Ant Colony Induced Decision Trees for Intrusion Detection

Frans Hendrik Botes[1], Louise Leenen[1, 2] and Retha De La Harpe[1]
[1]Cape Peninsula University of Technology, South Africa
[2]Council for Scientific and Industrial Research, South Africa
fbotes2@gmail.com
delaharper@cput.ac.za
lleenen@csir.co.za

Abstract: In the ashes of Moore's Law, companies have to acclimatise to the vast increase of data flowing through their networks. Reports on information breaches and hackers claiming ransom for company data are rampant. We live in a world where data requirements have become dynamic, where things are constantly changing. The field of intrusion detection however have not changed much, traditional detection methods are still the norm for commercial products promoting a rigid, manual and static detection platform. Intrusion Detection Systems (IDS) analyse network traffic to identify suspicious patterns with the intention to compromise the system. Practitioners train classifiers to classify the data within different categories e.g. malicious or normal network traffic. Machine learning has great potential when applied in the intrusion detection domain: decision trees (DT), random forests (RF) and ant colony optimization (ACO) are all popular research topics. This paper focuses on the recent advances within machine learning, specifically the Ant Tree Miner (ATM) classifier. The ATM classifier proposed by Otero, Freitas & Johnson (2012) builds decision trees using ant colony optimization instead of traditional C4.5 or CART techniques. Our experimental process ensures reliability, comparability and reproducibility, which are lacking in some previous research within the field. This approach is intended to improve on previous studies combining both domains. The ATM classifier has not been tested in the intrusion detection domain.

Keywords: ant tree miner, ant colony optimization, decision trees, intrusion detection, swarm intelligence

1. Introduction

Kemmerer and Vigna (2002) defined intrusion detection (ID) as the process of distinguishing between malicious or unauthorized network activities. As mischievous and malevolent activities become more advanced and adaptable, intrusion detection techniques are required to perform more intelligently in order to overcome avant-garde attacks (Kumar, 2007; Hoque et al., 2012; Aghdam & Kabiri, 2016). Computer environments are still being penetrated and the amount of data harboured and transported throughout company networks has vastly increased in recent years. The data boom has left companies with significant vulnerabilities waiting to be exploited.

Bilge and Dumitras (2012) performed an empirical study on zero-days attacks, and found that typical zero-day attacks last 312 days on average. Once the vulnerability is disclosed publicly, the number of exploits increases by a factor of five. In addition, as Burdette (2016) noted, most successful corporate attacks are only discovered on average after 314 days. Frei (2014) from FireEye also supports this alarming statement.

The real crux within the cybersecurity field is detection. It is difficult to prevent something you are unaware of, and without detection, no controls will be able to properly satisfy the underlying risk. The detection problem lead to the ongoing research in intrusion detection using machine learning techniques such as decision trees, random forests and ant colony optimization as classification techniques to classify intrusion data.

Despite vast amounts of research in intelligent ID domains, few studies have found an effective solution (Sommer & Paxson, 2010; Mohammad, Sulaiman & Khalaf, 2011; Lee et al., 2002). Irreproducible results and unreliable research techniques are to blame (Section 4). However, it has been shown that it is possible to create intelligent ID components. Decision trees and random forests have proved to be among the best classifiers to use for intrusion detection (Albayati & Issac, 2015). Ant colony optimization, although unconventional within intrusion detection, does have significant value when used to solve optimization problems or even to classify data (López-Ibánez,Stützle & Dorigo, 2015).

Section 2 covers background information. In Section 3 contains an introduction to the Ant Tree Miner Classifier, and a description of the experimental setup is given in Section 4. Results are discussed in Section 5, and the conclusion and a discussion of future research are given in Section 6.

2. Background information

In this section, we discuss intrusion detection systems and the different detection methodologies used in order to detect malicious data. The benefits and implementation of machine learning techniques follow as we look at popular techniques such as decision trees and ant colony optimization in the intrusion domain.

2.1 Intrusion detection systems

Different types of intrusion detection systems (IDS) can be identified based on how the systems operate and gather information. In order to sift through all the data and identify malicious activities, a detection component is required. An IDS makes use of several methodologies in order to detect the malicious data or anomalies, and the most used detection methodologies are signature-based, anomaly-based detection and newer hybrid methods (Scarfone & Mell, 2007; Modi et al., 2013).

