Testing Ensembles for Intrusion Detection: on the Identification of Mutated Network Scans

Silvia González[1], Javier Sedano[1], Álvaro Herrero[2], Bruno Baruque[2], and Emilio Corchado[3]

[1]Instituto Tecnológico de Castilla y León
C/ López Bravo 70, Pol. Ind. Villalonquejar, 09001 Burgos, Spain
javier.sedano@itcl.es

[2]Department of Civil Engineering, University of Burgos, Spain
C/ Francisco de Vitoria s/n, 09006 Burgos, Spain
{ahcosio, bbaruque}@ubu.es

[3]Departamento de Informática y Automática, Universidad de Salamanca
Plaza de la Merced, s/n, 37008 Salamanca, Spain
escorchado@usal.es

Abstract. In last decades there have been many proposals from the machine learning community in the intrusion detection field. One of the main problems that Intrusion Detection Systems (IDSs) - mainly anomaly-based ones - have to face are those attacks not previously seen (zero-day attacks). This paper proposes a mutation technique to test and evaluate the performance of several classifier ensembles incorporated to network-based IDSs when tackling the task of recognizing such attacks. The technique applies mutant operators that randomly modifies the features of the captured packets to generate situations that otherwise could not be provided to learning IDSs. As an example application for the proposed testing model, it has been specially applied to the identification of network scans and related mutations.

Keywords: Network Intrusion Detection, Computational Intelligence, Machine Learning, IDS Performance, Classifiers.

1 Introduction

One of the most harmful issues of attacks and intrusions, which increases the difficulty of protecting computer systems, is the ever-changing nature of attack technologies and strategies.

For that reason, among others, IDSs [1], [2], [3] have become an essential asset in addition to the computer security infrastructure of most organizations. In the context of computer networks, an IDS can roughly be defined as a tool designed to detect suspicious patterns that may be related to a network or system attack. Intrusion Detection (ID) is therefore a field that focuses on the identification of attempted or ongoing attacks on a computer system (Host IDS - HIDS) or network (Network IDS - NIDS).

ID has been approached from several different points of view up to now; many different Computational Intelligence techniques - such as Genetic Programming [4], Data Mining [5], [6], [7], Expert Systems [8], Fuzzy Logic [9], or Neural Networks [10], [11], [12] among others - together with statistical [13] and signature verification [14] techniques have been applied mainly to perform a 2-class classification (normal/anomalous or intrusive/non-intrusive).

IDS evaluation is not a clear cut task [15]. Previous works have presented several techniques to test and evaluate misuse detection models for network-based IDSs. A testing technique to prove the effectiveness and capability of any visualization-based IDS, employing numerical data to confront unknown attacks has been previously proposed [16], [17]. In this case, the method is used to asses different classifiers in the detection of mutated network scans. The ability to detect such scans can help identifying wider and potentially more dangerous threats to a network. The main advantage of this testing model is that it provides the classifiers with brand new attacks - network scans in this case -.

A port scan may be defined as a series of messages sent to different port numbers to gain information on their activity status. These messages can be sent by an external agent attempting to access a host to find out more about the network services the host is providing. A port scan provides information on where to probe for weaknesses, for which reason scanning generally precedes any further intrusive activity. This work focuses on the identification of network scans, in which the same port is the target for a number of computers. A network scan is one of the most common techniques used to identify services that might then be accessed without permission [18].

The remaining sections of this study are structured as follows: section 2 introduces the proposed testing technique. While the applied classifiers are described in section 3, experimental results are presented in section 4. The conclusions of this study are discussed in section 5, as well as future work.

2 A Mutation Testing Technique for IDSs

Testing an ID tool is the only way to establish its effectiveness. This paper focuses on checking the performance of IDSs when confronting with unknown anomalous situations.

Misuse IDSs based on signatures rely on models of known attacks. The effectiveness of these IDSs depends on the "goodness" of their models. This is to say, if a model of an attack does not cover all the possible modifications, the performance of the IDS will be greatly impaired.

The proposed mutation testing model was previously applied to a visualization-based IDS [16], [17] and is based on mutating attack traffic. In general, a mutation can be defined as a random change. In keeping with this idea, the testing model modifies different features of the numerical information extracted from the packet headers.

