of the observed data with a fixed treshold.
- Neural Networks - creates a net of perceptrons with the adaptation ability.
- Fuzzy Logic - sees features as fuzzy variables and classifies using continuous values.
- Genetic Algorithms - derives classification rules and selects the most discriminant features.
- Support Vector Machines - finds a hyper-plane that efficiently separate both classification options.
- Decision Trees - creates a tree, where each end-leaf node represents a class, and edges correspond to different values on attributes.
- Clustering - unsupervised algorithm that groups data into clusters based on their similarity.

3.4 Hybrid Classifiers

A hybrid classifier is consisted of two components. The first one preprocesses the input data and sends it to the second classifier, which comes up with the final results. The first component of hybrid classifiers can be used either as a clustering technique to find the classes which the second will decide from; cascading on which the second one classifies what has been rejected; or as an performance optimizer for the second model, which corresponds to the integrated method.

3.5 Ensemble Classifiers

Ensemble classifiers are the combination of weak learning techniques. Each classifier is trained with a different subset of the data, then the test case is processed by all of them and, finally, labeled by the majority.

4 A-NIDS working systems

The review on [5] presents the A-NIDS systems labeled in two categories: commercial and research systems. Commercial systems usually work with signature-based in their core, while research systems implement prototypes and innovative methodologies.

4.1 A-NIDS architecture

While some implementations details may vary, the basic layout of A-NIDS is as presented on [2]: **Traffic Data Acquisition**: gathers information from network frames for further processing.

Traffic Features Generator: extracts features from the captured traffic. Such features can be classified as *low-level* (obtained directly from the raw data), *high-level* (deducted from a subsequent process), *packet* (gathered from packet headers), *flow* (contains information of network connections) and *payload* (obtained from packet payload).

Incident Detector: identifies intrusive activities. Can be either misuse-based or anomaly-based.

Traffic Model Generator: contains the base information used to guide the Incident Detector.

Response Management: initiate maneuvers to outcome a possible intrusion. It is initiated by the incident Detector.

4.2 Commercial

One of the first widely known anomaly detection projects was SPADE, a plugin for Snort that analysed packet transfer searching for anomalous behavior. Meanwhile, Stealthwatch used flow-based anomaly detection.

Recent systems use a distributed architecture by using sensors and a central console to manage the detection process. This the case of DeepSight, which uses a statistical approach.

Most of the commercial systems use a signature-based detection module along with the anomaly-based core. These systems fall into the Hybrid category.

4.3 Research

Research designed NIDS make use of innovative techniques, most of them still in development and unable for commercial use. The paper [5] cites Bro, which uses semantic analysis at the application layer, and EMERALD, which includes rule-based and Bayesian classification.

Former systems focused in statistical approaches, while new ones are diversified, being Markov models, N-grams and SVM the most popular algorithms. A new characterstic of the reasearch A-NIDS is the combination of different specific modules for classification.

5 Data Preprocessing

As previously stated, anomaly-based systems are prone to false positives. Even a 1% false positive rate results in a huge number of bogus alerts.

Data preprocessing is, therefore, required to achieve a better performance on intrusion detection. Preprocessing converts network traffic into a series of observations, where each observation is represented as a feature vector. The following information about data and features was presented on [3].

5.1 Steps

The most relevant steps for NIDS data modeling are the following:

1. *Dataset creation*: identifies representative labeled dataset (either normal or anomalous) for training and testing. Labeling network traffic can be a difficult and time-consuming task, which generally requires an expert.

2. *Feature construction*: creates features with a higher discriminability. These features can be built by a human or by machine learning algorithms.

3. *Reduction*: also called *feature selection*, decreases the dimensionality of the dataset by discarding any irrelevant features. It is used to attenuate the *Curse of Dimensionality*.

5.2 Deriving

Different types of features used in anomaly-based NIDS were covered by [3]. Each feature type is derived from many data preprocessing methods including organizing packets into flows, analyzing application content for fields of interest or parsing individual network packet headers. The derivation of the candidate feature set is an essential step for anomaly-based NIDS. Nevertheless, further preprocessing can help increasing the efficiency of the NIDS.

Preprocessing methods from data mining can be used, including data transformation, reduction, and discretization. A data reduction technique often used was principal component analysis (PCA). PCA has been really useful on reducing the data dimensionality, consequently reducing the computational cost of the detection system.

