Evaluating Pattern Recognition Techniques in Intrusion Detection Systems*

M. Esposito[1,2], C. Mazzariello[1], F. Oliviero[1], S.P. Romano[1] and C. Sansone[1]

[1] University of Napoli Federico II
Dipartimento di Informatica e Sistemistica
Via Claudio 21 — 80125 Napoli, Italy

[2] CRIAI, Consorzio Campano di Ricerca per l'Informatica e l'Automazione Industriale
Piazzale E. Fermi 1 — 80055 Portici (Napoli) Italy

Abstract. Pattern recognition is the discipline studying the design and operation of systems capable to recognize patterns with specific properties in data sources. Intrusion detection, on the other hand, is in charge of identifying anomalous activities by analyzing a data source, be it the logs of an operating system or in the network traffic. It is easy to find similarities between such research fields, and it is straightforward to think of a way to combine them. As to the descriptions above, we can imagine an Intrusion Detection System (IDS) using techniques proper of the pattern recognition field in order to discover an attack pattern within the network traffic. What we propose in this work is such a system, which exploits the results of research in the field of data mining, in order to discover potential attacks. The paper also presents some experimental results dealing with performance of our system in a real-world operational scenario.

1 Introduction

Security of computer networks has been the subject of an intensive research activity in the last years. New solutions and techniques have been proposed to tackle the security issue. Firewalls and Intrusion Detection Systems (IDS) are the most well known tools which can be employed to protect the network from malicious activities. Indeed, firewalls are used to *prevent intrusions from happening*, whereas IDS *detect an intrusion while it is happening*. In particular, in order to accomplish its task, an intrusion detection system needs to have a pre-defined set of models, or "patterns", describing the behaviour of both "normal" and "malicious" network users. By monitoring the real traffic on the network, the system computes a current user profile which is compared with a set of pre-defined models in order to detect potential intrusions.

If we look at both normal and anomalous behaviors as patterns, we can use common pattern recognition techniques to find attack instances within the network traffic. IDS

* Research outlined in this paper is partially funded by the Italian "Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR)" in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)"

Esposito M., Mazzariello C., Oliviero F., P. Romano S. and Sansone C. (2005).
Evaluating Pattern Recognition Techniques in Intrusion Detection Systems.
In *Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems*, pages 144-153
DOI: 10.5220/0002575201440153

commonly base their detection ability on a set of attack models, making it both easy and fast to discover well known attacks. Though, they are often unable to detect unknown anomalies: even slight modifications of an attack pattern may result in a missed detection. The easiest techniques to use are based on attack signatures, which are organized in databases and are usually hand-coded by a network administrator. An attack signature is the fingerprint of a specific attack, is statically defined and strictly related to the attack type it is meant to detect. Pattern recognition techniques, instead, proved to have a higher generalization capability.

Based on the above considerations, our work aims to define a framework for extracting high-level knowledge from a large data set by means of pattern recognition techniques in order to discover a set of patterns able to distinguish between normal activities on one side and intrusions on the other. The framework is first presented and then evaluated by means of experiments conducted on real data.

2 Related Work

This work has many liaisons with both *intrusion detection* and *data mining*.

As to the first research field, intrusion detection is the art of detecting inappropriate, incorrect or anomalous activity within a system, be it a single host or a whole network. An Intrusion Detection System (IDS) analyzes a data source and, after preprocessing the input, lets a detection engine decide, based on a set of classification criteria, whether the analyzed input instance is normal or anomalous, given a suitable behavior model. Intrusion Detection Systems can be grouped into three main categories: *Network-based Intrusion Detection Systems* (N-IDS) [1], *Host-based Intrusion Detection Systems* (H-IDS) [2] [3] and *Stack-based Intrusion Detection Systems* (S-IDS) [4]. This classification depends on the information sources analyzed to detect an intrusive activity.

Intrusion Detection Systems can be roughly classified as belonging to two main groups as well, depending on the detection technique employed: *anomaly detection* and *misuse detection* [5]. Both such techniques rely on the existence of a reliable characterization of what is *normal* and what is not, in a particular networking scenario.