2.2 Machine learning with intrusion detection

Data mining, combined with machine learning is defined as processing data to gain the implied, prior unknown potential of useful information (Parihar and Tiwari, 2016). The benefit of machine learning in ID is the ability to withdraw the required and unknown information and regularities from massive network data. If machine learning techniques can detect patterns in a set of data, it should also be able to detect future intrusions in the same data. Classifiers are algorithms which receive labelled data from training datasets, align it to predefined groups defined by their specific qualities and then output a classifier that can predict the correct class to which a new item belongs. Classification of intrusion data is a mature research area and several classifiers have had success in detecting intrusions (Nguyen & Choi, 2008; Tsai, Hsu, Lin et al., 2009). The machine learning domain is evolving more rapidly than that of ID, and this creates an urgent need to find improved techniques with the ability to classify intrusion data. Decision Trees are regarded to be the most reliable and accurate method implemented within intrusion data (Albayati & Issac, 2015; Tavallaee et al., 2010). Decision trees are tree-like graphs that consist of internal nodes that represents a test of an attribute, branches that denote the outcome of such tests, and leaf nodes that outline the label. The path followed from the root node to the leaf indicates the rules for classification. The main advantage of DTs over other classification algorithms is that they provide a rich set of rules that are easy to understand and can be integrated with real-time technologies. Singh and Nene (2013) noted that although DTs are very accurate, they are computationally intensive. Machine learning ensemble methods are used to obtain better predictive performance that would have been impossible with any constituent learning method e.g. C4.5, tree learners, decision tree learners and Bayesian methods (Dietterich, 2000). Figure 1 illustrates a simple decision tree that classifies intrusion data. The protocols TCP and UDP represent branches of the root node 1. The two classifications, serror_rate and logged_in, are internal nodes. TCP protocol with a serror_rate value over 3, will trigger an alarm.

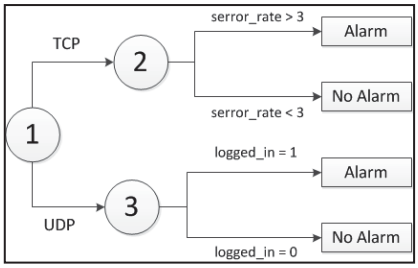

Figure 1: Decision tree intrusion classification example

Rai et al., (2016) proposed the creation of a DT algorithm in which they use their own split algorithm for implementation within ID. Tesfahun and Bhaskari (2013) used random forests as well as their own Synthetic Minority Oversampling Technique. Zhang and Zulkernine (2005) applied RF for ID and improved performance by building a balanced dataset. Tavallaee et al., (2009) applied DT, RF and several of the most popular classifiers in their NSL-KDD dataset, providing a baseline result.

Swarm intelligence attempts to replicate the nature of swarms and colonies. Ants they leave a chemical substance, pheromone, to mark a specific route. Pheromone evaporates slowly over time. The strength of the pheromone level will evaporate faster on a longer path because it takes longer to traverse. Thus, shorter paths are chosen more often, and build up higher pheromone levels than longer paths. Based on this knowledge, ACO algorithms are known for solving optimization problems and clustering data and have been particular successful

when applied in business, engineering and science. The algorithm by Dorigo and Gambardella (1997) introduced a new way of classifying data using the nature of ant colonies. Ramos and Abraham (2005) introduced the ANTIDS, an ant colony based clustering technique to detect intrusions. Tsang and Kwong (2006) improved the ant mining cluster algorithm and applied it to the KDD99 Cup dataset for intrusion detection. Aghdam and Kabiri (2016) created a feature selection process using ACO in ID exhibiting very low computational complexity when using a simplified feature set. A recent advance, combining ACO and DT, by Otero, Freitas & Johnson (2012) introduced a new way to build DTs. Their initial experiments with other machine learning datasets showed improvement from the traditional techniques used to build DTs.

3. The ant tree miner classifier

The unknown spectrum of inducing DT with ACO algorithms has been a new research topic since 2012 (Otero, Freitas & Johnson, 2012). The *divide-and-conquer* principle is used to induce a DT: it involves iterating from top-down, selecting the best attribute to label an internal node of the tree. However, this process becomes moot as the appropriate attribute has to be selected based on heuristic evaluation. The most popular heuristics for selecting the attributes in decision trees are CART, ID3 and the well-known C4.5 algorithm (Breiman et al., 1984; Quinlan, 1986 and 1993). Otero and et al. (2012) proposed a new method to induce DTs the method follows the traditional structure of ACO. Figure 2 shows a high-level overview of the ATM algorithm in pseudo-code.

```
Input: training samples, list of predictor features
Output: best tree
1. InitialisePheromones();
2. ComputeHeuristicInformation();
3. treeGB ← empty set;
4. m ← 0;
5. while m < maximum iterations and not CheckConvergence() do
6.         treeIB ← empty set;
7.         for n ← 1 to colony size do
8.                 treeN ← CreateTree(Examples, Attributes, − );
9.                 Prune(treeN);
10.                if Q(treeN) > Q(treeIB) then
11.                        treeIB ← treeN;
12.                end if
13.        end for
14.        UpdatePheromones(treeIB);
15.        if Q(treeIB) > Q(treeGB) then
16.                treeGB ← treeIB;
17.        end if
18.        m ← m + 1;
19. end while
20. return treeGB ;
```

