Journal of Computer and Communications, 2015, 3, 52-58
Published Online April 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.34006

A Decision Tree Classifier for Intrusion Detection Priority Tagging

Adel Ammar

College of Computer and Information Sciences, Computer Sciences Department, Al-Imam Mohammad Ibn Saud Islamic University, Riyadh, KSA
Email: Adel.Ammar@ccis.imamu.edu.sa

Received 31 March 2015; accepted 14 April 2015; published 16 April 2015

Abstract

Snort rule-checking is one of the most popular forms of Network Intrusion Detection Systems (NIDS). In this article, we show that Snort priorities of true positive traffic (real attacks) can be approximated in real-time, in the context of high speed networks, by a decision tree classifier, using the information of only three easily extracted features (protocol, source port, and destination port), with an accuracy of 99%. Snort issues alert priorities based on its own default set of attack classes (34 classes) that are used by the default set of rules it provides. But the decision tree model is able to predict the priorities without using this default classification. The obtained tagger can provide a useful complement to an anomaly detection intrusion detection system.

Keywords

Intrusion Detection, Network Security, Snort, Machine Learning, Classification, Decision Tree

1. Introduction

Snort is a widely used cross-platform libpcap-based [1] packet sniffer and logger that can be used as a lightweight network intrusion detection system (NIDS). Snort features rule-based logging to perform content pattern matching and detect a variety of suspicious network traffic and probes [2]. Nevertheless, Snort can not comply with the constraints of large throughput in high speed networks, leading to an important drop rate [3]-[5]. The idea is then to approximate the alert priorities issued by Snort, using a fast-executing machine learning model (namely a decision tree classifier), in order to obtain a real-time priority tagger of the suspected traffic, that can be used as a complement to an anomaly detection intrusion detection system.

The priorities issued by Snort have values of 1, 2, 3 or 4. A priority of 1 (high) is the most severe and 4 (very low) is the least severe. Only alerts corresponding to true attacks are selected for training and testing the model, so that the generated model is not merely an approximation of Snort but avoids Snort false alerts. In the selected

ISCX dataset of true attacks, priority 4 (which corresponds to the single class of TCP connections) is inexistent, and priority 2 is extremely rare. So we have exclusively considered priorities 1 (high) and 3 (low).

The main contribution of this paper is to demonstrate that we can approximate Snort priorities of true positive traffic (real attacks) in real-time, in the context of high speed networks, by a decision tree classifier, using the information of a few extracted features (protocol, source port, and destination port), with an accuracy of 99%. The remaining of the paper is organized as follows: in Section 2, we present the ISCX dataset that we use for our simulations. In Section 3, we present the decision tree classifier that we use for classifying the intrusion detection priorities. In Section 4, we detail the obtained results, before concluding in Section 5.

2. ISCX Synthetic Dataset

This dataset [6] is based on the concept of profiling which contain detailed description of intrusions and abstract distribution models for applications, protocols or lower level network entities. They create two classes of profiles α (attempt to describe an attack scenario in an unambiguous manner) and β (encapsulate extracted mathematical distributions or behaviors of certain entities) profiles. This abstract representation of events and behaviors seen in a network allows researchers to generate their various adequate datasets for the purpose of analysis, testing and evaluation of network intrusion detection systems from multiple different angles.

Authors generated through the introduction and utilization of profiles an intrusion detection dataset of one week. Their dataset consists of a set of statistical features captured via the use of single interface on the switch that all traffic are directed to it. The different attack scenario are executed at different periods and each attack composed by 5 steps: information gathering and reconnaissance (passive or active), vulnerability identification and scanning, gaining access and compromising a system, maintaining access and creating backdoors, and covering tracks. The main contribution of the delivered dataset is its inclusion of the captured load and not only description of it which enables several testing and validation of detection mechanisms.

Real traces were analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. The ISCX dataset provides: 1) realistic network and traffic; 2) labeled dataset; 3) total interaction capture; 4) complete capture; and 5) diverse intrusion scenarios. To generate our labeled alert datasets we performed the following steps:

1) Execution of Snort intrusion detection system on all the ISCX dataset traffic traces.
2) Extraction of all packets generating and triggering Snort alerts.
3) Mapping between these packets and their flows in the ISCX dataset since it is a flow based labeled dataset.
4) Labeling of the different alerts based on the ISCX labeled flows.