The main problem related to both anomaly and misuse detection techniques resides in the encoded models, which define normal or malicious behaviors. Although some recent open source IDS, such as SNORT[3][6] or Bro[4][7], provide mechanisms to write new rules that extend the detection ability of the system, such rules are usually hand-coded by a security administrator. This represents a weakness in the definition of new normal or malicious behaviors. Recently, many research groups have focused on the definition of systems able to automatically build a set of models. Data mining techniques are frequently applied to audit data in order to compute specific behavioral models (MADAM ID [8], ADAM [9]).

Coming to the second related research field, we recall that a data mining algorithm is referred to as the process of extracting specific models from a great amount of stored data [10]. Machine learning or pattern recognition processes are usually exploited in

[3] http://www.snort.org
[4] http://www.bro-ids.org

order to realize this extraction. These processes may be considered as off-line processes. In fact, all the techniques used to build intrusion detection models need a proper set of audit data. The information must be labelled as either “normal” or “attack” in order to define the suitable behavioral models that represent these two different categories. Such audit data are quite complicated to obtain.

We finally mention that this work also entails an analysis of the network traffic aimed at defining a comprehensive set of so-called *connection features*. Such a process requires that an ad-hoc classifier is defined and implemented. The greater the capability of the set of features to discriminate among different categories, the better the classifier. Many researchers have been working on the topic in the last few years.

In particular, we have adopted a model descending from the one proposed by Stolfo et al., who propose a set of connection features which can be classified in tree main groups: *intrinsic* features, *content* features, and *traffic* features. Intrinsic features specify general information on the current session, like the duration in seconds of the connection, the protocol type, the port number (i.e. the service), the number of bytes from the source to the destination, etc..

The content features are related to the semantic content of connection payload: for example, they specify the number of failed login attempts, or the number of shell prompts.

Finally, the traffic features can be divided in two groups: the *same host* and the *same service* features. The same host features examine all the connections in the last two seconds to the same destination host of the current connection, in particular the number of such connections, or the rate of connections that have a “SYN” error. Instead, the same service features examine all the connections in the last two seconds to the same destination service of the current one.

3 Rationale and Motivation

One of the main issues related to pattern recognition in intrusion detection is the use of a proper data set, containing user profiles on which the data mining processes work in order to extract the patterns. In principle, an efficient set of patterns for the detection has to contain all of the possible user behaviors. Moreover, according to all pattern recognition processes, the data set has to properly label the behavior profile items with either “normal” or “attack”. Although this might look like an easy task, labelling the data imposes a pre-classification process: you have to know exactly which profile is “normal” and which is not.

In order to solve the issue related to data set building, two main approaches are possible: the former relies on simulating a real-world network scenario, the latter builds the set using actual traffic.

The first approach is usually adopted when applying pattern recognition techniques to intrusion detection. The most well-known dataset is the so-called KDD Cup 1999 Data, which was created for the Third International Knowledge Discovery and Data Mining Tools Competition, held within KDD-99, The Fifth International Conference on Knowledge Discovery and Data Mining[5] that was created by the Lincoln Laboratory

[5] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

at MIT in order to conduct a comparative evaluation of intrusion detection systems, developed under DARPA (Defense Advanced Research Projects Agency) and AFRL (Air Force Research Laboratory) sponsorship[6].

This set was created in order to evaluate the ability of data mining algorithms to build predictive models able to distinguish between a normal behavior and a malicious one. The KDD Cup 1999 Data contains a set of "connection" records coming out from the elaboration of raw TCPdump data. Each connection is labelled as either "normal" or "attack". The connection records are built from a set of higher-level connection features, defined by Stolfo et al. [11], that are able to tell apart normal activities from illegal network activities.

Although widely employed, some criticisms have been raised against the 1999 KDD Cup Data [12]. Indeed, numerous research works analyze the difficulties arising when trying to reproduce actual network traffic patterns by means of simulation [13]. Actually, the major issue resides in effectively reproducing the behavior of network traffic sources.

