A MACHINE LEARNING APPROACH WITH VERIFICATION OF PREDICTIONS AND ASSISTED SUPERVISION FOR A RULE-BASED NETWORK INTRUSION DETECTION SYSTEM

José Ignacio Fernández-Villamor and Mercedes Garijo
Departamento de Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Spain

Keywords: Network Intrusion Detection Systems, Rules of inference, Machine learning, Decision trees, Self-organizing maps.

Abstract: Network security is a branch of network management in which network intrusion detection systems provide attack detection features by monitorization of traffic data. Rule-based misuse detection systems use a set of rules or signatures to detect attacks that exploit a particular vulnerability. These rules have to be hand-coded by experts to properly identify vulnerabilities, which results in misuse detection systems having limited extensibility. This paper proposes a machine learning layer on top of a rule-based misuse detection system that provides automatic generation of detection rules, prediction verification and assisted classification of new data. Our system offers an overall good performance, while adding an heuristic and adaptive approach to existing rule-based misuse detection systems.

1 INTRODUCTION

Network security is an important branch of network management, in which network intrusion detection systems (Mukherjee et al., 1994) provide attack detection features by monitorization of traffic data in demilitarized zones. Therefore, intrusion detection systems are a field of interest in the provision of network security under the assumption of existence of security vulnerabilities on hardware and software systems.

Intrusion detection systems can be classified into *anomaly detection systems* (Denning, 1987), which base detection on identification of abnormal user behaviour, and *misuse detection systems* (Deri et al., 2003), which base detection on identification of well-known attack patterns. An example of misuse detection system is Snort (Roesch, 1999), the *de facto* standard open source intrusion detection system. Snort uses a set of rules to detect attacks that exploit a particular vulnerability. These rules have to be hand-coded by experts to properly identify vulnerabilities. Not only does this require an expertise on the field but it also implies an effort of analysis of traffic data, which results in Snort having limited extensibility. This process implies: knowing which attacks the system was not prepared for, classifying these attacks and, finally, building new detection rules.

This paper proposes a machine learning layer on top of a rule-based misuse detection system such as Snort to provide automatic generation of detection rules, prediction verification and assisted classification of new data. On the compromise of expressing precise detection rules that are tied to a particular exploit against generating more general rules, our approach tries to be more untied by using, e.g., key indicators (Lee et al., 1999) instead of well-defined message patterns, so that the system is more suited for future unknown attacks. In other words, the system is aimed at detecting the nature of traffic data and classifying it into normal or abnormal traffic instead of focusing on identifying exploits or particular attacks. We think this is consistent with the idea of providing automation and ensuring freshness in attack detection as an added value to misuse detection systems, leaving the exploit identification task for security experts. The whole maintenance lifecycle of an intrusion detection system is considered, and a training pattern labeller, based on estimated accuracy of rules and self-organizing maps, is proposed to allow assisted classification of new traffic data.

Ignacio Fernández-Villamor J. and Garijo M. (2008).
A MACHINE LEARNING APPROACH WITH VERIFICATION OF PREDICTIONS AND ASSISTED SUPERVISION FOR A RULE-BASED NETWORK INTRUSION DETECTION SYSTEM.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 143-148
DOI: 10.5220/0001524801430148

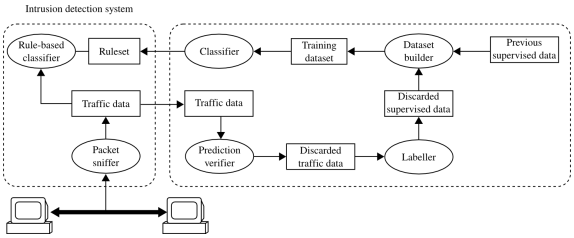

Figure 1: System architecture.

2 SYSTEM DESCRIPTION

A rule-based intrusion detection system has a set of detection rules that allows the detection of a particular set of attacks. To ensure freshness of detection capabilities new rules have to be included to detect attacks that the system was not prepared for previously. This involves three tasks: knowing which attacks the system was not prepared for, classifying these attacks and, finally, building new detection rules.

Our system is aimed at automating all these tasks as much as possible while being built on top of a rule-based intrusion detection system. The architecture is shown on figure 1 and has four basic modules:

- A classifier, trained to classify samples of traffic data to perform the basic functionality of a network intrusion detection system.
- A prediction verifier, which validates the predictions made by the classifier, in order to detect new traffic types and do reinforcement learning with them.
- A labeller for classification of new data, intended for reducing the effort of classification of new traffic data by grouping similar samples into clusters of the same class.
- A data set builder, which prepares a data set to retrain the classifier.