Figure 2: ATM pseudo code adopted from Otero and et al. (2012)

The ATM Classifier differs significantly from the Ant-Miner algorithm by building DTs instead of rules. The ATM method of inducing DTs has not previously been implemented within ID. Section 4 outlines how DT based algorithms are among the best to use within ID. Otero and et al. (2012) experimented with the new ATM classifier and the results concluded improved accuracy, performance and statistically significant differences from other classifiers. However, the question still begs: How will the ATM Classifier perform when implemented within a new domain with much more complex data?

4. Implementation and testing

Research and experimentation within the combined intrusion detection and machine learning fields have received a lot of criticism (Tavallaee et al., 2010; Sommer & Paxson, 2010). Poorly designed experiments have been to blame for discrepancies in success between research areas and practical implementations. Pitfalls are summarized below (Tavallaee et al., 2010; Sommer & Paxson, 2010):

- No clear indication of datasets used for training and test purposes
- Flawed datasets such as KDD99 used for machine learning
- No clear indication of parameters used, motivation for their parameter selection nor number of simulation run during experiments
- Lack of evaluating classifiers based on attack type

- Poor evaluation techniques - no consideration of performance or cost overheads

The experiment cycle (Figure 3) used to test the ATM classifier in ID has been designed in order to overcome the limitations and flaws in previous studies targeting the combined domains.

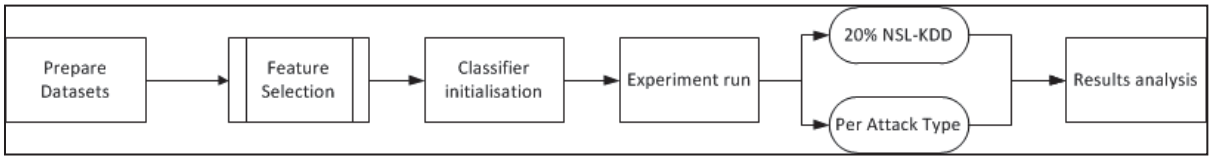

Figure 3: Experimental process

4.1 Datasets

The NSL-KDD dataset (Tavallaee et al., 2009) improves upon the popular KDD99 dataset. Due to the significant number of improvements, the NSL-KDD dataset was selected for training and testing the ATM classifier. It is important to classify each segment of the dataset used for specific purposes:

- Training dataset – Based on the NSL-KDD 20% training dataset, used to train the ATM classifier.
- Validation dataset – Based on the NSL-KDD 20% training dataset split into two parts, 66% training only and 34% validation (unseen). The dataset was used to validate the parameters for the classifier.
- Test dataset – NSL-KDD Test 21 dataset was used for test purposes as it features unknown attacks and therefore increases realism. Unlike other test datasets the Test21 specifically excludes easy detectable attacks and therefore bounds the performance to 65% (Tavallaee et al, 2009).

The datasets were also pre-processed by using feature-coding; categorical feature encoding was used to change the categories to numeric values, and the nominal fields will then be represented in numeric categories instead of text. Nominal fields represent certain classes or categories e.g. TCP, ICMP or UDP, hostnames, http, or echo etc. After the feature-coding process, the dataset had the features as displayed in Table 1. The datasets used in this research are available at http://github.com/FransHBotes/NSLKDD-Dataset.

The NSL-KDD dataset contains the following attack classes:

- In Probe attacks the intruder scan the system for vulnerabilities scoping the network, hardware and software in order to identify potential vulnerabilities to be exploited.
- User to Root (U2R) attacker tries to gain administrative (root) access to the system by exploiting vulnerabilities.
- Remote to Local (R2L) attacker tries to obtain local access across a network connection.
- Denial of Service (DOS) attacks tries to interrupt and cripple services on a host.

The classification task was only binary: normal or intrusive. Each attack class was split to evaluate the classifier per attack class and build an ensemble model.