Based on the considerations above, we have concluded that the KDD Cup 1999 Data can just be used to evaluate the effectiveness of the pattern recognition algorithms under study, rather than in the real application of intrusion detection.

Collecting real traffic can be considered as a viable alternative approach for the construction of the traffic data set [14]. Although it can prove effective in real-time intrusion detection, it still presents some concerns. In particular, collecting the data set by means of real traffic needs a data pre-classification process. In fact, as stated before the pattern recognition process needs a data set in which packets are labelled as either "normal" or "attack". Indeed, no information is available in the real traffic to distinguish the normal activities from the malicious ones in order to label the data set. So we have a paradox: *we need pre-classified traffic in order to extract the models able to classify the traffic*. Last but not least, the issue of privacy of the information contained in the real network data has to be considered: payload anonymizers and IP address spoofing tools are needed in order to preserve sensitive information.

This work aims to develop a real time intrusion detection system based on pattern recognition techniques. We have adopted the real traffic collection approach to extract the network behavior models. We will present in the paper a method to: (i) collect real data from a network; (ii) elaborate such information in order to build and appropriately label the associated data set. We also provide some figures about the effectiveness of the method proposed.

4 A feature-based framework for detecting attacks

The first issue that a real time intrusion detection system has to face is the computation of an up-to-date user behavior profile every time a new event occurs on the network. In particular, the set of features characterizing the current traffic profile has to be determined every time a new packet is captured from the network. Indeed, issues related to real time features computation have been usually neglected by previous research on

[6] http://www.ll.mit.edu/IST/ideval

pattern recognition applied to intrusion detection. For example, few works have dealt with the packet loss issue: if the profile extraction time is longer than the interarrival time (due to either a low computation speed or a high traffic rate), some packets may be lost, at the detriment of the detection ability.

In our previous work [15], we have evaluated the feasibility of a real time intrusion detection system. The system we developed is available at the SourceForge[7] site. In this section we present a different contribution, dealing with an approach to the off-line extraction of models which can be profitably exploited in the real time system.

As stated before, we need a proper data set on which the pattern recognition algorithm works in order to extract the "detection patterns" needed for the real time classification process. With our approach, we collect real traffic traces. We deem that such approach represents a desirable solution in case the computed patterns have to be applied in an actual operational scenario (see section 3). Our data set has been built by collecting real traffic on the local network at Genova National Research Council (CNR).

The *raw traffic* data set contains about one million packets, equivalent to $1GByte$ of data. The network traffic has been captured by means of the TCPdump tool and logged to a file. In order to solve the pre-classification problem (which, as already stated, requires labelling the items in the data set), we have used a previous work of Genova's research team. By using two different intrusion detection systems, researchers in Genova have analyzed the generated alert files and manually identified, in the logged traffic, a set of known intrusions. We have leveraged the results of this research in order to extract the connection features record and properly label it with either a *normal* or an *attack* tag, as it will be clarified in section 5.

After building the data set, we have focused on the management of the data in order to realize the pattern recognition process. Every record in the data set is composed of 26 connection features, namely Stolfo's "intrinsic" and "traffic" features. Indeed, just few features can be used to tell apart normal from anomalous traffic in the analyzed network scenario. In fact, some attacks can be classified only with a small set of connection features. This can be considered as an advantage: we can reduce the dimensional space of the data set, letting the pattern recognition process become simpler. Common to all the data mining processes, the issue of feature subset selection is known as *feature selection problem*.

In our context, we have adopted ToolDiag[8], a pattern recognition toolbox, in order to realize the feature selection. As a selection strategy we adopted Sequential Forward Selection with the Estimated Minimum Error Probability Criterion.

The last step in our work has concerned the extraction of network behavior patterns from the data set.

By using Stolfo's connection features — which cover a wide range of attack types — it is possible characterize the attacks by means of a set of rules. Supposing that the traffic data item can be represented in a vectorial space, a data mining process partitions such a space in a *normal* region and an *attack* region, based on the rule set; if the vector of features related to the current packet belongs to this space, an intrusive action is

[7] http://sourceforge.net/projects/s-predator
[8] http://www.inf.ufes.br/ thomas/home/tooldiag.html

detected. In this way the rule is not referred to a single attack; it is rather used in a more complex classification process.

