1. Intelligent Techniques

Various classification techniques originated from different disciplines have been applied to build IDS capable to detect the intrusions effectively and efficiently. Examples of these techniques may cover statistical approaches and AI-based methods including decision-tree-search, production rule systems, data mining, genetic algorithms (GA), and other ML techniques (neural network, SVM, Bayesian network, etc.), pattern recognition, and so forth. Recent advances in the field of AI led many researchers to appreciate the opportunities it may give in application designs and apply AI-based techniques for IDS design successfully. Potential benefits of AI-based techniques over other conventional techniques may include achieving such qualities as:

1) flexibility of the parameter value's choice (versus the strict threshold definition of the devia- tions in conventional techniques),

2) adaptability (versus using restricted specified rules with conventional techniques),

3) advanced pattern recognition abilities (and a capability of new pattern detection),

4) fast computing (in many instances faster than conventional ones), and

5) learning capabilities.

Historically, IDS designers tried first the rule-based approach, which was similar to one used in firewalls.

The static knowledge-based IDS can demonstrate a high recognition ability for known attacks but fail in other attacks detection, especially new unknown attacks.

The learning ability that ML approach enables quickly becomes the major advantage as:

1) intelligent techniques have the capability of learning by example that helps to generalize from a representative set of samples and allows detecting new types of intrusions;

2) with learning by example approaches, attack signatures can be extracted automatically from labeled traffic data, thus allowing to overcome the subjectivity of human interpretation of intrusive behavior, with the latter being implemented in many current IDS; and

3) learning approaches are able to adapt to unknown or strongly modified threats.

Some of the latest IDS design approaches have employed the following methodologies and models:

• Statistical based that employed an operational model, Markov model, etc.

• Cognition based or knowledge based - expert systems, fuzzy logic, description scripts, etc.

• ML based - Bayesian networks, neural networks, GA, etc.

• Artificial immunology.

At the same time, an application of the intelligent methods in IDS design remains a hot topic of the current research with numerous publications reporting multiple challenges related to AI-based techniques and datasets.

The challenges related to the "best" technique's selection include:

1) No single-classification technique is strong enough to detect all classes of attacks with accept- able false alarm and missing attack rates.

2) Some of the existing techniques fall into local minima in training, as in finding the global minimum

these techniques become too computationally expensive to be employed in real life applications.

3) Existing techniques are not capable to model correct hypothesis space of problems in many cases.

4) Some existing techniques, such as neural networks, are unstable in nature, so they produce dif- ferent results with different initialization points due to the randomness embedded into the training procedure.

5) Different techniques trained on the same data not only may differ in their global performances but also may show strong local variations. Each technique may have its own region in the fea- ture space where it performs the best.

ML approach requires a sizable, good quality dataset to be used for model learning. Building up such a dataset represents a considerable problem in a security domain. Related evaluation dataset challenges include:

1) lack of big datasets that could be employed as a benchmark for IDS design and testing various techniques,

2) lack of the sufficient amount of quality training data, and

3) class imbalance that often occurs in available datasets, especially between the number of sam- ples representing certain attacks against the normal behavior examples, may cause the results of classifiers to get biased toward the majority classes.

The subsections below provide a few examples and use cases that demonstrate how IDS are build utilizing various intelligent techniques.

1. IDS Design based on K-Means Algorithm

k-Means is a type of a vector quantization algorithm that employs ML approach to classify points

of data (observations) into k separate groups. It works by progressively trying to find k number of

stable groups in the dimensional space d. This problem solving requires very extensive computa-

tion, as it represents the problem class, that in computer science theory is called NP-hard. However,

heuristic algorithms exist that in many cases do solve the problem but do not provide a guarantee

to achieve the global optimum point. The classification problem formal specification: Given a set of data points (X1, X2, ..., X₁), where each point represents a d-dimensional real vector, k-means clustering aims to classify the n points into k (≤n) sets S = {S1, S2, Sk}, so as to minimize the within-cluster sum of squares (i.e. vari- ance).

