Evaluation of data mining techniques for suspicious network activity classification using honeypots data

André Grégio[1,2], Rafael Santos[1], Antonio Montes[2]
[1]Brazilian Institute for Space Research, Av. dos Astronauta, 1758, São José dos Campos, SP, Brazil, 12227-010;
[2]Dept. of Information Systems Security, Renato Archer Research Center, Rod. D. Pedro I, Km 143.6, Campinas, SP, Brazil, 13069-901.

ABSTRACT

As the amount and types of remote network services increase, the analysis of their logs has become a very difficult and time consuming task. There are several ways to filter relevant information and provide a reduced log set for analysis, such as whitelisting and intrusion detection tools, but all of them require too much fine-tuning work and human expertise. Nowadays, researchers are evaluating data mining approaches for intrusion detection in network logs, using techniques such as genetic algorithms, neural networks, clustering algorithms, etc. Some of those techniques yield good results, yet requiring a very large number of attributes gathered by network traffic to detect useful information. In this work we apply and evaluate some data mining techniques (K-Nearest Neighbors, Artificial Neural Networks and Decision Trees) in a reduced number of attributes on some log data sets acquired from a real network and a honeypot, in order to classify traffic logs as normal or suspicious. The results obtained allow us to identify unlabeled logs and to describe which attributes were used for the decision. This approach provides a very reduced amount of logs to the network administrator, improving the analysis task and aiding in discovering new kinds of attacks against their networks.

Keywords: Computer Security, Data Mining, Log Analysis, Intrusion Detection, Artificial Intelligence

1. INTRODUCTION

Since the 90's, the Internet has provided plenty of remote data access services. The facilities provided by the Internet modified the way society interacts, making life easier. People can communicate, shop and access their banks virtually. On the other hand, new ways to commit crimes appeared, fueled by the ease of Internet access and low risk for the criminals.

All this network interaction generates data that can (and must) be registered. The register of the traffic information is called logging and consists of the storage of data related to the network communication between the parts involved. This can be useful in fault monitoring or in the identification of events that might be attacks against the services.

The logging activity is apparently simple, but it has various concerns as more services lead to even more logs being generated. These concerns are related to the amount of storage, the utility of the stored logs, the workload of the person who will analyze those logs and the difficulty and time requirements to do that job. Except for storage requirements, which nowadays is not a real problem, all other concerns are human-centric – in other words, the processing, analysis and actions resulting from the analysis must be performed by a human expert, which is inherently slow and costly.

Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2007, edited by Belur V. Dasarathy, Proc. of SPIE Vol. 6570, 657006, (2007) · 0277-786X/07/$18 · doi: 10.1117/12.719023

The main problem related to logging is the great amount of data that has to be analyzed by a network administrator. In addition to the sheer volume of logs being sometimes impossible to analyze, network administrators are often overwhelmed by other activities, which can cause a decrease in the log analysis effectiveness, not allowing the task to be accomplished in a reasonable time and, ultimately, making the implementation of counter measures ineffective to minimize or avoid attacks.

There are some tools and techniques that helps the network administrator in the log analysis task, such as operating system tools to parse text and search for regular expressions, artificial ignorance and intrusion detection systems, which are superficially described as follows:

- Some tools like grep [1] can be used in scripts that search for patterns, in order to filter or find useful information in logs, improving the network administrator efficiency in the analysis job;
- Artificial ignorance, or whitelisting [2], is a technique to filter and order system events eliminating known irrelevant entries in log and isolating the relevant ones. This technique is based on the knowledge of the network administrator, who creates rules to generates lists of events, and is used to substantially decrease the amount of false alerts;
- Intrusion Detection Systems (IDS) are software that analyze network connections either to match known patterns and alert attacks (intrusion detection by abuse), such as SNORT[3], or to verify if the traffic characteristics are under a threshold that indicates normality (intrusion detection by anomaly), such as IDES[4]. These kind of software tends to be difficult to configure, requiring constant tuning and are somewhat limited, i.e., the IDS by abuse can only detects attacks whose signature exists (the known ones) and the IDS by anomaly is subject to a large amount of false-positives, since it is hard to define a normal profile for computer network's behavior.