The system is essencially a rule generator, so that realtime classification of data is performed by the rule-based network intrusion detection system, whereas our modules only perform off-line tasks. More specifically, the rule generation process starts after a set of traffic data is collected out of the intrusion detection system and proceeds as follows:

1. The prediction verifier estimates the validity of previous predictions of traffic classes and builds up a data set out of discarded samples, which require further supervision due to the system's inability to classify them properly.
2. The traffic data is supervised by a human agent by labelling data clusters which are generated by the labeller.
3. The data set builder constructs a data set out of samples from previous supervised traffic data and the newly supervised data.
4. The classifier is trained with the generated data set, which results in the generation of a set of rules that are used to refresh the intrusion detection system ruleset.

2.1 Machine Learning of Traffic Data

Several approaches have been considered for machine learning of attack patterns for intrusion detection systems. Neural networks have been used to achieve this task (Chavan et al., 2004; Kemmerer and Vigna, 2005), but have difficulties to generalize their knowledge and therefore to detect attacks that are not present in the training data (Bouzida and Cuppens, 2005). Other approaches have been used, such as statistical models (Ye et al., 2001; Ye et al., 2003) or Petri nets (Kumar and Spafford, 1994). None of them can be used naturally to build detection rules and thus are unpractical for our purposes.

Decision trees and rule-based systems (Hunt, 1962) have been also used for intrusion detection (Yu et al., 2007; Chavan et al., 2004) and offer good performance in terms of prediction rates and generalization to new attacks (Bouzida and Cuppens, 2005). The proposed system uses the rule learning-based algorithm C4.5 (Quinlan, 1993), which is essentially an extension of ID3 algorithm aimed at avoiding overfitting.

Considering a training data set, two thirds of it are used as a growing data set, while one third remains as a pruning data set. The growing data set is used to build an ID3 tree, where an entropy gain function is used to partition a data set S w.r.t. an attribute A.

The attribute with maximum entropy gain is chosen to partition the data set at each node, so that a tree is built iteratively. Continuous attributes are handled in an equivalent way by calculating thresholds through interpolation of consecutive values from the data set for each continuous attribute and choosing the threshold with maximum entropy gain.

After the decision tree is built, generation of rules of inference is straightforward by scanning all possible paths in the tree from the top node to its leaves. The left hand side of the rules is a combination of the conditions on each node, while the right hand side is each leaf's class. An estimation of accuracy of each rule is made by calculating the accuracy on the pruning data set. The resulting rules are pruned by removing trailing conditions on the left hand side only when the resulting estimated accuracy is not lower. Finally,

the rules are sorted in decreasing estimated accuracy order.

2.2 Prediction Verification

By using the mentioned classifier, a set of rules with an estimation of accuracy is obtained, which serves as certainty factor in the prediction of classes of traffic data. At detection time, a particular rule with an associated estimated accuracy will fire. At this point, it is possible to force a minimum estimated accuracy threshold A_{th} to accept a prediction, being this an heuristic that serves to discern traffic data that was considered at training time from data that was not. Therefore, setting an accuracy threshold lets populate a data set with data that is supposed to be new to the system and which therefore needs proper classification. As a result, the estimated accuracy of rules lets integrate prediction verification capabilities into the system.

2.3 Classification of New Data

A sample is regarded as new if the rule-based classifier is not able to properly classify it as normal traffic nor any kind of attack traffic, basing the decision upon an estimated accuracy threshold. As a result, this data needs manual supervision by an external agent for its classification. However, further help can be provided in this task by automatically grouping similar traffic data. Our system uses self-organizing maps to achieve this, which have proven useful in other works (Bashah and Shanmugam, 2005; Hoglund et al., 2000). Self-organizing maps (Kohonen, 1997) use an euclidean-similarity metric to achieve automatic clustering of data by defining an overlaying set of reference vectors on the feature space of the sample data set. Local-order relations are set on the reference vectors so that their values are depedent to each other neighbouring vector. The self-organizing algorithm defines a non-linear regression of the reference vectors through the data points, which results in the reference vectors being scattered among the space according to the data set's probability density function. This lets classifying all data samples that are represented by the same reference vector in one step and thus reducing the effort of supervision.