To eliminate irrelevant and redundant features, several automated feature selection algorithms also exist, resulting in similar data reduction. These data reduction methods provide an precise way of reducing a candidate feature set to a final feature set. It is possible that automated data reduction methods are can further improve the NIDS, although some systems are constructed only with expert domain knowledge to create decent feature sets.

6 Feature sets

6.1 MCD

Great part of the NIDS reviewed in [3] use flow or session network data. Features are then built from the flows. The most popular is the packet header method using multiple connection derived (MCD) features. These features are generally derived using mean, standard deviation, and percentage of flows, covering multiple sessions. Anomaly-based NIDS using these features are able to discriminate between normal network activity and uncommon traffic such as DoS and scanning.

MCD features are generally derived over a time window of connections. Most MCD features are volume-based, such as the count of connections to a particular destination IP address and port in a given time window. Therefore, multiple connection derived features can be easily used to detect uncommon traffic volumes related to DoS or probing. Since individual anomalous packets do not meet the volume-based value, they may be ignored.

6.2 SCD

Single connection derived (SCD) features are used to detect anomalous behavior within a single session. These can highlight an unexpected protocol, uncommon data sizes, unusual packet timing, or uncommon TCP flag sequences. Thereby, SCD features enable the detection of anomalous network usage caused by backdoors, HTTP tunnels etc.

The SCD features provide context which can be used to find contextual anomalies. For example, if the timing of packets within a monitored port connection does not match an expected profile, an anomaly can be raised, since this could mean a protocol tunneling.

6.3 KDD Cup 99

The KDD Cup 99 dataset includes a different SCD and MCD features, including 13 content-based features which can be used to detect R2L and U2R attacks. These content-based features were built by an expert, and involve higher level information such as the number of failed login attempts, if a root shell was obtained, and the number of file creation operations.

Considering the KDD Cup 99 is over ten years old, it is probable these content-based features would not be useful for detecting current attacks in network traffic of the present time, even though such features are very useful for the related dataset. To detect current exploits, entirely different content-based features need to be built.

7 Data

7.1 Types of data

The choice of network traffic features is largely affected by the detection requirements when designing a intrusion detection system. Separate anomaly detectors may be constructed for each of the feature sets if a general anomaly detection is desired. On the other hand, if a specific anomaly detection is wanted, a single feature set can be used.

7.1.1 Packet header

Packet header features have the qualities of being fast, with not much computation and memory needs, and avoid some legal concerns regarding network data analysis. The simplest feature set contains basic features extracted from individual packet headers. These features can be used to flag single packets which are anomalous when compared to a normal training model, or as a filtering process so only uncommon packets are used on further detection algorithms. However, individual packets cannot be used to identify uncommon patterns during a long time. There are some attacks which contain normal packet headers when analyzed individually, while trend or repetition over time is anomalous, for example DoS attacks and probing.

7.1.2 Packet payloads

While packet header feature sets have been largely used and have their qualities, they also have limitations. When attackes are aimed at applications the attack bytes are inside the packet body and, thus packet header techniques cannot be used. This represents a big downside, especially since several of the exploits of the present time are directed not at network services, but at applications.

NIDS must use payload-based features extracted from packet bodies to detect such types of attacks, since the packet headers can appear completely normal. Payload analysis is more computationally expensive than header analysis because it requires deeper packet inspection It deals with a variety of payload types (pdf, jpg, HTML), transfer compression, and covering up methods. However, the upside of payload analysis is having access to all bytes transferred between network devices, allowing a rich set of payload-based features to be built for anomaly detection.

Since payload analysis has a high complexity, several methods focus on small subsets of the payload, such as the HTTP request or only the JavaScript of downloaded content. Anomaly-based methods do not try to match signatures of known malware, but they can apply heuristics such as pattern matching for the presence of shellcode.

7.2 N-grams and LibAnomaly

Techniques for packet payloads are less well defined, while approaches for deriving discriminating features from packet headers are well established. However, two common approaches seem promising: N-grams and libAnomaly.

N-grams can be used with automatic feature selection algorithms. Profiles are constructed for different types of files, calculating byte-frequency distributions. The NIDS then compares the analyzed traffic with the profiles using Manhattan distance. LibAnomaly is a public available library for data preprocessing models.

7.2.1 Application

N-gram have been used generally for analyzing requests to servers. It is useful on detecting anomalous patterns, such as shellcode within the structured application protocols, without requiring domain knowledge.