Despite all efforts in detecting suspicious activities in networks, attacks are very frequent and often don't have specific targets. Vulnerability exploits are embedded in worms and malware (malicious software) to attack and invade as many computers as possible, in order to build networks of compromised machines (botnets) to perform other kind of attacks, nowadays often related to criminal organizations. Attacks coming from botnets can be denial of service against an enterprise or institution, spreading of unsolicited e-mail (spam) or even ways to anonymize the attacker's identity. Data from CERT.br (Brazilian Security Incident Treatment, Response and Study Center) in Figure 1 shows that the number of incidents reported are constantly increasing, and those reports are only possible if incidents are detected. The detection and consequent report of an incident occurs via log analysis, which makes clear the need for quick and precise log analysis.

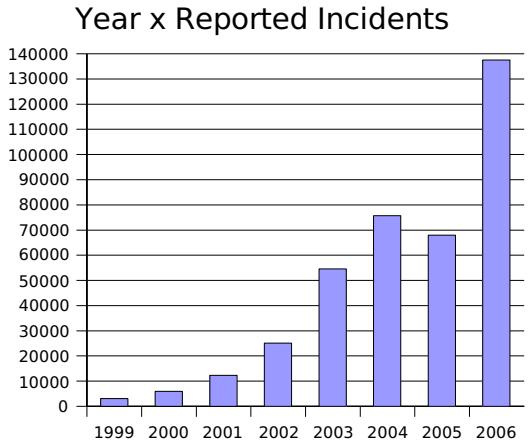

Fig. 1. Total number of incidents reported to Cert.BR per year (Adapted from http://www.cert.br).

In order to improve the intrusion detection task performance, researchers have been applying data mining techniques to log analysis. Data mining is one of the steps in the Knowledge Discovery in Databases (KDD) process. A simple description of KDD is "a general discovery process of useful knowledge (previously unknown) from large databases" [5]. Data mining techniques are used in some intrusion detection tools like MINDS[6], ADAM[7], GASSATA[8] and other.

Data mining requires a fairly large quantity of data to be analyzed and characterized. An useful data-gathering resource that can be used to discover suspicious events in networks are honeypots and honeynets. A honeypot [9] is a security resource whose value lies in being probed, attacked or compromised, while collecting data about the probing and attacks. Honeynets [10] are a specific type of honeypots, called high-interaction honeypots, which allow for greater interaction among the attacker and the computers/services attacked, while collecting data about the interactions. This happens because a honeynet doesn't only emulate a service, but it has indeed a working operating system and offers real services remotely. In Brazil, the Brazilian Honeypots Alliance monitors the national Internet space with more than 30 honeypot partners spread over the country*.

The purpose of this paper is to evaluate some data mining algorithms, applying them to the classification of suspicious network activities in order to verify their effectiveness under certain conditions. Our main concern is to reduce the amount of logs which inevitably will require human attention. We also propose the use of honeypots data to aid in exposing the attack trends to a determined network and to the generation of sets related to non-legitimate data, since we can assume that honeypots logs are all related to non-legitimate access attempts.

The paper is divided as follows: in section 2 we describe the network topology where the logs are collected and the scenario prepared for log analysis. In section 3 the methodology is explained, as well as the data mining algorithms and techniques that were chosen. Section 4 contains the results obtained with the application of these techniques. The fifth section contains conclusions and directions for future work.

2. SCENARIO

Logs can be generated by several sources, such as operating systems, firewalls, specific software, network devices, etc. In this work, we focus on logs from resources that can represent a risk to an institution's networks. Attacks to networks can be carried out by maliciously crafted network traffic, illegal operating system interactions and/or unauthorized access to resources or applications, but we will focus only on traffic information from network traffic logs.