In relation to an IDS design, one typically considers a data point consisting of data embedded into a traffic packet. For a packet, the applicable data may include not only the traffic and technol- ogy information located in the packet header but also the payload. This data can be treated as a vector that is analyzed with the k-means algorithm in order to classify all data packets into a cer- tain number of groups. In an anomaly detection, one group commonly represents a normal traffic while another one an anomaly. Or in a misuse detection, one may set up a few with each one com- posed from a certain type of anomaly representing a specific attack.

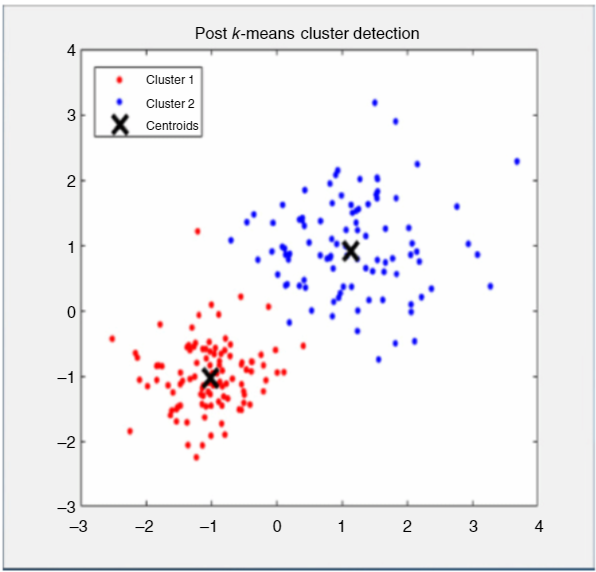
k-Means can be used for both anomaly and misuse detection with several major shared benefits. The first is that analyzing a new point is faster than with many other learning systems, since the time is related to the number of clusters and the number of dimensions in the system. This means that combining k-means with other approaches like k-means++ to minimize the number of clus- ters in the dataset can make the system exceptionally efficient. The same increase in speed can be achieved with a reduction in the number of dimensions used in the system. One example of this is the LKM algorithm, which performs a linear discriminant analysis (LDA) on the data before k-means is run. It reduces the data to two dimensions that allow both the k-means algorithm as well as the analysis of new packets to be performed much faster. Another major approach advan- tage is that the algorithm is unsupervised that means that it can merely be given data, which might not need to be labeled before its application.

One problem is that, like many ML-based detection models, k-means may produce a high rate of false positives compared to traditional rule-based IDS implementation models. Another problem is related to finding an appropriate cluster count to create the IDS network behavior model. Too few clusters and the model will have distinctly different groups clustered together that increases the rate of false classifications in the areas between them as well as produces centers that have low densities of points in their close proximity. Too many centers create clusters with no data points in them, which serve no purpose to normal traffic, and cause the classification system to slow down. An algorithm that builds off of k-means, called Y-means, helps to address this problem by avoiding making empty clusters. It attempts to open a reasonable number of clusters without their oversup- ply. The other main disadvantage of k-means is that the system is unable to adapt to shifting net- work traffic without retraining. If a new type of traffic is attempted, the k-means algorithm must be retrained and retested in order to learn how to properly identify it.

No matter which IDS type is employed, the k-means approach has to create a valid set of clusters, which is done as follows. First, one has to collect a normal dataset. Note that these sample packets contain only two vectors per point, or only two pieces of data. These might be taken directly from only some fields of the packets or perhaps some abstraction or conversion in order to reduce the dimension of the set. Regardless of the acquisition method, one ends up with a dataset that defines the normal network traffic model. In a real dataset, there would be many different vectors to each data point, since packets contain a lot of data even when excluding the payload. Now, one runs the k-means algorithm in order to build up a model of the standard network traffic in the monitored system. In the case of anomaly detection, we assume to build two clusters - see Figure 3.10 for this case illustration.

We need to determine how closely members of each cluster are positioned to the centroid loca- tion, that way we can find the boundaries of our two clusters. The boundaries give us an ability to distinguish if a point belongs to the cluster, or if the new data point is an outlier that does not belong to any of them.