Table 1: Features within NSL-KDD dataset

#	Description	#	Description	#	Description	#	Description
1	Duration	12	su_attempted	23	srv_serror_rate	34	dst_host_srv_diff_host_rate
2	src_bytes	13	num_root	24	rerror_rate	35	dst_host_serror_rate
3	dst_bytes	14	num_file_creations	25	srv_rerror_rate	36	dst_host_srv_serror_rate
4	Land	15	num_shells	26	same_srv_rate	37	dst_host_rerror_rate
5	wrong_fragment	16	num_access_files	27	diff_srv_rate	38	dst_host_srv_rerror_rate
6	Urgent	17	num_outbound_cmds	28	srv_diff_host_rate	39	protocol_type
7	Hot	18	is_host_login	29	dst_host_count	40	service
8	num_failed_logins	19	is_guest_login	30	dst_host_srv_count	41	flag
9	logged_in	20	Count	31	dst_host_same_srv_rate		
10	num_compromised	21	srv_count	32	dst_host_diff_srv_rate		

#	Description	#	Description	#	Description	#	Description
11	root_shell	22	serror_rate	33	dst_host_same_src_port_rate		

4.2 Feature selection

Feature selection extracts statistically relevant features from the datasets. We used the information gain and ranker feature selection method. The information gain evaluates the statistical significance of each feature based on the reduction of entropy after a dataset is split at that feature. The ranker method then ranks which features should be low rank or high ranked according to the feature selected. In most cases, all features with less than 1% information gain were removed from the feature-selected datasets. Table 2 summarises the features selected for each attack type and the full 20% training dataset. We used the WEKA toolkit (Hall et al., 2009). After the relevant features for each attack type were established, the datasets were trimmed to only include those highlighted during the feature selection process.

Table 2: Features selected for training ATM

Training Dataset	Features Selected
Probe	2, 40, 3, 30, 34, 9, 33, 32, 31, 38, 37, 41, 24, 25, 29, 39, 20, 26, 28, 27, 21, 22, 35, 1, 36
U2R	11, 40, 7, 30, 10, 29, 14, 1, 33, 21, 6, 13, 9, 41, 39, 19, 4
R2L	2, 40, 3, 30, 7, 33, 34, 21, 20, 29, 19, 28, 32, 39, 1, 36, 31, 41, 8, 9, 35, 27, 11, 4
DOS	2, 27, 26, 41, 40, 3, 20, 36, 35, 31, 32, 22, 23, 30, 9, 34, 29, 33, 21, 28, 38, 39, 1, 37, 24, 25
20% Training	2, 40, 3, 41, 27, 26, 30, 31, 32, 35, 9, 36, 22, 20, 23, 34, 29, 33, 28, 21, 38, 39, 24, 37, 25, 1

4.3 Evaluation techniques

Two techniques, performance metrics based on a confusion matrix and a cost evaluation taking account several factors, were chosen to validate the classifier's performance.

4.3.1 Performance metrics

Each event classified by the classifier can have four possible outcomes as illustrated in the confusion matrix in Table 3. Wu and Banzhaf (2010) stated that the effectiveness of an IDS should be evaluated by their ability to give correct classifications; they indicated most known performance metric as the Detection Rate (DR) with the False Alarm Rate (FAR), and that IDSs should have a high DR and low FAR.

Table 3: Confusion matrix (TP=true positive, FN=false negative, FP=false positive and TN = true negative)

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

On its own, the confusion matrix does not really represent any usable metrics to evaluate the IDS. Other important measures are (Wu & Banzhaf, 2010):

$$DR = TP \,/\, (TP + FN) \tag{1}$$

$$FAR = \frac{FP}{TN + FP} = 1 - TN \,/\, (TN + FP) \tag{2}$$

$$Error\ rate = (\mathrm{FP} + \mathrm{FN}) \,/\, (\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}) \tag{3}$$

$$Accuracy = TN + TP \,/\, (TN + TP + FN + FP) \tag{4}$$

$$F_{measure} = \frac{2 \times (TP/(TP + FP)) \times (TP/(TP + FN))}{(TP/(TP + FP)) + (TP/(TP + FN))} \tag{5}$$

The FAR refers to the number of normal instances classified as attacks, whereas the error rate numerates the number of incorrect predictions in total. The F-measure indicates the accuracy of a test by numerating a balance between the precision and recall. It is useful when test datasets are used, especially with unbalanced datasets. Unbalanced datasets tend to have a high accuracy and a low error rate but high FAR, by favouring the most common class (Weng & Poon, 2008). In these cases, the F-measure is considered to be superior. The accuracy rate is appreciated when training and test data are shared (Tavallaee et al., 2010).