The collected logs are from the Brazilian Institute for Space Research's De-Militarized Network (DMZ). The Institute has many services that are remotely accessible, such as Web servers, a File Transfer Protocol (FTP) server, a Secure Shell (SSH) server, which provides access to the internal network. This services are vital to the institute, since they allow the researchers to access the resources from everywhere and provide weather reports and other information to the community. The logs generated by network access to these services are stored in dump format in a collector for later analysis. There is also a honeypot in the same network that belongs to the Brazilian Distributed Honeypots Project which save its logs in dump format. The topology of the network used in this work is shown in Fig. 2.

*More information about the Brazilian Distributed Honeypots Project can be obtained at http://www.honeypots-alliance.org.br

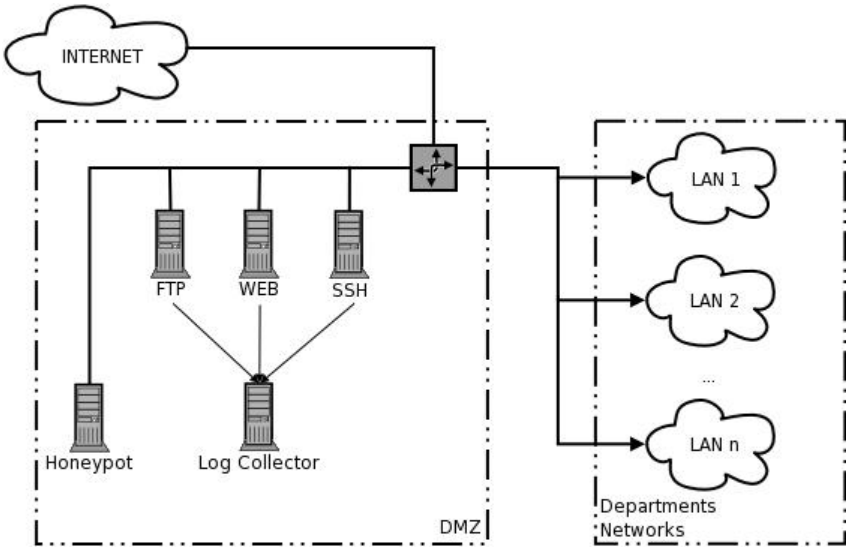

Fig. 2. Topology of Brazilian Institute for Space Research DMZ representing the scenario of this work.

The volume of network traffic stored in the collector approaches 1GB of data for normal days, making the human analysis impossible. A great amount of these logs is associated with legitimate traffic, i.e., normal users accesses or automated scripts that do routine information exchange. Another part of the logs is related to port scanning or malicious scripts trying to exploit or brute-force some protocol, being non-directed attack not directed to Institute's targets. Thus, most of the logs consist basically of noise (legitimate, inoffensive and/or known network traffic), interfering negatively with the analysis. The problem here is that the security analyst must know how to distinguish between interesting traffic and noise, performing the log analysis task as quickly as possible. This pre-analysis or filtering is very dependent on the analyst's expertise and demands a lot of time, even with automated tools to remove noise, so there is need of automated, intelligent techniques to aid the analyst, and data mining techniques seems to be useful.

3. METHODOLOGY

The process of testing the algorithms with the available data consists of many steps, involving the choice of algorithms, the attributes to represent the data, the pre-processing of data to create an input dataset, the application of the algorithm itself and finally, the evaluation of the results. As this work consists of classifying network logs in normal or suspicious, we choose to apply three techniques used in supervised classification tasks: k-nearest neighbors (KNN), neural networks and decision trees. The aim is to find which of them is more adequate to correctly classify our data, providing information about how the classification was done and reducing the false-positive rate, all in an acceptable amount of time.

3.1 Data mining techniques

The k-nearest neighbor hypothesis is that records of the same class will be close to each other in the attribute space[11]. The KNN classification algorithm is as follows: for each instance of data of unknown class we calculate the attribute space distance between it and all instances with known classes, selecting as the class for the unknown one the most present class in the K nearest instances.
Artificial neural networks are distributed or parallel computing structures that are composed by single processing units called neurons[12]. They can be used to solve complex learning tasks, being capable to handle specialized or generalized problems. There are several forms of neural networks,

but in this paper we are going to use Multi-Layer Perceptrons (MLP)[13].