By using libAnomaly, techniques for detecting attacks against web applications focused on constructing a handul of models for normal user content transferred to the applications. Malicious requests are likely to be anomalous with compared to at least one of the models because they generally differ from normal requests in some way. Analyzing client content is a less researched field, from an anomaly-NIDS perspective. The client techniques intended to detect current web threats such as drive-by downloads, cross-site scripting and other malicious JavaScript.

8 Domain knowledge

Although NIDS is intended to be fully automatic, its creation and adaptation still require an expert. Even the N-gram method required domain knowledge to apply it to relevant parts of the network traffic. LibAnomaly also requires significant domain knowledge to select which fields within the network shoudl be modelled. On the other hand, detection using packet headers requires less domain knowledge.

Data mining approaches can be used to derive MCD header features for detecting some exploits. The candidate set of features can then be processed on the feature selection stage. These automated approaches ensure the most discriminative available features are chosen for detecting labeled attacks, instead of an isolated process.

Considering several common attacks are now payload-based, there is an increasing importance on approaches for analyzing payloads and constructing relevant features to detect malicious behavior.

9 Performance Evaluation

Receiver Operating Characteristics (ROC) graphs are used as a method to visualize the performance of classifiers and analyze the hit and false alarm rates [4]. A ROC graph is based on only two classes: positive and negative. In terms of intrusion detection, a connection may be considered harmless (positive) or harmful (negative). To evaluate the correctness of the classifier, two other labels are used: true and false, which gives us four possible outcomes:

1. True Positive (TP) - A harmless connenection is correctly classified
2. True Negative (TN) - A harmful connection is correctly classified
3. False Positive (FP) - A harmful connection is treated as harmless
4. False Negative (FN) - A harmless connection is considered harmful

When the true nature of the test cases is known, it is possible to gather the accuracy, the TP_{rate} and the FP_{rate}, which are given by:

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

$$TP_{rate} = \frac{TP}{P}$$

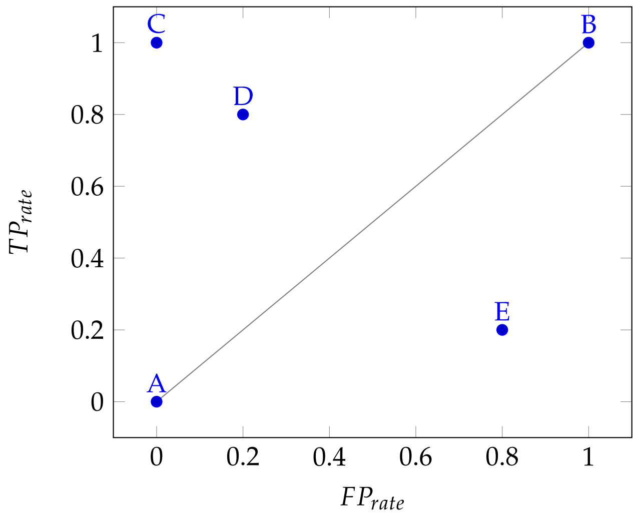
$$FP_{rate} = \frac{FP}{N}$$

These rates are used to plot the ROC space and they range between 0 and 1, inclusive. The TP_{rate} is used on the Y axis and the FP_{rate} is used on the X axis. Therefore, the ROC graph shows the relation between True Positive and False Positive. Figure 1 shows a simple ROC curve with five classifiers A, B, C, D and E.

Each classifier may output a single point in the ROC space. If the point is located on (0,0), such as classifier A on Figure 1, it means there was no test subject was considered harmless. Classifiers near the Y axis can be considered "conservative", as they need strong evidence to flag a subject as positive. On the other hand, B, positioned on the (1,1) point, represents a system that doesn't accuse any connection of misbehavior. Classifiers on the right side of the graph may be considered "liberals" for they make positive assumptions even with low evidence.

A perfect classification would result on the point (0,1), where C is, which means FP_{rate} was 0% and TP_{rate} was 100%, that is, every intrusive connection would be detected and no harmless connection would trigger the alarm. Thus, the higher the TP_{rate} is and the lower FP_{rate} is, the better is the classifier. If a classifier is located below the diagonal line, it means the performance is worse than randomly guessing. The lower right triangle is usually in blank because, on such cases where binary classifiers have a high miss rate, it is possible to invert the results. Therefore, E (0.8,0.2) is as effective as D (0.2,0.8).