4.3.2 Cost evaluation

Axelsson (2000) iterated the importance of computational complexity in ID, drawing specific focus to the resources, storage and specifically time (can it perform in real-time). Our aim is to establish ATM as a classifier for ID, either real-time or offline. The core of our cost analysis is the result in the time and complexity of the constructed DTs. The time in seconds as a comparison result may be irrelevant due to different computer setups (Tavallaee et al., 2010). We argue that transparent design motivates reproducibility for use as an evaluation technique. Ling, Yang, Wang, et al. (2004) emphasized that less leaf nodes result in reduced cost in DTs. The time taken to construct the decision tree is also a consideration.

The quality of the decision tree is based on the relationship between training examples and the error rate, in ATM it influences the amount of pheromone required to produce the model (Otero et al., 2012). A high quality tree uses more pheromone. Our cost score follows (lowest score represents best cost):

$$CS = (log(LN) \times \frac{T}{60})/(1 - TQ) \quad (6)$$

with *LN* = total number of leaf nodes within the model, *T* = the time taken to build in seconds, and *TQ* = total tree quality for the DT.

4.4 Classifier initialization

The main purpose this study was to experiment with the ATM classifier applied in the ID domain. The java binaries can be obtained from the project myra github repository (Otero, 2015). Version 4.1 of the Ant Tree Miner classifier was used with tuning of the following parameters:

- Colony Size (cs) - number of artificial ants within each colony
- Max Iterations (mi) – iterates until a global-best tree is found or mi is reached
- Evaporation Factor (ef) - factor to which pheromone will evaporate for each entry in the pheromone matrix

The optimal parameters for the classifier are identified by performing a validation experiment cycle, using the validation datasets. Increasing the parameters will not improve performance or accuracy much as indicated by Otero et al. (2012): a colony size of 5 and evaporation factor of 0.9 performs slightly better than other combinations, and on average 200 iterations were performed until the classifier converges. Several versions of the ATM were experimented with, each with different settings in an attempt to tune the parameters. Table 4 summarizes the parameters, and Table 5 shows the results. Each experiment was performed 10 times and the results averaged for analysis.

Table 4: ATM validation experiment parameters

Parameter	ATM - df	ATM – cs	ATM - ef	ATM - mi	ATM - op
Colony Size	5	2	5	5	2
Max Iterations	200	200	200	50	150
Evaporation Factor	0.9	0.9	0.3	0.9	0.9

Table 5: ATM validation experiment results

Evaluation	DF	CS	EF	MI	OP
Error Rate	0.50%	0.65%	0.48%	1.63%	0.95%
Accuracy	99.50%	99.35%	99.52%	98.37%	99.05%
DR	99.75%	99.60%	99.72%	98.76%	99.36%
FAR	0.00789	0.00939	0.00714	0.02074	0.01312
Tree Quality	99.24%	99.13%	99.13%	98.11%	98.82%
Leaf Nodes	197.40	165.00	215.10	115.70	130.40
Runtime (s)	295.08	98.46	489.98	65.05	76.99
F-measure	1.00	0.99	1.00	0.98	0.99
Cost	1477.21	419.60	2200.97	118.33	230.41

ATM – DF designates the default parameters for the ATM classifier as selected by Otero et al. (2012).

The ATM – op version was considered the optimal parameter based on the results. The results revealed that reducing the colony size vastly reduces the classifier cost while only slightly affecting accuracy. Reducing the colony size favors the reduction of leaf nodes. The results contradict evaluations by Rami and Panchal (2012) who showed an increase in the colony size decreased the accuracy. Decreasing the evaporation factor builds more accurate classifiers, but does not warrant the significant increase in cost, as this increases the amount of pheromone used. When the maximum iterations are decreased, the accuracy slightly declines, however with exceptional reduction in cost, the parameter will influence the runtime significantly. However, Rami and Panchal (2012) noted the accuracy decreased using the ATM classifier whenever the number of iterations reached 1000, thus supporting the search for optimal parameters based on lowering the values.

4.5 Experiment setup

Tavallaee et al. (2010) highlighted the concerns and unreliability of evaluating classifiers without unseen data. With the experiment run two cycles where performed. The first cycle focus on the classifier's ability to classify the full NSL-KDD dataset, whereas the Train 20% and Test21 dataset were used.

The second cycles digs deeper, peeking into the classifier's ability to detect each attack on its own. Each experiment cycle was performed 10 times and the results averaged for analysis. The experiments were performed on a Lenovo Ideapad 510s, Windows 10, an Intel Core i5-6200U processor and 8GB DDR4 ram.