Attack detection with k-means is quite straightforward. After the analysis above is done and the model is built up, the model can be then fed with new data representing the traffic characteristics. The IDS model will compare each data record against all known centers and determine to which one this record belongs based on the shortest distance. Then the algorithm would check to see if it is within the boundaries for that center and classify this record accordingly. If the record is located too far from each centroid, it is considered as an outlier. The same approach of using k-means to classify traffic can be applied to misuse detection. The system must go through the same training procedure as before, now though it is given a set of packets containing known attacks it is desig- nated to defend against. It can work much like the previous example, though now passing traffic data points that do not match properly to the model several centroids, it is given, are rejected as outliers. One important thing to note is that since the data is predefined and thus each cluster must be distinct, each attack should have one or more cluster that covers it, or if fewer centroids exist several different types of attacks might have to be classified together if they are too similar or can- not be fully separated.



1. IDS Design based on K-Nearest Neighbour Algorithm

The k-nearest neighbor algorithm, commonly shortened to kNN, is widely employed for intrusion detection, for both cases of a misuse detection and an anomaly detection. The core idea of kNN method is to classify unknown data points based on the number of known data points in their own proximity. The algorithm has to be given both a set of points that have a known classification, as well as a number k (k is a natural number, typically rather small), and sometimes, the distance d in the features space. In this method, new data, extracted from a coming traffic, is considered as a point in multidimensional features space and then is compared against labeled points in the fea- tures space and the nearest k points are found. Then, in some modifications but not always, the method ignores points not located within the distance d of the new point, thus limiting the distance to known data as well as requiring to have enough points representing a certain traffic type nearby to classify this record as the same attack type. The classification can also be done by just using a conventional kNN algorithm (without using the d distance), but the system would need more data. The algorithm modification described here, however, does apply the distance d to help narrow down the valid neighbors. It can then be tested by comparing more known points from a validation set, which must have been previously unused and ideally selected randomly from the initial dataset, to verify that k and d are chosen accurately to classify the validation data. Note that because the algorithm requires pre-classified data, it is considered to be a supervised learning algorithm.

For example, they could be obtained either by directly interpreting the content of the traffic packet or by decomposing the packet header information into vectors that describe the nature of the packet. An example of the latter would be the IP address of the sender of the packets.

The KNN method is widely employed in IDS design. The two types of IDS systems that we are discussing here will be misuse and anomaly detection systems. Both of these systems make use of the classification ability of this algorithm, where new packets are compared to known packets to determine which category they belong to. The method explores the idea that an unknown object that needs to be classified is similar in nature to the known class objects to which its data points are the closest to in the specified feature space. The exact specifics of each of the two main uses might differ, but there are some common characteris- tics in terms of an output based on the k and d parameters selected. The first, k, is commonly known when discussing kNN, which is the number of neighbors to consider for the final clas- sification, represented by the number of k in this method. The second parameter, d, however, is introduced in this method and is used to help define the boundary line for the data to be considered.

The main idea is that a smaller d tends to mean that points must be closer together, and thus must be more similar. This allows the IDS to be able to classify points that otherwise might be misclassified by being near a cluster of one type, when they are even closer to fewer number of points of a different type. In Figure 3.11 one can see that both parameters are commonly connected to each other. This shows the need for having a varying distance d, especially along the boundaries between benign and malicious traffic. If for instance the traffic above is of a benign point, but the nearest neighbor count is five, then a constricting distance is required, and can help to better define the boundaries between benign and malicious data. This comes as a trade-off between a lower rate of false positives and an ability to consistently recognize a modified variation of a known attack if used in misuse detection.