When dimensioning the self-organizing map, some problems need to be overcome such as choosing a number of nodes that makes the map able to adapt to all the data set or enhancing rare cases to be considered apropriately by the map. Applying the self-organizing map algorithm to all the data set might result in the map's inability to adapt to euclidean-too-separated values and might also fail to consider rare cases that are not too relevant in the probability density function. To prevent this from happening, visual inspection of Sammon's mappings (Sammon, 1969) of different maps helps to choose a correct form of the array or adapt the probability density function, but is a manual task that is not desired in our system and therefore a different approach is used. In our case, the system performs a division in several subsets of the original discarded set to try to obtain subsets with similar features and increase the self-organizing map's accuracy. Different heuristics can be used to perform this division, such as partitioning through certain fields like protocol type or type of service (Yu et al., 2007), being all these approaches aimed at reducing information entropy of the resulting subset. The classifier's ruleset is a pruned version of a decision tree that, as described on section 2.1, is built through information entropy reduction with the supervised training data set, and thus is a possible heuristic for reducing information entropy on the discarded samples data set.

To achieve the subdivision, samples are grouped in our system by hierarchical coincidence of the classifier's rule clauses. More precisely, each sample fires a particular rule, whose left-hand side is defined by a list of clauses $(c_1, c_2, ..., c_i)$, ordered by classification relevance as a result of the C4.5 algorithm. Therefore, this allows hierarchical grouping of similar samples by removing trailing clauses and grouping all the samples that share the same clauses. A depth value needs to be set in this case, with a higher value resulting in obtaining a higher number of subsets, and a lower value producing bigger ones with more heterogeneous samples. The resulting sequence of clauses is extended with protocol, type of service and flag fields to build a subset identifier for each sample.

Finally, the self-organizing map algorithm is applied on every subset. A 3:2 aspect ratio is used on the maps' dimensions in order to favour learning stability, with an hexagonal topology and a total number of nodes which is equal to 10% of the subsets cardinality with a dimensions limit of 30x20.

2.4 Retraining

C4.5 rule-learning algorithm is batch-training-based. To provide reinforced learning, the approach which has been used in this system is to build a new data set with different proportions of samples. At this point, three types of samples are found in the system: discarded samples during prediction verification, training data set samples that are detected correctly and training data set samples that are not detected cor-

Table 1: Classifier performance on the training data set.

Prediction / real	normal	probe	dos	u2r	r2l	Total
normal	99.95%	1.23%	0.01%	25.00%	8.66%	99.77%
probe	0.01%	98.52%	0.00%	0.00%	0.79%	99.26%
dos	0.02%	0.25%	99.99%	0.00%	0.00%	99.99%
u2r	0.00%	0.00%	0.00%	75.00%	0.00%	100.00%
r2l	0.02%	0.00%	0.00%	0.00%	90.55%	98.29%
Total	99.95%	98.52%	99.99%	75.00%	90.55%	99.94%

Table 2: Classifier performance on the testing data set.

Prediction / real	normal	probe	dos	u2r	r2l	Total
normal	99.49%	17.76%	2.76%	54.29%	90.79%	73.29%
probe	0.26%	70.21%	0.01%	0.00%	3.16%	80.91%
dos	0.22%	12.03%	97.22%	0.00%	0.03%	99.72%
u2r	0.02%	0.00%	0.00%	35.71%	2.62%	5.38%
r2l	0.01%	0.00%	0.00%	10.00%	3.40%	95.52%
Total	99.49%	70.21%	97.22%	35.71%	3.40%	92.36%

rectly. The proportion of samples of each kind and the total amount of them, in the form of the triple (n_+, n_-, n_{dis}), will determine the newly built data set and thus the classification capabilities of the new classifier.

3 PERFORMANCE EVALUATION

The system has been evaluated against KDD'99 data set (University of California, 1999), from The Third International Knowledge Discovery and Data Mining Tools Competition. This data set is made up of almost five million samples of data connections that are classified and grouped under a set of traffic classes called *normal* (ordinary, non-malicious traffic), *probe* (monitorization and probing activities), *dos* (denial of service attacks, which often imply flooding activities), *u2r* (user to root, which refers to users trying to acquire root privileges) and *r2l* (remote to local, which refers to remote unauthorized log-in).

A total of 41 attributes define each connection, with symbolic fields such as type of service or type of transport connection and continuous fields such as average packet size or login attempts. Some of these fields are aggregated ones, e.g., connections to the same host in a 2-second window, and are included due to its proved relevance in attack detection (Lee et al., 1999). Other features of the data set are data inconsistency, in the sense that certain samples with the same fields belong to different classes, and existence of new attack types in the testing data set, which allows evaluation of generalization capabilities of classifiers. Therefore, all these data set features and its heuristic approach makes this data set an appropriate tool to tune our system and evaluate it.