5. Results and discussion

Our results are scrutinized per experiment cycle performed. We have highlighted the significance of conducting reliable, reproducible and transparent research. Several studies on classifiers in ID promise detection and accuracy rates of over 99%, but this needs to be considered in the context of how the experiments were performed. Without applying the basic principles these studies have become almost impossible to replicate or compare. There is a definitive issue within the cross domain research areas to provide comparable results, and this statement is supported by Tavallaee et al. (2010) and Sommer & Paxson (2010). Whilst creating the NSL-KDD dataset Tavallaee et al. (2009) also experimented with several classifiers on their Test21 dataset. We can compare the Test21 results with the FULL and FULLFS results obtained: in the results from the validation and test cycles, the high accuracy rates are significantly reduced, as the Test21 dataset includes hard to detect attacks (Tavallaee et al., 2009). This explains the significant drop in accuracy across the board.

Cases where cross-validation or the split of a training dataset (as with the validation dataset) are not comparable due to the selection process involving an essence of randomness when the test datasets were created. Table 6 summarises the results obtained from the experiments run using the ATM – op version (FS denotes the feature selected experiments performed; ATMa Per Attack represents the results from training the ATM per attack type and then combining each prediction model, tree quality and leaf nodes are averaged).

Table 6: ATM results on NSL-KDD Test21 dataset

Evaluation	FULL	FULL FS	DOS	DOS FS	Probe	Probe FS	R2L	R2L FS	U2R	U2R FS	ATMa Per Attack
Error Rate (%)	39.72	39.06	24.99	24.89	21.99	22.63	52.10	51.56	7.99	8.31	35.15
Accuracy (%)	60.28	60.94	75.01	75.11	78.01	77.37	47.90	48.44	92.01	91.69	64.85
DR (%)	84.94	84.11	93.40	91.71	88.79	90.24	98.22	98.17	99.86	99.88	57.05
FAR (%)	45	44	34	33	32	34	91	90	92	96	0
Tree Quality (%)	98.91	99.00	99.72	99.69	99.39	99.42	99.55	99.56	99.91	99.91	99.64
Leaf Nodes	207.10	158.20	83.10	96.20	63.50	107.40	35.20	28.20	3.00	3.50	46.2
Runtime (s)	241.46	239.09	157.89	134.27	61.45	59.75	20.10	20.35	2.04	1.74	241.48
F-measure	0.44	0.44	0.71	0.71	0.79	0.79	0.62	0.63	0.96	0.96	0.73
Cost	852.67	875.73	1819.41	1419.46	303.34	347.52	114.51	110.65	17.69	17.29	1861.01

5.1 KDD NSL-Test21 full discussion

Table 7 outlines the results of Tavallaee et al. (2009), evaluating several machine learning classifiers on their Test21 dataset. The J48 DT employs the C4.5 split. Intuitively, RF and NB Tree can all be regarded as ensemble methods i.e. a combination of classifiers or prediction models and marked * in Table 7. The ATM is however, not truly an ensemble classifier, and comparisons should be done with novel algorithms such as J48, Random Tree, SVM and M-Layer Perceptron. The ATMa which combines prediction models can be considered an ensemble classifier and compared with other ensemble classifiers such as random forests or NB tree.

Table 7: Accuracy of several classifiers on Test21 dataset

Classifier	Accuracy (%)
SVM	42.29
NB Tree*	66.16
Random forest *	63.26
M-Layer Perceptron	57.34
Random Tree	58.51
J48 (DT)	63.97
ATM	60.94
ATMa*	64.85

With few applicable results to compare with the ATM's performance, the classifier obtained more than satisfactory results in classifying all attacks, considering only the accuracy. Excluding the ensemble classifiers, the ATM obtains results only 3% lower than the traditional DT classifier and makes a very strong case as an intrusion classifier by outperforming several other classifiers. Overall, the novel ATM ranks 4th when compared to other novel and ensemble classifiers. However, when we ensemble the classification models build from training each attack type, the classifier scores 65% accuracy. The ensemble result is denoted as ATMa in Table 6 and 7. ATMa outperforms most classifiers and highlights vast potential for improving the ATM classifier by means of ensemble techniques. The approach is similar to bagging, however instead of randomly sampling the dataset, the ATMa classifies data based on the prediction models built from each attack type. The improvement in accuracy comes at a cost due to the high tree quality and runtime. By ensembling the ATM the FAR rate is reduced to 0%, the low detection rate of 57% raises another potential area for improvement. When compared with the FULL results obtained the ATMa can be considered a better ID classifier.