Let's now explore the usability that KNN may have in misuse detection. For this application, the method application is straightforward. The method will be given a set of pre-classified points that it will use to compare to future data. The system's dataset must have not only known attacks but also a number of examples of benign traffic. Should the system not being using a distance d, then more samples of each dataset must be included so as to better define where a boundary between malicious or benign data goes. The method also requires parameters k and d to properly classify new traffic. If effective k and d values are not known, one can be found by the method above where

a testing set is fed into the system to find out at what values of k and d the system produces the

fewest false results. After the IDS results are determined to be satisfactory by the expert setting up

the system, it can then be used to directly classify traffic. The system itself now knows not only

traffic that is safe but also various attacks it is supposed to defend against. The algorithm can check

new packets against the known set and try to identify them. This is done by finding the kNNs

within the distance d from the data point and classifying the point based on what classification of

the neighbors was most common. This distance-restricted neighbors' approach, however, does

have the downside that there can be a tie. In this case, the method would err on the side of caution,

declaring a point as an attack, and in the event of a tie between different attack types, it would

report an attack of an ambiguous type, stating that it is one of the tied attacks, though the method

cannot determine which one. The KNN method can also be used for anomaly detection, also utilizing the classification feature of KNN, though with a different approach than the one used for misuse detection. The method exploits an assumption that malicious data is intrinsically different from normal data, and there- fore has a measurable distance from the data points representing a normal traffic. This can be further refined by requiring some number of known points to be within the d distance from the unknown point, making the ultimate approach similar to the previous system in terms of imple- mentation. The anomaly detection would require a similar but a distinct model from the misuse detection. For instance, the data that would serve as the dataset for this IDS would be collected from the network itself. Based on how clustered the data is, the distance d is determined. It should be chosen carefully, as it directly determines the ability for the system to both avoid false positives as well as to reduce false negatives.

Although KNN can allow for building a strong classification system for new data, the method has two major downsides. The first is that the algorithm itself is unable to run without a pre-classified dataset. The labeled dataset is necessary to effectively run the method but obtaining it could be costly in terms of resources and expertise needed since the datasets might vary from system to system. The second problem is that the algorithm execution can be very slow, especially with a larger dataset, since every point in the dataset is analyzed. This means that the execution time increases depending on the number of data points in the training data.

A diagram of a diagram of a diagram

Description automatically generated

1. IDS Design based on Genetic Algorithm

Genetic algorithms (GA) represent a type of a learning system that mimics the evolution of real life, where the fit can survive and reproduce, and over time become better suited for the task. This method has been employed in design of both anomaly detection and misuse detection. The IDS design steps follow up the same overall concept, although the fine details of each step may differ as will be described below. Note that the GA include not one specific method but rather the family of similar approaches, which have the common similarity of mimicking the process of natural selection and evolution. However, the family members might demonstrate a significant deviation from each other. In this section, we will describe major features that might vary in a particular implementation. GA have a few major components, which are used to describe them, namely, the search space, the population, the fitness function, the reproduction or recombination algorithm, and the mutation algorithm.

The search space is the area that all members of the population are in and represents all possible combinations of feature values. It is assumed that somewhere in this space exists at least one opti- mal solution to the problem or the solution that scores the highest possible score on the fitness function. The population consists of the current members that are to be rated by the fitness func- tion. The GA execution consists of numerous rounds, which are similar to each other on their functions and code. In each round the number of possible solutions, that is called a current genera- tion, are examined and new solutions, representing a new generation, are generated based on the previous ones. Typically, the bottom part of these members will be eliminated each generation round as more combinations of the higher scoring algorithms are added. This causes an overall trend of creating a new generation, where the surviving portion is at least as good as the surviving portion of the previous round. The population is always a subset of the search space, in which each member will be tested and either kept or eliminated in the upcoming round. Testing includes the score calculation based on the fitness function, which is applied to all generation members near the end of each cycle.

The first step of each iteration is the recombination, or repopulation, step. The recombination algorithm is the main component of the repopulation phase of the GA. During this phase the algo- rithm chooses the members with good fitness function values (see Figure 3.12) and combines them together to form new members that can then be added to the new generation. Typically, the fitness function is deterministic that allows the mixing of traits to hopefully produce a better population. However, as some kind of randomness is involved in the new generation members choice, an improvement should work as a trend. The new generation choice algorithm may vary. Commonly, it combines a random choice with some weighting associated with how often each member is cho- sen for reproduction. Weighting is based on the fitness of the members, with a favoring for the higher scoring members.