Figure 1 - Five classifiers plotted on a ROC curve



10 Implementations and tests

Throughout the literature, two recent papers presented promising implementations regarding A-NIDS. One of them [6] presents an intelligent algorithm and the other [7] shows the performance gain when using a Field Programming Gate Array (FPGA). Furthermore, I have ran a series of simple tests to compare the performance of the proposed algorithms.

10.1 The Intelligent Algorithm

The simulation presented on [6] used the KDD Cup 99 dataset. It contains 32 continuous and 9 nominal features, along with normal traffic, probing, denial of service, user to root and remote to local attacks.

The algorithm works by first preprocessing the training and testing data and generating a random initial solution. The solution is updated at each iteration of the algorithm. While a treshold isn't satisfied, Support Vector Machine and Simulated Annealing are utilized to select the best featue set and then, Decision Tree and SA are used to increase testing accuracy and build the decision rules. At the end, the best accuracy, selected features and decision rules are announced.

10.1.1 Support Vector Machine

SVM works by mapping the training set into a higher-dimensional feature space for better discrimination. It is based on minimizing the structural risk. The input data is mapped by the a kernel function, which is usually linear, polynomial, sigmoid or radial basis. The latter is preferred on SVM due to its capability to deal with high dimensional data.

SA is applied to choose the best values for the parameters used on the Radial Basis Function.

10.1.2 Decision Tree

DT is a classification method based on a recursive greedy algorithm utilizing divide-and-conquer strategy. It creates a decision tree containing a root node, which is used to select the attribute with the maximum value of information; internal nodes, which correspond to the features; branches, which are the possible values of the features; and leaves, which contain the decision label for the given test case.

10.2 FPGA programming

SVM was also utilized on [7] to implement A-NIDS on a FPGA, taking advantage of its parallel processing potential. While the previous algorithm was aiming only on better accuracy, this paper focuses on the acceleration of the classification process.

Three different architectures are proposed.

1. *Heterogeneous Baseline Classifier*. The objective is the exploitation of precision requirements of the attributes. It tries to maximize parallelization by maintaining the ratio between digital processing (DSP) and logic blocks.

2. *Fit Cascade Chain*. Takes advantage of the custom-arithmetic potential offered by the FPGA to create a cascade with two different arithmetic precision. The first one is a low-precision classifier with small cost and high throughput. Only the data which couldn't be classified with certainty are fed to the second classifier, which is the high-precision one, thus with higher cost but less throughput requirements.

3. *Reconfigurable Cascade Chain*. It was designed for problems with large amounts of data, which could exceed the hardware's limitations. This architecture proposes a reconfiguration of the FPGA after the training stage and expands the possible design space.

10.3 Own tests

A series of tests was donducted in order to see in practice the performance of classification algorithms. The idea was to see the impact of the size of the training set related to the test set and to compare the results with the previously proposed algorithms. The classification method used was a pure SVM, which was ran through Weka.

10.3.1 Weka

Waikato Environment for Knowledge Analysis (Weka) is a workbench for machine learning algorithms. The software was developed at the University of Waikato and is licensed under GNU General Public License. The main features of this tool include the great amount of classification algorithms implemented, data preprocessing, attribute selection and data visualization. Weka works as an interface, which means new classification algorithms coded in java can be easily executed and have its results visualized [1].

10.3.2 The tests

For the purpose of this paper, however, only the default implementation of SVM was used. The initial part of the test consisted on acquiring and dividing the KDD Cup 99. Five different partitionings were made:

1. 10% for training and 90% for testing
2. 20% for training and 80% for testing
3. 30% for training and 70% for testing
4. 40% for training and 60% for testing
5. 50% for training and 50% for testing

Then, for each partitioning, the SVM was used to generate the model and classify the instances. The command used on Weka included a preprocessing using shrinking heuristics, weights of 1, seed of 3, error of 0.001 and the radial basis kernel function.

10.4 Results

The KDD Cup 99 dataset was divided in ten parts; while 9 were used as training cases, 1 was used as test input. Different approaches were used to compare the proposed algorithm with.

The proposed algorithm in [6] had 99.96% accuracy and used only 23 features. This represents a fortunate contrast to other results. The standalone SVM used with Weka achieved 99.64% only in a particular fold, which was in test number 5. Test number 1 resulted in 99.22% accuracy and took around 16 hours to process. Tests number 2, 3 and 4 correctly guessed 99.45%, 99.56% and 99.62% respectively.