The modifications created by this model may involve changes in aspects such as: attack length (amount of time that each attack lasts), packet density (number of packets per time unit), attack density (number of attacks per time unit) and time intervals between attacks. The mutations can also concern both source and destination ports, varying between the different three ranges of TCP/UDP port numbers: well known (from 0 to 1023), registered (from 1024 to 49151) and dynamic and/or private (from 49152 to 65535).

Time is another fascinating issue of great importance when considering intrusions since the chance of detecting an attack increases in relation to its duration. There are therefore two main strategies:

– Drastically reduce the time used to perform a scan.
– Spread the packets out over time, which is to say, reduce the number of packets sent per time unit that are likely to slip by unnoticed.

In this study, the mutations are applied to data related to network scans. It should be taken into account that any of the possible mutations may be meaningless such as a sweep of less than 5 hosts in the case of a network scan.

Changes can be made to attack packets taking the following issues into account:

– Number of scans in the attack (that is, number of addressed port numbers).
– Destination port numbers at which scans are aimed.
– Time intervals when scans are performed.
– Number of packets (density) forming the scans (number of scanned hosts).

3 Classifiers and Ensembles

As previously explained, one of the most interesting features of IDSs would be their capability to automatically detect whether a portion of the traffic circulating the network is an attack or normal traffic. Automated learning techniques are algorithms designed specifically for the purpose of deciding about new presented data.

Usually that kind of algorithms suffer from common problems, such as the overfitting to the data used for training - and therefore, poor generalization capabilities -, the stuck on local minima in their learning function or a high computational complexity when dealing with complex data. One of the most widespread and useful techniques in order to avoid such problems is the ensemble learning scheme [19], [20]. The main idea behind this kind of meta-algorithms is to train several slightly different simpler classifiers and combine their results in order to improve the results obtained by a single, usually more complex, one [21].

In the present study several of these algorithms have been considered both for the base classifiers and for the ensemble training in order to have a significant wide array of possible algorithms to compare their performance results on mutated data sets.

Among the base classifiers, it should be mentioned clustering algorithms such as the k-Nearest Neighbours (IBK) [22], instance-based statistical classification algorithms such as the Simple Classification and Regression Decision Tree (CART) [23] and the REP-Tree [24] and artificial neural-network such as the Radial Basis Function Network [25].

Among the ensemble meta-algorithms that make use of the previous mentioned simple algorithms, the test performed has made use of basic algorithms such as the MultiClass Classifier [26], used to adapt binary classifiers to multi-class problems, Bagging [27], Adaptative Boosting (AdaBoost) [28], or Random Forest [29] and compared their results with more modern boosting algorithms such as the LogitBoost

[30] or the StackingC [31]. As results prove, ensemble learning adds an important value to the analysis, as almost all variants consistently improve results obtained by the single classifier.

4 Experimental Results

This section describes the datasets used for evaluating the proposed testing method and how they were generated. Then, the obtained results are also detailed.

4.1 Datasets

Real-life datasets have been previously applied to perform ID [16], [17], it has been proved that this low-dimensional datasets allow the detection of some anomalous situations [11]. Packets travelling along the network are characterized by using a set of features, extracted from the packet headers contribute to build up the neural-network input vector, $\mathbf{x} \in \Re^5$; these features can be listed as follows:

- Timestamp: the time the packet was sent.
- Source port: the port number of the device that sent the packet.
- Destination port: the port number of the target host, i.e. the host to which the packet is sent.
- Protocol ID: an integer number that identifies the protocol over TCP of the packet.
- Size: the packet size (in Bytes).

From these datasets, one of them was selected as the original one that was later mutated. It contained examples of two different attacks:

- Three different network scans aimed at port numbers 161, 162 and 3750. A time difference between the first and the last packet included in each sweep of 17 866 ms for port number 161, 22773 ms for port number 162 and 17755 ms for port number 3750.
- An MIB (Management Information Base) information transfer event. This anomalous situation and its potential risks are fully described in [17], [32].

On the basis of this original dataset, several mutations were generated according to the testing technique previously described.

Several testing data sets containing the following key features were obtained by mutating different characteristics of the original data:

- Case 1 (modifying both the amount of scans and the destination ports):
  - Data set 1.- only one scan: port 3750.
  - Data set 2.- two scans: ports 161 and 162.
  - Data set 3.- only one scan: port 1734.
  - Data set 4.- two scans: ports 4427 and 4439.
- Case 2 (modifying both time and the number of scans):
  - Data set 5.- three time-expanded scans: ports 161, 162 and 3750.
  - Data set 6.- three time-contracted scans: ports 161, 162 and 3750.
  - Data set 7.- one time-expanded scan: port 3750.