In order to extract the set of rules from the data set, we have adopted the SLIPPER[9] [16] tool. SLIPPER is a rule-learning system exploiting the Boosting technique [17].

5 Experimental results

In this section we present some experimental results concerning the attack detection capabilities attained by using the proposed approach. We will mainly focus on the missed detection rate and, more important, on the false alarm rate, which is a critical requirement for an effective intrusion detection system [18]. Though in other pattern recognition applications a false positive rate below 5% may be a very satisfactory value, in intrusion detection such a rate may not be acceptable. For example, if we imagine to work on a network with a packet rate of 1000000 packets per hour, a false alarm rate of 0.1% would lead to 1000 annoying alert messages sent to the administrator every hour: though characterized by a very low false alarm rate, the number of unjustified alerts would be too high and would lead the administrator to ignore or eventually switch the intrusion detection system off.

Table 1. Detection accuracy after feature selection

	Train Error Rate	**Test Error Rate**	**Hypothesis Size**	**Learning Time**
Test 1	0.25%	0.35%	9 Rules, 29 Conditions	196.14s
Test 2	0.18%	0.31%	12 Rules, 46 Conditions	211.78s
Test 3	0.22%	0.28%	9 Rules, 34 Conditions	202.08s
Test 4	0.23%	0.26%	9 Rules, 31 Conditions	182.32s
Test 5	0.21%	0.32%	9 Rules, 41 Conditions	264.87s
Test 6	0.20%	0.35%	9 Rules, 31 Conditions	222.76s
Test 7	0.20%	0.31%	9 Rules, 31 Conditions	202.62s
Test 8	0.15%	0.29%	13 Rules, 45 Conditions	243.16s
Test 9	0.20%	0.30%	10 Rules, 29 Conditions	233.16s
Test 10	0.24%	0.31%	24 Rules, 85 Conditions	244.37s
Test 11	0.17%	1.38%	10 Rules, 38 Conditions	198.12s
Test 12	0.25%	0.32%	9 Rules, 31 Conditions	225.62s
Test 13	0.19%	0.29%	13 Rules, 40 Conditions	195.63
Test 14	0.17%	0.32%	7 Rules, 24 Conditions	188.12s
Test 15	0.21%	0.30%	11 Rules, 43 Conditions	223.65s
Test 16	0.23%	0.28%	4 Rules, 9 Conditions	186.62s
Test 17	0.21%	0.28%	7 Rules, 26 Conditions	246.65s
Test 18	0.17%	0.18%	14 Rules, 59 Conditions	244.29s

We ran different tests on the huge amount of data collected at the CNR laboratories in Genova (Italy). As stated before, we have a 1000000 packets log.

[9] http://www-2.cs.cmu.edu/ wcohen/slipper/

Table 2. Detection accuracy after feature selection – Average values

Train Error Rate	Test Error Rate	Hypothesis Size	Learning Time
0.20%	0.36%	10 Rules, 37 Conditions	217.33s

Table 3. Detection accuracy after filtering and feature selection

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	33.59%	0.06%
2nd Half	**1st Half**	50.41%	0.03%

Table 4. Detection accuracy without feature selection

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	13.57%	0.16%
2nd Half	**1st Half**	55.32%	0.07%

Table 5. Detection accuracy after filtering without feature selection

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	13.79%	0.16%
2nd Half	**1st Half**	62.19%	0.05%

Table 6. Detection accuracy without feature selection – Trin00 attack

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	4%	0%
2nd Half	**1st Half**	0%	0%

Table 7. Detection accuracy after filtering and without feature selection – Trin00 attack

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	0%	0%
2nd Half	**1st Half**	0%	0%

Table 8. Detection accuracy without feature selection – Scan SOCKS attack

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	5.91%	0%
2nd Half	**1st Half**	38.98%	0%

Table 9. Detection accuracy after filtering and without feature selection – Scan SOCKS attack