As for the FPGA implementation, the speed gain was very interesting. Tests ran on MNIST Dataset had shown a speed-up of 7x when using the FPGA heterogeneous classifier, when compared to GPU works. Nonetheless, the reconfigurable cascade classifier achieved a 20x speed-up.

11 Conclusion

11.1 Comparisons of related work

Anomaly-based IDS is a promising field of study and there is a lot of research to be done. Machine learning on anomaly-based systems tries to automatically detect any kind of anomalous behavior. Analyzing the data of papers published between 2000 and 2007 (shown in [8]), it is possible to infer the following:

1. Most of the articles refer to the use of single classifiers, although hybrid ones seem to be taking place during the recent years. Only a few consider ensemble classifiers.
2. The most studied single classifiers on intrusion detection are SVM and KNN. As for hybrid classifiers, the integrated method is the most popular.
3. As there are very few public datasets for intrusion detection, the majority of studies have been made based on KDD Cup 99 and DARPA 1998 datasets.
4. Even though 30 out of 56 studies didn't use feature selection, it is certain that preprocessing the data improves accuracy on detection algorithms.

Regarding the data, the first techniques used to create models considering the whole payload. However, they restrict the context for each model by separating the traffic considering packet lengths and destination ports. Later techniques consider specific parts of the payloads. It seems more subtle anomalies can be detected at a lower false positive rate when using more specific context.

An important specific approach is the HTTP payload analysis. HTTP is used today to transport all kinds of traffic.

The proposed algorithm on [6] is an efficient approach for anomaly-based IDS, but it still needs to be tested with today's traffic.

12 Open Issues

Considering the data presented on the article, one can infer that some features are more suitable to detect certain attacks than others. An issue on this is the high computational cost to extract these features. Nonetheless, some allegedly real-time IDSs work possibly on batch detection, considering the time window to extract and process the information. Only a few suggested approaches had their performance analyzed, according to [2].

There is a lack of appropriate metrics, the cost to process high speed network packet flow, the analysis of ciphered data and the high rate of false positive.

All current approaches still need human help. Even the approaches with automated adaptation require a human to preprocess the input data. A possible workaround could be accomplished by using hybrid approaches, which combine well-established NIDSs with automatically generated traffic models.

Finally, another issue is the lack of public network traffic data. The KDD Cup 99 has been useful reasearching, but it is more than ten years old and certainly does not represent a real model of today's network traffic.

References

[1] R. Bouckaert, E. Frank, M. Hall, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. Weka—experiences with a java open-source project. *Journal of Machine Learning Research*, 11:2533–2541, 2010.

[2] C. A. Catania and C. G. Garino. Automatic network intrusion detection: Current techniques and open issues. *Computers and Electrical Engineering*, 38:1062–1072, 2012.

[3] J. J. Davis and A. J. Clark. Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, 30:353–375, 2011.

[4] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, 2004.

[5] P. García-Teodoro, J. Díaz-Verdejo, G. Marciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques and systems and challenges. *Computers & Security*, 28:18–28, 2009.

[6] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee. An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. *Applied Soft Computing*, 12:3285–3290, 2012.

[7] M. Papadonikolakis and C.-S. Bouganis. Novel cascade fpga accelerator for support vector machines classification. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1040–1052, 2012.

[8] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36:11994–12000, 2009.

Appendix A - Taxonomy

The taxonomy, regarding the different types of attacks, accepted in the paper is as follows:

- *Probing*: the intention of the attacker is to gather information about the system.
- *DoS*: Denial of Service is the attempt to prevent legitimate users from accessing the system.
- *U2R*: User to Root are attacks that try to gain root access to the system.
- *R2L*: Remote to local attacks intend to invade a local network from an external one.

As for NIDS specifications:

- *Detection method*: can be either *misuse-based* or *anomaly-based*.
- *Model acquisition*: relates to the traffic model, which can be human written or automatically generated.
- *Usage frequency*: can be either in *real-time* or by *batch*.
- *Architecture*: can be either *centralized* or *distributed*.
- *Prediction accuracy*: indicates how good a system detects intrusions.
- *Processing time*: measures how quickly it takes to a NIDS process an event.
- *Adaptability*: considers the ability to detect unknown attacks.
- *Resource consumption*: measures the memory and other resources usage.