The final step of the repopulation cycle involves adding small mutations to random members and adding those into the population as well. These mutations are typically both random and minor. This process can either mutate a member permanently or be a copy of the member with the mutation added into the population independently. Mutation is introduced to add more random- ness into the selection process in order to prevent the method getting stuck into a local maximum and not being able to find the global maximum and running an endless cycle that is unable to produce a good enough member, called the "wolf trap phenomenon" (Sivanandam and Deepa 2007).

A diagram of a process

Description automatically generated

After the repopulation is complete, the algorithm moves on to the fitness evaluation stage to remove the lower performing members of the generation as determined by the fitness function.

The fitness function is one of the most vital parts of the algorithm, since its scoring not only determines if a suitable candidate has been found and thus the algorithm can cease but also defines the method goal. Also, it determines the choice of the new generation members in the repopula- tion process serving as a life-or-death executor. In a GA-based IDS, this function would be one that tests the generated algorithm with a supplied set of test packets. Note that this function can be composed from multiple criteria that the method must attain, for example, both missing attacks and false alarm rates. Although the GA general idea can be used for a misuse-based IDS design, the specific details of

each system must be more closely analyzed and defined to make a successful IDS system as many GA variations have been proposed and tried. In order to be acceptable, a member must be above the threshold level, which corresponds to having minimum allowed missing attacks and false alarm rates. For example, the minimum detection rate could be set as 90% (that corresponds to the missing attack rate less than 10%) and the true-positive rate (which here is defined as the inverse of the false alarm rate) could be defined as 85%. If none of the current generation member reaches those thresholds, the generation is culled so that only the most successful members survive to the next round.

For the repopulation stage, the method would be fairly straightforward. Each member would be chosen randomly for each reproduction iteration, usually with weights preferring those that scored higher in the previous round. Any particular member can be chosen more than once, but the same pair should not be chosen more than once. This means that having two copies of the same member is avoided, since duplicates do not contribute to the search for an acceptable answer.

There are several major advantages to this approach. This method can be very effective in detect- ing attacks that follow the same scenarios with the same signature features that the method was trained to detect. Another advantage is that the method can be trained by an administrative per- sonnel within the organization that the algorithm is being implemented for. This means that one does not have to wait for a signature to be published, and thus new threats can be responded to more quickly, provided that a comprehensive dataset can be acquired. Also, the method execution can stop at any point when the time or resources are up.

Some drawbacks do limit this approach. The method efficiency highly depends on an initial generation as it can fail to generate members that meet the requirements set at the method launch. This can be mitigated by an increased mutation rate or the maximum number of iterations allowed in the algorithm. The other major problem is that as the ML approach, the GA method relies on having a comprehensive training dataset, and therefore is vulnerable to overfitting to the dataset. Since the algorithm only optimizes for the given training set, a training set that is biased toward a subset of some attacks or even most commonly to samples representing normal traffic can fail to detect other common variations of the attack that were omitted. The final problem is that the GA can take quite a while to train in order to achieve its goals. This time scales both with the complex- ity of the fitness function that the GA is optimizing as well as linearly to the size of the population and the number of iterations to be performed. This means that the system itself could take a lot of resources if a large population and iteration count are used to reach the best possible solution or to get very close to it.

When creating an anomaly based IDS, a similar approach for the GA can be employed. It will follow very similarly to the algorithm that was set up before, but now the only classification result is if the packet represents a normal traffic or not. Consider the previous method but now instead of trying to positively identify an attack, it will attempt to positively identify a normal traffic. Assumptions about the dataset include that the normal traffic packets have some common but distinct patterns, or at least they do not share the same pattern with anomalous traffic. The method itself could be set up very similarly to the misuse detection. The initial generation population, repopulation, and mutation methods are set up the same way, but the fitness function should be formulated specific to anomaly detection.