By the aforementioned steps we constructed a labeled IDS dataset (containing 7779 alerts). For this purpose, we used several tools such as TCPReplay, TCPDump, etc.

In the present study, from the original features, we only considered 5 that can be easily extracted from real-time traffic (protocol, source IP, source port, destination IP, destination port). Taking the IP addresses as input makes the model network-dependent. This implies that the training procedure should be redone before being applied to a new network.

3. Decision Tree Classifier

3.1. Background

The Decision Tree classifier (DT) is one of the possible approaches to multistage decision making (which includes various other techniques, such as table look-up rules, decision table conversion to optimal decision trees, sequential approaches, ...). The basic idea involved in a multistage approach is to break up a complex decision into a union of several simpler decisions. One of the most important features of DTs is their capability to break down a complex decision-making process into a collection of simpler decisions, thus providing a solution which is often easier to interpret [7].

Decision Trees are used for classification of both nominal and numerical data, which makes it suitable for the available inputs, in our case, which are all nominal (even though they are represented by numbers, these numbers are symbols and do not quantitative values). A decision tree is a tree-like model. It is more like an inverted tree because it has its root at the top and it grows downwards. This representation of the data has the advantage compared with other approaches of being meaningful and easy to interpret. The goal is to create a classification model that predicts the value of the target attribute (attack priority) based on several input attributes. Each interior node of the tree corresponds to one of the input attributes. The number of edges of a nominal interior node is equal to the number of possible values of the corresponding input attribute, while outgoing edges of numerical attributes are labeled with disjoint ranges. Each leaf node represents a value of the label attribute given the values of the input attributes represented by the path from the root to the leaf [8].

In single stage classifiers, only one subset of features is used for discriminating among all classes. This feature subset is usually selected by a globally optimal criterion, such as maximum average interclass separability. In decision tree classifiers, on the other hand, one has the flexibility of choosing different subsets of features at different internal nodes of the tree such that the feature subset chosen optimally discriminates among the classes in that node. This flexibility may actually provide performance improvement over a single-stage classifier. Besides, in a decision tree, global complex decision regions (especially in high-dimensional spaces) can be approximated by the union of simpler local decision regions at successive levels of the tree [7] [9]. The choice of a decision tree model is also justified by the real-time constraints, in the context of high speed networks. In fact, during the operational phase (once the training process is finished), the decision issued by a DT is very fast since it is based on a limited number of nested simple conditional statements.

3.2. Training Procedure

Decision trees are generated by recursive partitioning, *i.e*. repeated splitting on the values of attributes. The attribute is selected for splitting depending upon a selection criterion (e.g. information gain [10] [11], gain ratio [12], gini index [13], accuracy). The basic idea in choosing any splitting criteria at an internal node is to make the data in the son nodes "purer" (*i.e*. belonging to a single class). In general, the recursion stops when all the training instances have the same label value, *i.e*. the subset is pure. Or recursion may stop if most of the instances are of the same label value. This is a generalization of the first approach; with some error threshold. Moreover, there are other halting conditions such as [8]:

- There are less than a certain number of instances in the current subtree.
- No attribute reaches a certain threshold.
- A predefined maximal depth is reached.

A decision tree is trained on ISCX labeled dataset, using Rapidminer software [8]. Only alerts corresponding to true attacks (2873 alerts) are selected for training and testing the model, so that we avoid Snort false alerts.

Originally, the priorities issued by Snort have values of 1, 2, 3 or 4. A priority of 1 (high) is the most severe and 4 (very low) is the least severe. But in the selected true attacks, priority 4 (which corresponds to the single class of TCP connections) is inexistent, and priority 2 is extremely rare (only 6 alerts). So we have exclusively considered priorities 1 and 3. Snort issues alert priorities based on its own default set of attack classes (34 classes, see **Table A1**) that are used by the default set of rules it provides.

We randomly subdivided the 2873 Snort true alerts into 3 datasets:

- Learning dataset: 20% (573 samples). The learning dataset is used to train the decision tree, through the recursive partitioning procedure described above.
- Single-split validation dataset: 20% (573 samples), used for validating the learning, in order to avoid overfitting (fitting to noise). The validation database does not participate. After the training, the performance on the validation dataset is used to select the best hyper-parameters (splitting criterion, maximal depth, minimal leaf size, etc.).
- Testing dataset: 60% (1720 samples). These samples do not participate to the training process, in any way. They are only used for the final performance assessment. Thus, they give a closest insight into the real performance and generalization capability of the model on new unseen data.