3.1 Classifier Performance

Our rule-based classifier was trained with a subset of the KDD'99 training data set. Performance on the training set and the testing set is shown on tables 1 and 2, respectively. Its difficulty to detect certain attacks in the training data set is consequence of its compromise to generalise to new attacks, which is achieved by the rule-pruning phase described in section 2.1, with the classifier presenting the known difficulties on this sample data (Bouzida and Cuppens, 2005). Overall performance is comparable with other classifiers, such as KDDCup'99 winner (Pfahringer, 1999), a boosting-based classifier which offers 92.71% accuracy.

As described in section 2.2, a prediction verifier is used to discard potential samples whose predictions might be *a priori* regarded as unacceptable. This is achieved by using rules' estimation of accuracy, calculated on the pruning data set, as the confidence factor. The results of different accuracy thresholds A_{th} are shown on table 3, which, as expected, lets improve the overall accuracy of the classifier. This allows our classifier to outperform all others, at the cost of marking conflicting samples as discarded. By observing the results, an accuracy threshold of 0.98 seems an appropriate value by offering an acceptable compromise between packet discard ratio and accuracy and thus has been used in the rest of experiments in this paper.

3.2 Labeller Performance

Discarded samples are collected by the prediction verifier for further supervision. In our system, this task is assisted by the sample labeller. The discarded samples obtained from the classifier are grouped into a number of subsets according to the sample fields and rule clauses. Afterwards, the self-organizing map al-

Table 3: Effect of accuracy threshold.

A_{th}	Discards	Accuracy
0.0	0.00%	92.36%
0.9	1.13%	93.11%
0.95	1.59%	93.19%
0.96	1.59%	93.19%
0.97	1.59%	93.19%
0.98	1.83%	93.21%
0.99	5.73%	94.07%
0.995	5.73%	94.07%
0.999	5.73%	94.07%
0.9999	100.00%	–

gorithm is applied on these subsets, and a number of nodes is obtained, being this number proportional to the subsets cardinality. Different rule depth values can be used in the subdivision of the original discarded samples data set. Different results were obtained, as shown on table 4, with higher depth values offering a higher accuracy. As long as the highest depth value requires classification of only 15% of samples while offering the highest accuracy, it can be considered the optimal depth value for the labeller and has been used in further experiments.

Table 4: Labelling performance.

Depth	Subsets	Nodes	Accuracy
0	74	13.36%	89.03%
2	95	13.89%	90.43%
4	118	14.43%	91.41%
6	150	15.71%	93.19%
8	162	15.94%	95.33%

3.3 Overall Performance

After labelling of discarded packets has been achieved, the classifier is retrained by building a new training set. Different parameters are possible when building the new training set; n_+ and n_- determine the proportion of training samples that were correctly and incorrectly classified, respectively, and n_{dis} determines the proportion of samples that were discarded during prediction time. The result of varying these parameters is shown on table 5. Accuracies A_+ and A_- are calculated on the correct and incorrect subsets of the training data set, while A_{dis} is calculated on the correctly labelled discarded data set. Overall accuracy A is calculated on the full testing data set.

By observing the results, the classifier shows an optimal performance with $n_+ = 0.3$, $n_- = 0.1$ and $n_{dis} = 0.6$. It is noticeable that there is a top accuracy that is achievable in the discarded data set which is determined by the accuracy of the labeller. Also, apparently the overall accuracy should have a value between A_+, A_- and A_{dis}, although this does not happen due to the fact that the newly built training data was a combination of correctly and incorrectly classified samples from the training set and discarded samples from the testing set, while this last set is built heuristically and therefore does not necessarily include the problematic samples which would be useful for a second learning phase.

Table 5: Performance after retraining.

n_+	n_-	n_{dis}	A_+	A_-	A_{dis}	A
0.5	0.4	0.1	99.71%	100.0%	92.55%	92.32%
0.5	0.1	0.4	99.64%	100.0%	94.61%	93.32%
0.3	0.2	0.5	99.47%	100.0%	95.08%	93.41%
0.4	0.1	0.5	99.56%	100.0%	94.94%	93.46%
0.5	0.3	0.2	99.54%	100.0%	93.45%	93.53%
0.3	0.1	0.6	99.57%	100.0%	94.94%	93.63%

4 RELATED WORK

Rule-based systems have been widely used in previous intrusion detection systems. Their performance in terms of accuracy and speed makes them appropriate for intrusion detection tasks, while their internal representation of knowledge in the form of rules favours human interpretation.