Training Set	Test Set	Missed Detections	False Alarms
1st Half	**2nd Half**	6.54%	0%
2nd Half	**1st Half**	48.30%	0%

First of all, we decided to subsample the data by a factor of $1/10$ in order to reduce the computation time of the results; as stated before, we use ToolDiag for the feature selection step and SLIPPER for the classification. In the first experiment we first subsample the data-set by choosing one connection record out of ten, then we split the subsets in two parts. On each of the half-subset obtained we perform feature selection and, by examining the discriminating power and the number of occurrences over the whole data set of the selected features, we choose an "optimum" set of 8 features out of the 26 features available. By "optimum" feature, we mean a feature whose ability to discriminate between attacks and normal traffic, within the training data, is the highest with respect to the discriminating power of all the examined features. We consider then, in turn and for each subset, the first half as the training set, and the second half as the test set; then we swap training and test sets, using the second half of each subset as the training set and the first half as the test set. All these experiments are useful to understand which is the best data set we have, as we suppose to have no prior knowledge about the discriminating power of the connection records included in each one of them. In table 1 we see the performance attained over the 18 experiments. It is worth to notice that the classification error over the test set is, except for a couple of cases, below 0.50%, and the number of rules and conditions much lower than the number of rules commonly used in SNORT, which is about 1500. Furthermore, the computation times for the classification criteria seems to be very reasonable, if compared with the time required when we don't employ feature selection. In table 2 we point out the average values emerging from the analysis of the presented results.

It is worth pointing out that the data we are working on contain some connection records tagged as *uncertain*. During the data preparation, we decided to label as attacks the connection records corresponding to the packets classified as attacks by both the IDS used at Genova CNR; in case only one of the used tools raised an alert, in this first experiment we decided to label the corresponding packet as normal. It is straightforward, indeed, to have a doubt about this approach: what if the *uncertain* packets were attack packets? Would this affect in a meaningful way the detection capability of the system? We had two chances: we could consider the *uncertain* packets as attacks as well, though this would have led us to a complementary mistake with respect to the one committed so far; we could, as well, simply discard such packets, considering them as belonging to an unknown class of traffic. Thus we built and processed a "filtered out" data set, made up by all the connection records corresponding to packets whose classification was clear enough, obtained by deleting the *uncertain* connection records by the set.

Again we proceeded with feature selection and obtained, in the same way as before, the best set of eight features. On the filtered data we decided to deploy a test by using the whole dataset, with no subsampling. We divided the dataset in two halves and, in Test 1 we considered the first half as the training set, and the second half as the test set; in Test 2, instead, we consider the second half of the data set as the training set and the first half as the test set.

Furthermore, to test the effect of feature selection on the detection capability of the system, we decided not to apply subsampling, and to test the classifier on the datasets before and after the filtering process described above (tables 4, 5). We notice a very low

false alarm rate, which is good, and a missed detection rate sometimes around 60%. This might seem a not so good result, but it is not; missing an attack packet does not mean to miss the whole attack itself; in fact, an attack pattern may consist of a burst of packets thus, not detecting a few of such packets doesn't mean to lose the attack. Stressing again the false alarm rate problem, we notice that the rate obtained within our experiments is very low, and encouraging for the development of this kind of detection techniques.

In order to strengthen these observations, we also sketch, in tables 6-9, the detection capabilities tested over two precise attacks: Trin00 and Scan SOCKS probe. Trin00 is always detected by our IDS, while for Scan SOCKS we miss some attack packets; anyway, as probes are not real attacks, we can consider such results good as well. Probe attacks usually are just the preliminary phase of an attack, thus it is important to detect their occurrence as soon as possible, as our IDS does; once we know a scan is in progress, we can strengthen our defences in order to protect the system from the attack which will likely happen later.

As we have a little lower missed detection rate when not using feature selection, we noticed an increase of one order of magnitude in rule calculation time and number of rules. This is due to the fact that we have to strike the balance between detection accuracy, number of adopted criteria and computation time.