We used 5 input features to predict the priority, that we extracted from the ISCX dataset:

1) Protocol;
2) Source IP;
3) Source port;
4) Destination IP;
5) Destination port.

4. Results

The experiments were conducted using Rapidminer 5 software, running on a laptop with Intel core i7-3630QM processor (2.4 Ghz), 16 GB RAM.

Different splitting criteria have been tested (information gain, gain ratio, gini index, accuracy). They all gave similar results on the validation dataset. In fact, none of the proposed splitting criteria in the literature has proved to be universally better than the rest, and they all yield similar results, although their complexity may vary significantly [14] [15].

The decision tree depicted in **Figure 1** resulted from a training using gain ratio as a splitting criterion. We notice that neither IP source nor IP destination features have been retained. This is a positive point that makes the model network-independent. The DT yields an overall accuracy of 99%, using the information from only three features (protocol, source port, and destination port), and with a maximal depth of only 4, which makes its decision on new traffic very fast.

Once the decision tree has been trained on the learning database, the classification model can be applied on unseen data sets for the prediction of the label attribute. At each node, a decision is made and only one path is traversed from the root of the tree to a terminal node (leaf). **Table 1** shows the confusion matrix over the 573

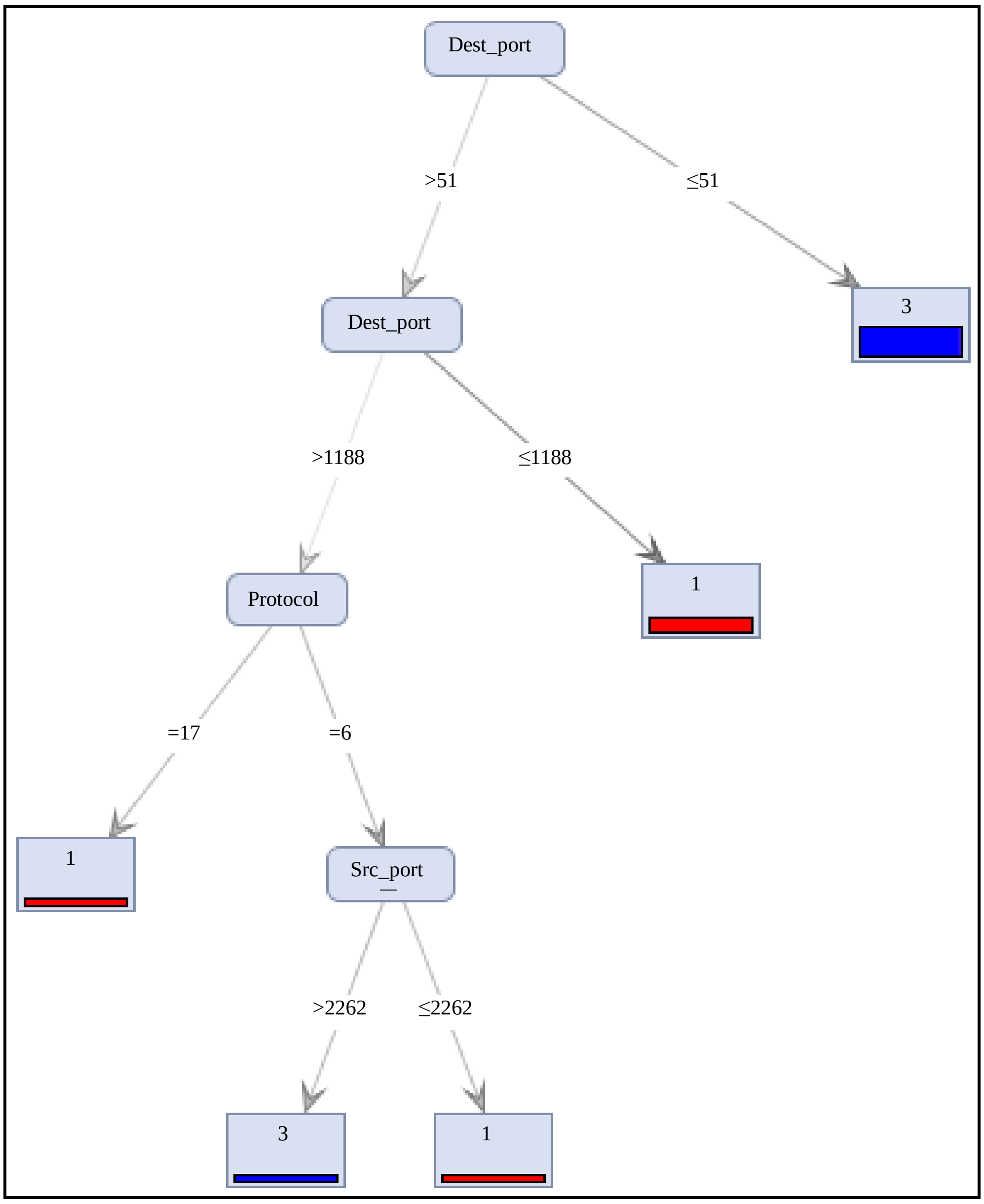

Figure 1. Decision tree for detecting attack priorities.

Table 1. Confusion matrix over the validation database.

	True (Snort) priority = 3	True (Snort) priority = 1	Class precision
Predicted (NN) priority = 3	423	2	99.53%
Predicted (NN) priority = 1	2	146	98.65%
class recall	99.5%	98.7%	

Table 2. Confusion matrix over the testing database.

	True (Snort) priority = 3	True (Snort) priority = 1	Class precision
Predicted (NN) priority = 3	1277	5	99.6%
Predicted (NN) priority = 1	12	426	97.3%
class recall	99.1%	98.8%	

samples of the validation dataset. We obtain an accuracy of 99.3%. The ratio of misclassified samples of (true) class 1 (1.3%) is slightly larger than the number of misclassified samples of (true) class 3 (0.5%).