– Case 3 (modifying both the amount of packets and the destination ports):
  - Data set 8.- two 5-packet scans: ports 4427 and 4439.
  - Data set 9.- two 30-packet scans: ports 1434 and 65788.

The first issue to consider is the amount of scans. Data sets containing 1 scan (Data sets 1, 3 and 7), 2 sweeps (Data sets 2, 4, 8 and 9) or 3 sweeps (Data sets 5 and 6) have been used. Each scan is aimed at a different port number. The implications are crystal clear; hackers can check the vulnerability of as many services/protocols as they want.

A scan attempting to check port protocol/service can be aimed at any port number (from 0 to 65535). The data sets contain scans aimed at port numbers such as 161 and 162 (well known ports assigned to Simple Network Management Protocol), 1434 (registered port assigned to Microsoft-SQL-Monitor, the target of the W32.SQLExp.Worm), 3750 (registered port assigned to CBOS/IP ncapsalation), 4427 and 4439 (registered ports, as yet unassigned) and 65,788 (dynamic or private port).

In order to check the performance of the described ensembles in relation to the time-related strategies, data sets 5, 6 and 7 have been used. Data set 5 was obtained by spreading the packets contained in the three different scans (161, 162 and 3750) over the captured session. In this data set, there is a time difference of 247360 ms between the first (in the sweep aimed at port 161) and the last scan packet (in the scan aimed at port 3750). The duration of the captured session (all the packets contained in the data set) is 262198 ms, whereas in the original data set the scan lasts 164907 ms. In the case of data set 7, the same mutation has been performed but only for packets relating to the scan aimed at port 3750. On the other hand, the strategy of reducing the time was used to obtain data set 6. In this case, the time difference between the first and the last packet is about 109938 ms.

Finally, the number of packets contained in each scan was also considered. In the case of a network scan, each packet means a different host targeted by the scan. Data sets 8 and 9 were designed in complying with this idea. Data set 8 contains low-density scans given that they have been reduced to only 5 packets. It was decided that a scan aimed at less than 5 hosts should not constitute a network scan. On the other hand, data set 9 contains medium-density scans. In this case, each one of them has been extended to 30 packets.

4.2 Results

For the sake of brevity, experiments were conducted only on two of the data sets described above: data set 6 and 9. All the classifiers were trained on the original data-set, comprising an MIB information transfer and scans aimed at port numbers 161, 162 and 3750.

To check the performance of the classifier ensembles when confronting different numbers of classes, the data sets were labeled in a different way. Two different labels were assigned to packets in dataset 6, differentiating attacks from normal traffic. On the other hand, four different classes were defined for data set 9, namely: normal, scan#1, scan#2, and MIB transfer.

WEKA software [33] was used to train and classify the data sets, with various ensembles and classifiers. In the experimentation, cross validation is used with a value of 10 K-fold. Tables 1 and 2 show the training and validation/classification performance of the applied models.

Table 1. Results on data set 6 (two classes).

Ensembles	**Classifier**	**Correctly Classified Instances**
FilteredClassifier	MLP, REPTree, PART, id3, SMO, SMOreg, Winnow, SPegasos	Training (99.983%) Classification (100%)
Adaboost	JRip	Training (99.983%) Classification (100%)
MultiboostAB	JRip	Training (99.983%) Classification (100%)
MultiboostAB	REPTree	Training (99.9489%) Classification (100%)
RandomSubSpace	REPTree	Training (99.983%) Classification (100%)
RandomSubSpace	SImpleCart	Training (99.9659%) Classification (100%)
RotationForest	REPTree and PART	Training (99.983%) for both Classification (100%) for both
AttributeSelectedClassifier	SImpleCart	Training (99.8636 %) Classification (100%)
Bagging	REPTree	Training (99.9659%) Classification (100%)

Table 2. Results on data set 9 (four classes).