5.2 Results per attack type

By splitting the datasets per attack type, the ATM faced a highly unbalanced dataset; this is supported by the results when considering low error rates and high false alarm rates. Regrettably, comparative results split per attack type, using the same training and test dataset, are scarce. The ATM obtained the best accuracy when classifying U2R attacks (92.01% accuracy), with the highly unbalanced dataset the F-measure of 0.96 could be considered a very good result. This is very significant since U2R attacks are considered to be exceptionally dangerous. Revathi and Malathi (2014) experimented with detecting U2R attacks using the NSL-KDD dataset, and their Multi-layer perceptron obtained the highest result with 88.46% accuracy. They however failed to mention which training and test datasets were used. The ATM's ability to detect DOS and Probe attacks could be considered favourable, as this includes the lowest percentage of FAR with high DR over 90%. The accuracy of 78% can also be considered to be good. It is important to keep in mind that the Test21 dataset only includes attacks that are hard to detect. Noureldien and Yousif (2016) evaluated the accuracy of various machine learning techniques to detect DOS attacks, using the Train20% and Test21 NSL-KDD datasets. The ATM classifier outperforms the J48 DT classifier in detecting DOS attacks, by 3%. In general, we note the ATM classifier performs better when split between each attack type than tasked with detecting all types of attacks.

6. Conclusion and future research potential

In this paper, the ATM classifier has been successfully implemented in the ID domain. The results obtained are satisfactory and among the top percentile when compared with several other classification techniques. The results revealed by training the classifier per attack type and then ensemble the individual classification models, outperforms most classifiers when compared with result obtained by Tavallaee et al. (2009). Our ensemble ATMa was able to classify attacks with 65% accuracy, with a FAR rate of 0%. This shows vast potential for implementing ensemble techniques with the ATM classifier. This study also supports the fact that DT based classifiers are still among the best to use in ID. The use of a new cost metric to analyze the ATM classifier allows a baseline for further experiments, especially improved versions of the classifier. The cost formula is specifically

tailored to the ATM classifier and has shown usefulness when performing validation experiments. The contradicting results found in evaluating the parameters highlights a need for further investigation, specifically including some affinity to the costs. The high quality processes with regards to comparability, transparency and reproducibility within the experiments instantly vanquishes the current academic issues found in similar research. Even though the ATM classifier shows favorable results, from an intrusion detection perspective the classifier will be better for signature based detection, this statement is supported by the disparity between validation and Test21 dataset results. We argue that the ATM will be very apt as it shows high accuracy, and low FAR rates during the validation experiments. Several avenues for future research emerged. Albayati and Issac (2015) noted DT perform better when tasked with large datasets, the ATM classifier was limited to training on only 20% of the full NSL-KDD dataset. Chennupati (2014) highlighted the instability of the ATM classifier and applied the bagging ensemble method to significantly improve accuracy and reduce error rates. Our own ATMa improved the accuracy by combining the prediction models from each attack type model, similar to the bagging approach. With a baseline for ATM within ID established, it is evident that other future research should involve applying ensemble methods on the ATM classifier specifically in intrusion detection. In general, the implementation and results are very good when compared with other well-known methods such as DT, SVM and Multi-layer perceptron.

References

Aghdam, M. H. & Kabiri, P. (2016) 'Feature Selection for Intrusion Detection System Using Ant Colony Optimization.', *IJ Network Security* 18(3), 420--432.

Albayati, M. & Issac, B. (2015) 'Analysis of intelligent classifiers and enhancing the detection accuracy for intrusion detection system', *International Journal of Computational Intelligence Systems* 8(5), 841--853.

Axelsson, S. (2000) 'The base-rate fallacy and the difficulty of intrusion detection', *ACM Transactions on Information and System Security (TISSEC)* 3(3), 186--205.

Bilge, L. & Dumitras, T. (2012) Before we knew it: an empirical study of zero-day attacks in the real world, *in* 'Proceedings of the 2012 ACM conference on Computer and communications security', pp. 833--844.

Breiman, L.; Friedman, J.; Stone, C. J. & Olshen, R. A. (1984) *Classification and regression trees*, CRC press.

Burdette, P. (2016) 'Timeline of an attack', *Network Security* 2016(9), 16--17.

Chennupati, G. (2014) 'eAnt-Miner: An Ensemble Ant-Miner to Improve the ACO Classification', *arXiv preprint arXiv:1409.2710*.

Dietterich, T. G. (2000) Ensemble methods in machine learning, *in* 'International workshop on multiple classifier systems', pp. 1--15.

Dorigo, M. & Gambardella, L. M. (1997) 'Ant colonies for the travelling salesman problem', *biosystems* 43(2), 73--81.

Frei, S. (2014) 'The Known Unknowns', *Update*.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P. & Witten, I. H. (2009) 'The WEKA data mining software: an update', *ACM SIGKDD explorations newsletter* 11(1), 10--18.

Hoque, M. S.; Mukit, M.; Bikas, M.; Naser, A. & others (2012) 'An implementation of intrusion detection system using genetic algorithm', *arXiv preprint arXiv:1204.1336*.