A decision tree[14] is a kind of machine learning algorithm known as successive partitioning algorithm. These algorithms work with the partitioning of the original data set in successively more homogeneous subgroups, which are partitioned in nodes until the desired detail level is accomplished. A partition decision must be taken in each node, in order to classify this data according to the label of a leaf-node in a subgroup.

These techniques are implemented in the Waikato Environment for Knowledge Analysis (WEKA) software package[15], a open source software under the GNU GPL. WEKA is a collection of machine learning algorithms for data mining tasks, which contains tools for data pre-processing, classification, regression, clustering, discovery of association rules and visualization.

3.2 Choice of attributes

A preliminary analysis was done to choose which attributes are important to classify the traffic as normal or suspicious. As we are working on network traces and the objective is not to search for attack signatures on data, i.e. on the packets' payloads, the attributes must be derived from the TCP/IP headers of the packets. Each packet has a TCP/IP header that contains some traffic information, such as source and destination IP address, source and destination port, transport protocol, amount of data sent in the packet, sequence and ACK number, information flags and the payload itself. In this work we intend to separate the logs according to the label assigned to the sessions. A session is a communication involving two unique pairs – source IP and port, destination IP and port – which have begin and end timestamps and can be considered complete.

The session analysis provides other information such as number of packets and number of bytes exchanged over this specific session. We work under the hypothesis that there is a limited set of attributes associated with the TCP/IP session that can be used to differentiate an attack from a normal session. We chose only seven attributes to be evaluated, in opposition to other approaches that uses more than 30 different attributes, such as proposed in KDD Cup 1999[16].

The attributes were chosen in a way that there is no linear combination among them, i.e, an attribute “average of bytes” is the linear combination of two other possible attributes: number of packets and quantity of bytes. Thus, we choose to have the minimum number of attributes that can be taken directly from the sets of TCP/IP headers from a network session. The attributes that are used in the datasets of this work are:

- Session duration: the time between the initial three-way handshake and the end of a session;
- Server port: the service offered by a server from our network;
- Server packets: the number of packets sent by the server of the session;
- Server bytes: number of bytes sent by the server of the session;
- Client packets: the number of packets sent by the client of the session;
- Client bytes: number of bytes sent by the client of the session;
- Class: a nominal value that can be either normal or suspicious and is related to the source of data, i.e., data from a honeypot are labeled as suspicious and data from the DMZ are labeled as normal.

3.3 Data pre-processing

In order to evaluate the data mining algorithms, two kinds of data sets were separated: a dataset containing network traffic related to the DMZ and labeled as normal, and a dataset with network traffic directed to this DMZ associated honeypot. Data obtained from a honeypot is inherently

associated with an attack since there are no real services in it. As an initial evaluation, it's worth to emphasize that the normal-labeled data is unfiltered, i.e., it is quite possible that there is some attack data concealed in the data collected from the DMZ.

To test the algorithms, we build several datasets consisting of real data from the DMZ and from the honeypot, separated by day. Some tools were developed to aid the task of getting the log data in tcpdump format and transforming it in arrays of attributes that can be read by WEKA algorithms. These tools extract basic TCP/IP header information from each session and label them according to its origin (DMZ or honeypot), converting the dumps to the ARFF (Attribute-Relation File Format) format files used in WEKA.

With the pre-processing task completed, the chosen algorithms were tested, using as input the generated data sets. The next section contains the tests done with each algorithm. We also present and comment the results.

4. RESULTS

The data was collected in some days of the months of February, April and June, 2005. The logs were collected daily and pre-processed as described in section 3. Corrupted logs or incomplete sessions were discarded, so only reliable data was used, and reliable data could be obtained only for some days of the chosen period. Those days and the corresponding amount of normal and suspicious log entries (sessions) are listed on Table 1.

Table. 1. Number of normal and suspicious sessions by the day of month.