Ensembles	**Classifier**	**Correctly Classified Instances**
FilteredClassifier	MLP, REPTree, PART, id3, SMO	Training (99.9489%) for MLP Training (99.8977%) for Reptree Training (99.9489%) for PART Training (99.9148%) for id3 Training (99.9659%) for SMO Classification (81.25%) for all of then. 12 errors in the one of the classes.
Adaboost	JRip	Training (99.9148%) Classification (100%)
MultiboostAB	JRip	Training (99.9148%) Classification (100%)
MultiboostAB	REPTree	Training (99.983%) Classification (100%)
RandomSubSpace	REPTree	Training (99.9659%) Classification (100%)
RandomSubSpace	SImpleCart	Training (99.983%) Classification (100%)
RotationForest	REPTree and PART	Training (99.983%) for RepTree Classification (89.0625%) for RepTree.

		Training (99.983%) for PART Classification (53.125%) for PART.
AttributeSelectedClassifier	SImpleCart	Training (99.983%) Classification (100%)
Bagging	REPTree	Training (99.983%) Classification (100%)

From Tables 1 and 2, it can be concluded that classification models are able to carry out the classification of the two data sets and get the right classification for more than 99.9% of the classes.

Table 3 shows the characteristics and options of the chosen ensembles, together with their tuned values.

Table 3. Selected options of the ensembles for experimental study.

Ensembles	**Options**
FilteredClassifier	Name of the filter "Discretize"
Adaboost	Number of boost iterations (10), seed for resampling (1), use resampling instead of reweighting (false), percentage of weight mass (100).
MultiboostAB	Number of boost iterations (10), number of sub-committees (3), seed for resampling (1), use resampling instead of re-weighting (false), percentage of weight mass (100).
RandomSubSpace	Number of iterations (10), Size of each subSpace (0.5), seed for resampling (1).
RotationForest	Maximum size of a group (3), Minimum size of a group (3), number of iterations to be performed (10), number of groups (false), filter used "Principal Components", percentage of instances to be removed (50), seed for resampling (1).
AttributeSelectedClassifier	Name of an attribute evaluator (CfsSubsetEval), name of a search method (BestFirst).
Bagging	Size of each bag (100), compute out of bag error (False), number of bagging iterations (10), seed for resampling (1).

5 Conclusions and Future Work

This paper has proposed a mutation testing model for classifiers ensembles performing ID on numerical traffic data sets. It is aimed at assessing the generalization capability of the applied classifiers when confronting with zero-day attacks.

Experimental results show that the applied classifier ensembles properly deal with the analyzed data, containing mutated network scans. It can then be concluded that the applied models are able to properly detect new attacks related with previously unseen scans.

Future work will be based on the mutation of some other attack situations and the broadening of considered classifiers and ensembles.

Acknowledgments

This research has been partially supported through the project of the Spanish Ministry of Science and Innovation TIN2010-21272-C02-01 (funded by the European Regional Development Fund). The authors would also like to thank the vehicle interior manufacturer, Grupo Antolin Ingenieria S.A., within the framework of the MAGNO2008 - 1028.- CENIT Project also funded by the MICINN, the Spanish Ministry of Science and Innovation PID 560300-2009-11 and the Junta de Castilla y Len CCTT/10/BU/0002.