Kemmerer, R. A. & Vigna, G. (2002) 'Intrusion detection: a brief history and overview', *Computer* 35(4), 27--30.

Kumar, S. (2007) 'Survey of current network intrusion detection techniques', *Washington Universty St. Louis*.

Lee, W.; Fan, W.; Miller, M.; Stolfo, S. J. & Zadok, E. (2002) 'Toward cost-sensitive modelinig for intrusion detection and response', *Journal of computer security* 10(1-2), 5--22.

Ling, C. X.; Yang, Q.; Wang, J. & Zhang, S. (2004) Decision trees with minimal costs, *in* Proceedings of the twenty-first international conference on Machine learning', pp. 69.

Lypez-Ibбnez, M.; Stützle, T. & Dorigo, M. (2015) 'Ant colony optimization: A component-wise overview', *Techreport, IRIDIA, Universite Libre de Bruxelles*.

Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A. & Rajarajan, M. (2013) 'A survey of intrusion detection techniques in cloud', *Journal of Network and Computer Applications* 36(1), 42--57.

Mohammad, M. N.; Sulaiman, N. & Khalaf, E. T. (2011) 'A novel local network intrusion detection system based on support vector machine', *Journal of Computer Science* 7(10), 1560.

Nguyen, H. A. & 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.

Noureldien, N. A. & Yousif, I. M. (2016) 'Accuracy of Machine Learning Algorithms in Detecting DoS Attacks Types', *Science and Technology* 6(4), 89--92.

Otero, F. E. B.; Freitas, A. A. & Johnson, C. G. (2012) 'Inducing decision trees with an ant colony optimization algorithm', *Applied Soft Computing* 12(11), 3615--3626.

Parihar, L. S. & Tiwari, A. (2016) 'Survey on Intrusion Detection Using Data Mining Methods', *International Journal for Science and Advance Research In Technology*.

Quinlan, J. R. (1986) 'Induction of decision trees', *Machine learning* 1(1), 81--106.

Quinlan, J. R. (2014) *C4. 5: programs for machine learning*, Elsevier.

Rai, K.; Devi, M. S. & Guleria, A. (2016) 'Decision Tree Based Algorithm for Intrusion Detection', *International Journal of Advanced Networking and Applications* 7(4), 2828.

Rami, S. P. & Panchal, M. H. (2012) 'Comparative Analysis of Variations of Ant-Miner by Varying Input Parameters', *International Journal of Computer Applications* 60(3).

Ramos, V. & Abraham, A. (2005) Antids: Self orga nized ant-based c lustering model for intrusion det ection system'Soft Computing as Transdisciplinary Science and Technology', Springer, , pp. 977--986.

Revathi, S. & Malathi, A. (2014) 'Detecting user-to-root (U2R) attacks based on various machine learning techniques', *International Journal of Advanced Research in Computer and Communication Engineering* 3(4), 6322--6324.

Scarfone, K. & Mell, P. (2007) 'Guide to intrusion detection and prevention systems (idps)', *NIST special publication* 800(2007), 94.

Singh, J. & Nene, M. J. (2013) 'A survey on machine learning techniques for intrusion detection systems', *International Journal of Advanced Research in Computer and Communication Engineering* 2(11), 4349--4355.

Sommer, R. & Paxson, V. (2010) Outside the closed world: On using machine learning for network intrusion detection, *in* 'Security and Privacy (SP), 2010 IEEE Symposium on', pp. 305--316.

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.

Tavallaee, M.; Stakhanova, N. & Ghorbani, A. A. (2010) 'Toward credible evaluation of anomaly-based intrusion-detection methods', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40(5), 516--524.

Tesfahun, A. & Bhaskari, D. L. (2013) Intrusion detection using random forests classifier with SMOTE and feature reduction, *in* 'Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference', pp. 127--132.

Tsai, C.-F.; Hsu, Y.-F.; Lin, C.-Y. & Lin, W.-Y. (2009) 'Intrusion detection by machine learning: A review', *Expert Systems with Applications* 36(10), 11994--12000.

Tsang, C.-H. & Kwong, S. (2006) Ant colony clustering and feature extraction for anomaly intrusion detection'Swarm Intelligence in Data Mining', Springer, , pp. 101--123.

Weng, C. G. & Poon, J. (2008) A new evaluation measure for imbalanced datasets, *in* 'Proceedings of the 7th Australasian Data Mining Conference-Volume 87', pp. 27--32.

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

Zhang, J. & Zulkernine, M. (2005) Network Intrusion Detection using Random Forests., *in* 'PST'.