The anomaly detection approach described above has a couple of challenges to overcome. The first is that the system as proposed has no ability of learning new activity and thus cannot adjust to the changing needs of the network for anomaly detection. The second problem is that training the system likely will take even longer than misuse detection to train, since the fitness algorithm would generally have a much larger set of data than a misuse model.

1. ANN Structures and their choice for intrusion detection
   1. Shallow ANN Topologies and their Ensembles

Artificial neural networks (ANN) in combination with other AI techniques have become one of the major methodologies employed in IDS design and implementation. This section describes various ANN topologies and compares the ways and methods of improving the ANN-based IDS performance along with the resource consumption. With the goal to design effective and effi- cient IDS, it discusses the choice of the ANN architecture and its parameters, the choice of an ANN fully connected topology versus a partial connectivity, and the IDS design in a form of a hierarchical system of heterogeneous ANN-based agents. While a wide set of parameters are investigated, the particular attention is paid to the choice of ones responsible for the resource consumption such as the training set size and the training time in an ANN application, the number of generations in GA, and the number of agents in a multiagent systems. In this section we want to demonstrate a variety of possible approaches to the ANN-based IDS design and to encourage further investigation. We believe this material could be used for the development of a number of projects.

Although an IDS technology started off by building an expert system for anomaly detection, the ability to learn and later on to evolve becomes a prevailing requirement in further development. This condition of learning and self-modification makes the AI techniques, and ANN in particular, one of the most promising research directions on the intrusion detection arena. Over the years, the design advancement has gone from rule-based systems to statistical analyzers based on the deci- sion tree algorithms and to more complicated data mining techniques and ANN structures. Within the ANN category, an IDS research has evolved from homogenous structures implementing either supervised or unsupervised learning toward complex hierarchical heterogeneous systems merging together different AI models and techniques in order to achieve the strong ability of adaptation through self-evaluation and evolution.

The choice of the ANN topology is crucial for improving an IDS performance. Due to a limited resource availability, a fixed structure of overall connectivity between neurons may not provide the optimal performance within a given training period. A small network may not perform effectively because of its limited information processing power. A large network, on the other hand, may have some of its connections redundant. It is also known that an ANN with an excessive number of neu- rons may provide a more accurate detection ability but may produce worse generalization for unknown attacks due to overfitting. Recently, various methods were proposed to solve the problem of learning the network structure and the connecting weights, where in general, the network structure is consid- ered to be a fully connected network as the number of the hidden nodes is fixed. In our second use case (Section 3.6.5.4), we discuss a method of IDS design based on an application of neural network, which is trained initially and optimized later with GA application. This method attempts to minimize both the training and execution time along with maximizing the performance.

* 1. Experimental Setup and Datasets

Numerous experiments have been performed by many research teams to study AI technique appli- cations in an intrusion detection. Many researchers used Defense Advanced Research Project Agency (DARPA) and Knowledge and Data Mining (KDD) Cups benchmark datasets for an IDS design and its performance evaluation. In the following projects, we started with the KDD dataset

The content of this subsection is based on the following papers: Novikov et al. (2006a,b) as an initial point also. An initial dataset has been preprocessed for later employment in ANN training and testing. This preprocessing, aimed at generating the training set with the specified selected features, allows for a significant dimension reduction and a consequent decrease in the training time requirements. It lets produce various training sets and procedures for different agents. This feature has a particular importance in our heterogeneous system of different agents with various functionalities.