References

1. Computer Security Threat Monitoring and Surveillance. Technical Report. James P. Anderson Co, (1980)
2. Denning, D.E.: An Intrusion-Detection Model. IEEE Transactions on Software Engineering 13(2), 222-232 (1987)
3. Chih-Fong, T., Yu-Feng, H., Chia-Ying, L., Wei-Yang, L.: Intrusion Detection by Machine Learning: A Review. Expert Systems with Applications 36(10), 11994-12000 (2009)
4. Abraham, A., Grosan, C., Martin-Vide, C.: Evolutionary Design of Intrusion Detection Programs. International Journal of Network Security 4(3), 328-339 (2007)
5. Julisch, K.: Data Mining for Intrusion Detection: A Critical Review. In: Applications of Data Mining in Computer Security. Kluwer Academic Publishers. Advances in Information Security 33-62 (2002)
6. Giacinto, G., Roli, F., Didaci, L.: Fusion of Multiple Classifiers for Intrusion Detection in Computer Networks. Pattern Recognition Letters 24(12), 1795-1803 (2003)
7. Chebrolu, S., Abraham, A., Thomas, J.P.: Feature Deduction and Ensemble Design of Intrusion Detection Systems. Computers & Security 24(4), 295-307 (2005)
8. Kim, H.K., Im, K.H., Park, S.C.: DSS for Computer Security Incident Response Applying CBR and Collaborative Response. Expert Systems with Applications 37(1), 852-870 (2010)
9. Tajbakhsh, A., Rahmati, M., Mirzaei, A.: Intrusion Detection using Fuzzy Association Rules. Applied Soft Computing 9(2), 462-469 (2009)
10. Sarasamma, S.T., Zhu, Q.M.A., Huff, J.: Hierarchical Kohonenen Net for Anomaly Detection in Network Security. IEEE Transactions on Systems Man and Cybernetics, Part B 35(2), 302-312 (2005)
11. Herrero, Á., Corchado, E., Gastaldo, P., Zunino, R.: Neural Projection Techniques for the Visual Inspection of Network Traffic. Neurocomputing 72(16-18), 3649-3658 (2009)
12. Zhang, C., Jiang, J., Kamel, M.: Intrusion Detection using Hierarchical Neural Networks. Pattern Recognition Letters 26(6), 779-791 (2005)
13. Marchette, D.J.: Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint. Information Science and Statistics Springer-Verlag New York, Inc. (2001)
14. Roesch, M.: Snort–Lightweight Intrusion Detection for Networks. In: 13th Systems Administration Conference (LISA '99). 229-238 (1999)
15. Ranum, M.J.: Experiences Benchmarking Intrusion Detection Systems. NFR Security Technical Publications. (2001)
16. Corchado, E., Herrero, Á., Sáiz, J.M.: Testing CAB-IDS through Mutations: on the Identification of Network Scans. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) International Conference in Knowledge-Based and Intelligent Engineering & Information Systems (KES 2006). LNAI, vol. 4252, pp. 433-441. Springer, Heidelberg (2006)
17. Corchado, E., Herrero, Á.: Neural Visualization of Network Traffic Data for Intrusion Detection. Applied Soft Computing 11(2), 2042–2056 (2011)

18. Abdullah, K., Lee, C., Conti, G., Copeland, J.A.: Visualizing Network Data for Intrusion Detection. In: Sixth Annual IEEE Information Assurance Workshop - Systems, Man and Cybernetics. pp. 100-108 (2005)
19. Sharkey, A.J.C., Sharkey, N.E.: Combining Diverse Neural Nets. Knowledge Engineering Review 12(3), 231-247 (1997)
20. Polikar, R.: Ensemble Based Systems in Decision Making. IEEE Circuits and Systems Magazine 6(3), 21-45 (2006)
21. Ruta, D., Gabrys, B.: An Overview of Classifier Fusion Methods. Computing and Information Systems 7(1), 1-10 (2000)
22. Bailey, T., Jain, A.: A Note on Distance-Weighted k-Nearest Neighbor Rules. IEEE Transactions on Systems, Man and Cybernetics 8(4), 311-313 (1978)
23. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth Inc., Belmont, CA 358 (1984)
24. Zhao, Y., Zhang, Y.: Comparison of Decision Tree Methods for Finding Active Objects. Advances in Space Research 41(12), 1955-1959 (2008)
25. Moody, J., Darken, C.J.: Fast Learning in Networks of Locally-tuned Processing Units. Neural computation 1(2), 281-294 (1989)
26. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing Multiclass to Binary: a Unifying Approach for Margin Classifiers. Journal of Machine Learning Research 1, 113-141 (2001)
27. Breiman, L.: Bagging Predictors. Machine Learning 24(2), 123-140 (1996)
28. Freund, Y., Schapire, R.E.: Experiments with a New Boosting Algorithm. In: International Conference on Machine Learning. pp. 148-156 (1996)
29. Breiman, L.: Random Forests. Machine Learning 45(1), 5-32 (2001)
30. Friedman, J., Hastie, T., Tibshirani, R.: Additive Logistic Regression: a Statistical View of Boosting. The Annals of Statistics 28(2), 337-407 (2000)
31. Seewald, A.K.: How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness. Nineteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., (2002)
32. Corchado, E., Herrero, Á., Sáiz, J.M.: Detecting Compounded Anomalous SNMP Situations Using Cooperative Unsupervised Pattern Recognition. In: Duch, W., Kacprzyk, J., Oja, E., Zadrozny, S. (eds.) 15th International Conference on Artificial Neural Networks (ICANN 2005). LNCS, vol. 3697, pp. 905-910. Springer, Heidelberg (2005)
33. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. ACM SIGKDD Explorations Newsletter 11(1), 10-18 (2009)