Month-Day	Number of normal sessions	Number of suspicious sessions
February-01 (02/01)	376136	26026
February-02 (02/02)	293058	24008
February-03 (02/03)	299394	24012
February-04 (02/04)	290343	16016
February-05 (02/05)	8193	264
February-06 (02/06)	113535	12024
February-07 (02/07)	119898	10836
February-08 (02/08)	120366	10010
February-09 (02/09)	52074	2412
April-28 (04/28)	189806	19038
April-29 (04/29)	220838	20020
April-30 (04/30)	119774	12462
June-01 (06/01)	571550	26026
June-04 (06/04)	122064	6513
June-06 (06/06)	204112	13013
June-07 (06/07)	175848	15015
June-28 (06/28)	10914	714

One important fact about the attack data must be mentioned. Since attacks correspond to just a tiny fraction of the network activity, they amount to just some entries on the collected data. The algorithms being evaluated tends to yield simple classification schema for the data, therefore if the attack data were to be used as-is, it would be completely ignored (considered as normal) in order to obtain a simpler, faster classifier. To avoid this, the attack data was replicated, without addition of derived or random perturbation data, to amount at least 5% of the total data.

The next subsections will show the results of the application of the three chosen algorithms. The tables used to describe the results obtained consist of the following fields:

- Model building time, i.e. the time in seconds spent by WEKA to build a model to evaluate the data;
- False positives, or the percentage of normal data misclassified as suspicious;
- False negatives, or the percentage of suspicious data misclassified as normal;
- Correctness, that means the percentage of data correctly classified with the current model related to all instances evaluated;
- Effectiveness, that measures the usability of the model for other data sets. It is similar to the "Correctness", but we apply the model generated to data from the previous day available (or next day available, in the case of a data set belonging to a different month).

4.1 KNN Results

The first batch of tests was done using the most simple method possible with the KNN algorithm: the 1-nearest neighbor. This classifier works calculating distances and comparing them in the attributes space, labeling a instance with the class of its nearest neighbor. In our case, the separation of data will be done with the creation of a set of hyperplans to divide the instances in normal or suspicious. Depending on the quantity of data and the number of neighbors that will be used as comparison parameter, the KNN could be very slow to process, but it is considered to have a good classification, independent of the data distribution model.

The first try was taking too much time, so we decided to verify the classification time of KNN with a step by step increasing procedure. Thus, we tried to classificate the smallest data set (02/05) with the 1-nearest neighbor. This test last 70 seconds and resulted in 100% of correctness. So, we applied the 3-nearest neighbor in the same data set, resulting in the same 100% of correctness in 91 seconds. After it, we passed to the (06/08) data set, that was finished in 197 seconds with 97.76% of correctness. An increase of approximately 30% in the data set almost triple the classification time. The next data set (02/09) had a better correctness rate (99.25%), but its classification time was of 5163 seconds. The last test was done on the (04/28) data set, which have 189706 normal instances and 19038 suspicious instances. After more than 3 hours of test (exactly 11676 seconds), the classifier had just evaluated 36900 instances (17.67% of total).

With more neighbors used in comparison, the time used in classification should greatly increase, due to the amount of distance calculations required. These results showed the impossibility to use the KNN in our data sets without modifications, because most of our data sets have more than a hundred thousands of points and the timing is a main factor in our case. A variation to consider is to store less labeled samples to optimize the comparison, in a scheme of summarization or grouping of similar instances. In this way, the complexity of processing decreases, reducing consequently the execution time.

4.2 Artificial Neural Network Results

The architecture of MLP used was 6:2:2, i.e., six input neurons, two neurons in the hidden layer and two output neurons that indicates the classes.

As we can't see in a traditional graphic the separability of data, because our space of attributes is 6-dimensional, we have to guess the number of neurons on the hidden layer. The hidden layer neurons are used to create hyperplans and combinations, aiming to classify non-linearly separable distributions. Initial tests with the data showed that more than one hidden layer slow the process and don't present better results.

The time spent in each test varied between 37 and 2216 seconds, which is acceptable. The false positives rate is lower than 1% in all data sets, but the false negatives rate is too high in most of cases. In the (02/02) data set, there was no classification at all. Thus, the suspicious sessions are being considered normal and, despites the false positives rate being low, providing a reduced data set to the analysts, with this models they will not see a lot of attacks or suspicious behavior in their networks. The results are presented in Table 2.