After the learning phase, we applied the trained decision tree to the independent testing database. This larger dataset (60% of the original) neither participated to the training, nor to the tuning of the hyper-parameters. **Table 2** shows the performance obtained on the testing database. We obtain an accuracy of 99.0%. The ratio of misclassified samples of (true) class 1 is 1.2%, while the number of misclassified samples of (true) class 3 is 0.9%. In fact, the two types of misclassification were given the same cost in the learning process, but since the classes are imbalanced (class 3 being three times more frequent than class 1), the classifier tends to better classify the more frequent class. This is a well known problem for machine learning classifiers, encountered in various domains [16]-[18], and can be addressed correctly. If one considers, for instance, that classifying a sample of class 1 (high priority) as class 3 (low priority) is costlier than the contrary, then this could be taken into account in the training procedure by over-representing class 1 (or under-representing class 3) in the learning database.

5. Conclusion

We have presented a decision tree classifier that approximates the priorities issued by Snort, with an accuracy of 99%, while avoiding its false alerts, and being able to run in real-time, in the context of high speed networks. Future work includes generating a larger learning database of labeled true attacks, in order to include the rare priority values that have been skipped in this work: medium priority (2), and very low priority (4).

References

[1] McCanne, S., Leres, C. and Jacobson, V. (1994) Libpcap, Lawrence Berkeley National Labs.

[2] Roesch, M. (1999) Snort: Lightweight Intrusion Detection for Networks. *LISA*, **99**, 229-238.

[3] Lo, O., Graves, J. and Buchanan, W. (2010) Towards a Framework for the Generation of Enhanced Attack/Background Network Traffic for Evaluation of Network-Based Intrusion Detection Systems. *Proceedings of the* 9*th European Conference on Information Warfare and Security*, Academic Conferences Limited, 190.

[4] Day, D. and Burns, B. (2011) A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines. *ICDS* 2011, *The* 5*th International Conference on Digital Society*, 187-192.