* 1. IDS Design with multiple Intelligent Heterogenous Agents

This section describes a novel intrusion detection and classification model, which was inspired by a biological and social organization of human operation. This model is simulating both the way how the brain neocortex and its cortical columns compose a hierarchical distributed system and the way an organization provides the same hierarchy through supervision and distribution of work among various entities. To synergistically merge these two aspects, a multi-agent architecture was designed that created a set of individual agents who cooperated and collaborated with each other to autonomously perform classification on arbitrary data using a hierarchical paradigm. In this application, the data comes from the network and/or system monitoring. The model is also based on the analysis of different methods for performing mutual learning through a cooperation of dis- tributed agents, which ranged from simple methods aiming at proving the concept to complex methods of further extending the process into new realms reinforcing the same conclusion. In order to implement a design that allows for an arbitrary application of various iterative classifica- tion algorithms, a set of six heterogeneous agents was created comprising over 47000 lines of code. These agents are segmented from the group by a way of responsibilities and physical connections. The realization discourse was provided by a hybrid peer-to-peer agent framework implemented in JADE, a well-tested Java agent platform, whose design follows the specifications provided by the IEEE FIPA committee for agents and multi-agent systems.

Developed agents are classified into six groups, each of which are assigned the following responsibilities:

• Data preprocessing and hosting (the data sources)

• Supervision and aggregation (the supervisors)

Classification and training (the employees)

• Result proxying (the collectors)

• Result display and validation (the result receivers).

• Monitoring of other agents (the monitors).

Each of these groups is designed as a separate agent that will use the capabilities provided by JADE along with configuration and custom protocols in order to perform all its responsibilities with no manual control required. As this complete autonomy is hard to debug and validate in order to make the progress viewable, each agent is designed with a graphical user interface (GUI) for visualization of its currently active processes, data, message input/output, and network usage. Each agent will perform its own responsibilities while cooperating with others. The inputs of the architecture originate from the data source, whose functions include reading the compressed data- sets into memory and converting those datasets into numerical form for usage in classification and training by the supervisor agents. The supervisor agents then use these data to engage supervisory activities involving classification and result analysis through aggregation and testing to determine employee capabilities and long-term performance by comparing against the previously achieved results. The employees on the other hand perform the actual classifications, which in this imple- mentation are realized by arbitrarily designed neural networks, and respond to the supervisors.

3 Intrusion Detection Systems

Supervisory activities consist of performing retraining and retesting as needed. The outputs of the architecture originate from the aggregation process, the supervisor performs, and get transmitted to a set of proxy agents, the supervisor has subscribed to. These proxy agents then relay the trans- mitted results to their subscribed result receivers, where the results are automatically validated and then displayed in graph, chart, table, and textual forms.

The experiments, which were performed, gauge the architectures capabilities with rela- tion to classification accuracy, adaptability, and fault tolerance. For all three experiments, the same set of configuration parameters and ANN models were used, thereby avoiding introducing further parameters to the experimental processes. The first experiment applied the datasets and attempted to determine how the classification error rate and false-nega- tive/false-positives rates would decrease or increase as the number of employees grows up. The tested hypothesis was that as the employee's number was increased the misclassifica- tion rate would decrease as there would be more individuals performing classification and the likelihood that a majority of employees would misclassify would therefore decrease. The hypothesis was verified as the classification error rates went down as the employee amount was decreased, as can be seen in Figure 3.20. One can observe a significant improve- ment in IDS detection and recognition rates while replacing one classifying agent with two; however, with the further increase of a number of employees, no significant difference in overall performance is recorded.

For the second experiment, we split the dataset into k disjoint sets where each k set contained a randomly chosen amount of the target classes (various attacks), which the originated dataset maintained. This allows for introduction of those k datasets at varying periods. The hypothesis was that the introduction of the previously unlearned target classes, constituting novel attacks, would result in retraining and therefore adaptation to a set of data that was not analyzed previously. The results, which can be seen in Figure 3.21, validate this hypothesis.

Multi-agent IDS composed from cooperating and mutually learning heterogeneous ANN struc- tures proved to be a livable solution in IDS design. Their application results in improving the over- all adaptability and flexibility. However, the most significant performance gain so far has been achieved with a replacement of a single ANN structure with a double one, while further increase in the number of agents may not produce an additional considerable increase. One explanation of this phenomenon could be that the performance has already been very high and near perfect. Although one has to remember that while agents are able to teach one another to recognize newly introduced attacks, learning requires some time and resources. During this time the recognition performance may drop down substantially.