(Wuu and Chen, 2003) uses discrete attributes and is focused on the generation of attack signatures and thus does not consider prediction verification or assistance on classification of new data.

(Yu et al., 2007) uses a different heuristic for prediction verification based on its boosting-based classifier, which consists of a set of binary rule-based classifiers. Each of their binary classifiers has an associated confidence factor which is combined with the rest of classifiers' to estimate prediction confidence. While their confidence factor shows a good performance, having a set of binary classifiers implies using many rulesets. This contrasts with our approach, which contains an only ruleset and thus can be integrated into an existing rule-based intrusion detection system.

5 CONCLUSIONS

Network intrusion detection systems have to deal with continuous changes in software vulnerabilities, attacks and exploits. Our system reuses a rule-based intrusion detection system such as Snort to implement an adaptive machine-learning-based layer on top of it. While assuming this constraint, our system offers an overall good performance and adds features such

as prediction verification for automatic collecting of problematic data and assisted classification of training data, which favours freshness of detection rules and adds an heuristic and adaptive approach to existing rule-based misuse intrusion detection systems.

ACKNOWLEDGEMENTS

This research is funded in part by the Spanish Government under the R&D project IMPROVISA (TSI2005-07384-C03)

REFERENCES

Bashah, N. and Shanmugam, B. (2005). Artificial Intelligence Techniques Applied to Intrusion Detection. In *IEEE Indicon Conference, Chennai, India*.

Bouzida, Y. and Cuppens, F. (2005). Neural networks vs. decision trees for intrusion detection. In *Proceedings of the 43rd annual Southeast regional conference*.

Chavan, S., Shah, K., Dave, N., and Mukherjee, S. (2004). Adaptive Neuro-Fuzzy Intrusion Detection Systems. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*.

Denning, D. (1987). An Intrusion-Detection Model. In *IEEE transactions on software engineering*.

Deri, L., Suin, S., and Maselli, G. (2003). Design and implementation of an anomaly detection system: An empirical approach. In *Proceedings of Terena TNC, 2003*.

Hoglund, A. J., Hatonen, K., and Sorvari, A. S. (2000). A Computer Host Based User Anomaly Detection System Using Self Organizing Maps. In *Proceedings of the International Joint Conference on Neural Networks, IEEE IJCNN 2000, Vol. 5, pp. 411-416*.

Hunt, E. B. (1962). Concept learning: an information processing problem. Wiley.

Kemmerer, R. A. and Vigna, G. (2005). Hi-DRA: Intrusion Detection for Internet Security. In *Proceedings of the IEEE, October 2005*.

Kohonen, T. (1997). Self-Organizing Maps. Springer-Verlag New York, Inc.

Kumar, S. and Spafford, E. (1994). A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Security Conference*.

Lee, W., Stolfo, S., and Mok, K. W. (1999). A Data Mining Framework for Building Intrusion Detection Models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*.

Mukherjee, B., Heberlein, L. T., and Levitt, K. N. (1994). Network Intrusion Detection. In *IEEE Network*.

Pfahringer, B. (1999). Winning the KDD99 classification cup: Bagged boosting. In *ACM SIGKDD Explor., vol. 1, no. 2, pp 65-66*.

Quinlan, R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, Inc.

Roesch, M. (1999). Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the 13th USENIX conference on System administration*.

Sammon, J. W. (1969). A nonlinear mapping for data structure analysis. IEEE Transactions on Computers, C-18(5):401-409, May 1969.

University of California (1999). The Third International Knowledge Discovery and Data Mining Tools Competition Data. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

Wuu, L. C. and Chen, S. F. (2003). Building Intrusion Pattern Miner for Snort Network Intrusion Detection System. In *IEEE Computer Society*.

Ye, N., Emran, S., Li, X., and Chen, Q. (2001). Statistical Process Control for Computer Intrusion Detection. In *Proceedings DISCEX II*.

Ye, N., Vilbert, S., and Chen, Q. (2003). Computer Intrusion Detection through EWMA for Auto Correlated and Uncorrelated Data. In *IEEE Transactions on Reliability*.

Yu, Z., Tsai, J. J. P., and Weigert, T. (2007). An Automatically Tuning Intrusion Detection System. IEEE Transactions on Systems, Man, and cybernetics, vol. 37, no. 2, April 2007.