[5] Albin, E. and Rowe, N.C. (2012) A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems. 26*th International Conference on Advanced Information Networking and Applications Workshops* (*WAINA*), 122-127. http://dx.doi.org/10.1109/WAINA.2012.29

[6] Shiravi, A., Shiravi, H., Tavallaee, M. and Ghorbani, A.A. (2012) Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection. *Computers & Security*, **31**, 357-374. http://dx.doi.org/10.1016/j.cose.2011.12.012

[7] Safavian, S.R. and Landgrebe, D.A. (1991) Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems*, *Man and Cybernetics*, **21**, 660-674. http://dx.doi.org/10.1109/21.97458

[8] Akthar, F. and Hahne, C. (2012) Rapid Miner 5 Operator Reference.

[9] Murthy, S.K. (1998) Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. *Data Mining and Knowledge Discovery*, **24**, 345-389. http://dx.doi.org/10.1023/A:1009744630224

[10] Berzal, F., Cubero, J.C., Cuenca, F. and Martı́n-Bautista, M.J. (2003) On the Quest for Easy-to-Understand Splitting Rules. *Data & Knowledge Engineering*, **44**, 31-48. http://dx.doi.org/10.1016/S0169-023X(02)00062-9

[11] Quinlan, J.R. (1986) Induction of Decision Trees. *Machine Learning*, **1**, 81-106. http://dx.doi.org/10.1007/BF00116251

[12] Quinlan, J.R. (2014) C4.5: Programs for Machine Learning. Elsevier.

[13] Olshen, L.B.J.F.R. and Stone, C.J. (1984) Classification and Regression Trees. *Wadsworth International Group*, **93**, 101.

[14] Berzal, F., Cubero, J.C., Cuenca, F. and Martı́n-Bautista, M.J. (2003) On the Quest for Easy-to-Understand Splitting Rules. *Data & Knowledge Engineering*, **44**, 31-48. http://dx.doi.org/10.1016/S0169-023X(02)00062-9

[15] Martin, J.K. (1997) An Exact Probability Metric for Decision Tree Splitting and Stopping. *Machine Learning*, **28**, 257-291. http://dx.doi.org/10.1023/A:1007367629006

[16] Kubat, M., Holte, R.C. and Matwin, S. (1998) Machine Learning for the Detection of Oil Spills in Satellite Radar Images. *Machine Learning*, **30**, 195-215. http://dx.doi.org/10.1023/A:1007452223027

[17] Fawcett, T. and Provost, F.J. (1996) Combining Data Mining and Machine Learning for Effective User Profiling. *KDD*, 8-13.

[18] Zhou, Z.H. and Liu, X.Y. (2006) Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem. *IEEE Transactions on Knowledge and Data Engineering*, **18**, 63-77. http://dx.doi.org/10.1109/TKDE.2006.17

Appendix

Table A1. Snort default classifications.

Class type	Description	Priority
Attempted-admin	Attempted administrator privilege gain	1
Attempted-user	Attempted user privilege gain	1
Inappropriate-content	Inappropriate content was detected	1
Policy-violation	Potential corporate privacy violation	1
Shellcode-detect	Executable code was detected	1
Successful-admin	Successful administrator privilege gain	1
Successful-user	Successful user privilege gain	1
Trojan-activity	A network trojan was detected	1
Unsuccessful-user	Unsuccessful user privilege gain	1
Web-application-attack	Web application attack	1
Attempted-dos	Attempted denial of service	2
Attempted-recon	Attempted information leak	2
Bad-unknown	Potentially bad traffic	2
Default-login-attempt	Attempt to login by a default username and password	2
Denial-of-service	Detection of a denial of service attack	2
Misc-attack	Misc attack	2
Non-standard-protocol	Detection of a non-standard protocol or event	2
Rpc-portmap-decode	Decode of an RPC query	2
Successful-dos	Denial of service	2
Successful-recon-largescale	Large scale information leak	2
Successful-recon-limited	Information leak	2
Suspicious-filename-detect	A suspicious filename was detected	2
Suspicious-login	An attempted login using a suspicious username was detected	2
System-call-detect	A system call was detected	2
Unusual-client-port-connection	A client was using an unusual port	2
Web-application-activity	Access to a potentially vulnerable web application	2
Icmp-event	Generic ICMP event	3
Misc-activity	Misc activity	3
Network-scan	Detection of a network scan	3
Not-suspicious	Not suspicious traffic	3
Protocol-command-decode	Generic protocol command decode	3
String-detect	A suspicious string was detected	3
Unknown	Unknown traffic	3
Tcp-connection	A TCP connection was detected	4