6 Conclusions and Future Work

Intrusion detection system based on pattern recognition techniques definitely represent a very interesting tool to use. We show a very low false alarm rate, which is the most important requirement for an effective IDS. Though the missed detection rate is not as low as the false alarm rate, it is encouraging pointing out that missing a single attack packet does not mean to miss the whole attack itself; particular attack types, like scans or probes, might make the job of an IDS harder. As to the evolving nature of scan attacks, we have to take into account, when evaluating the detection capability of such attacks, the transient which the IDS, like all the systems in nature, is subject to. After a short transient, indeed, also probe attacks are discovered and reported to the administrator. When using pattern recognition in intrusion detection, we have to face the trade off between detection accuracy and resource consumption, where our resource is namely the computation time. In this case, when not using feature selection, we obtain a slightly more precise detection, by producing in a very long time (about 2000 seconds) a huge number of classification criteria (over 100) with an even higher number of conditions. Thus, when designing and configuring such a system, a preliminary phase of trade off evaluation is mandatory.

Of course it will be helpful, in the future, to test the proposed approach over a set of different classes of network traffic, as well as to inject new attacks and evaluate in greater detail the attack prediction capability. Furthermore, it will be the subjetc of our future research the analysis of techniques based on multiple classifiers. By using multiple classification strategies, we can gather the results attained by different classification strategies, thus improving the overall attack detection capability of the system.

References

1. Vigna, G., Kemmerer, R.: Netstat: a network based intrusion detection system. Journal of Computer Security **7** (1999)
2. Andersson, D.: Detecting usual program behavior using the statistical component of the next-generation intrusion detection expert system (nides). Technical report, Computer Science Laboratory (1995)
3. Tyson, M.: Derbi: Diagnosys explanation and recovery from computer break-ins. Technical report (2000)
4. Laing, B., Alderson, J.: How to guide - implementing a network based intrusion detection system. Technical report, Internet Security Systems, Sovereign House, 57/59 Vaster Road, Reading (2000)
5. Bace, R.G.: Intrusion Detection. Macmillan Technical Publishing (2000)
6. Baker, A.R., Caswell, B., Poor, M.: Snort 2.1 Intrusion Detection - Second Edition. Syngress (2004)
7. Paxson, V., Terney, B.: Bro reference manual (2004)
8. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information and System Security (TISSEC) **3** (2000) 227–261
9. Barbara, D., Couto, J., Jajodia, S., Popyack, L., Wu, N.: Adam: Detecting intrusion by data mining, IEEE (2001) 11–16 Workshop on Information Assurance and Security.
10. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery in databases. AI Magazine (1996) 37–52
11. Stolfo, S.J., Fan, W., Lee, W., Prodromidis, A., Chan, P.: Cost-based modeling for fraud and intrusion detection results from the jam project. In: Proceeding of the 2000 DARPA Information Survivability Conferance and Exposition (DISCEX '00). (2000)
12. McHugh, J.: Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Transactions on Information and System Security **3** (2000) 262–294
13. Paxson, V., Floyd, S.: Difficulties in simulating the internet. IEEE/ACM Transactions on Networking **9** (2001) 392–403
14. Mahoney, M.: A Machine Learning Approach to Detecting Attacks by Identifying Anomalies in Network Traffic. PhD thesis, Florida Istitute of Technology (2003)
15. Esposito, M., Mazzariello, C., Oliviero, F., Romano, S.P., Sansone, C.: Real Time Detection of Novel Attacks by Means of Data Mining. In: Proceedings of 2005 ICEIS Conference, ACM (2005)
16. Cohen, W., Singer, Y.: Simple, fast, and effective rule learner. In: Proceedings of the Sixteenth National Conferance an Artificial Intelligence, AAAI Press / The MIT Press (1999) 335 – 342
17. Meir, R., Ratsch, G.: An introdution to boosting and leveraging. In Mendelson, S., Smola, A., eds.: Advanced Lectures an Machine Learning, Springer Verlag (2003) 119 – 184
18. Axelsson, S. In: The base-rate fallacy and the difficulty of intrusion detection. Volume 3 of ACM transaction on information and system security. ACM (2000) 186–205