Considering the objectives, which is to reduce data to human analysis, the neural networks provided good results, but this reduction caused a side effect that is to let suspicious data pass undetected. Furthermore, the model is easily reusable (the architecture and weights), but the interpretation of results is difficult, because of the complexity of the classification mechanism. Some tests using 10 neurons in the hidden layer was done, to see whether it separates better with more hyperplans. This new test just increased the evaluation time and don't have any effects on the results. For example, using the (02/06) data set with this architecture resulted in a model building and evaluation time of 7354 seconds while the correctness rate increased 0.23%.

Table 2. Results obtained using a 6:2:2 MLP neural network

Date	Model building time (seconds)	False positives	False negatives	Correctness	Effectivity
(02/01)	1281.65	0.11%	30.76%	97.89%	99.76%
(02/02)	1098.86	0	100%	92.42%	96.37%
(02/03)	950.39	0	0.04%	99.99%	91.71%
(02/04)	974.38	0.49%	62.25%	96.27%	96.92%
(02/05)	49.28	0	7,57%	99.76%	99.80%
(02/06)	450.95	0.13%	36%	96.42%	86.20%
(02/07)	467.82	0.02%	27.77%	97.67%	99.99%
(02/08)	481.89	0.004%	80%	93.85%	99.70%
(02/09)	165.68	0.01%	33.33%	98.51%	99.26%
(04/28)	1047.1	0.16%	52.63%	95.05%	99.85%
(04/29)	741.69	0.47%	50%	95.40%	94.71%
(04/30)	467.07	0.35%	22.52%	97.55%	94.73%
(06/01)	2216.69	0.87%	12.28%	98.62%	99.88%
(06/04)	470.2	0.06%	50.97%	95.02%	98.36%
(06/06)	598.81	0.09%	15.38%	98.99%	96.18%
(06/07)	580.87	0.01%	13.33%	98.94%	95.84%
(06/28)	37.91	0.43%	42.85%	96.95%	98.17%

4.3 Decision Tree Results

Last, we have carried out tests using decision trees, a supervised classification algorithm. Since it is important to understand the classification process, it is not convenient to have a tree with a great depth, in order to facilitate the results interpretation. So we use a pruned tree with confidence factor of 0.25.

As all the attributes are tested in order to create nodes and this creation is based on the information gain provided by the attribute in relation to the class, the decision trees don't have the processing load involved in training/learning or calculating distances among several instances. So, the results appears quickly. The discretization of numeric attributes contributes to the comprehension of the final results, that are the best obtained in the tests. The false positives rate obtained for each data set is very low, while the false negatives rate are much better than that obtained with neural networks. The results of these tests are in the Table 3.

Table 3. Results obtained using a C4.5 decision tree.

Date	Model building time (seconds)	False positives	False negatives	Correctness	Effectivity
(02/01)	48.05	0.05%	23.07%	98.45%	99.91%
(02/02)	19.38	1.78%	25%	96.45%	97.96%
(02/03)	51.54	1.62%	0	98.49%	98.21%
(02/04)	29.35	1.47%	0	98.59%	98.33%
(02/05)	0.62	0	0	100%	98.85%
(02/06)	11.01	0.97%	0	99.11%	99.54%
(02/07)	11.82	1.08%	0	99%	98.86%
(02/08)	6.06	0.02%	0	99.97%	100%
(02/09)	3.97	0.01%	16.66%	99.24%	99.97%
(04/28)	24.06	2.19%	0	98%	98.21%
(04/29)	41.45	1.37%	10%	97.57%	97.84%
(04/30)	19.98	0.14%	9.67%	98.95%	99.86%
(06/01)	78.11	0.1%	3.84%	99.73%	96.23%
(06/04)	10.92	0.84%	0	99.19%	97.34%
(06/06)	20.71	0.07%	7.69%	99.46%	99.90%
(06/07)	8.27	0.11%	0	99.89%	99.95%
(06/28)	0.98	0.35%	33.19%	97.62%	98.04%

5. CONCLUSION

In this paper we attempted to solve a network-security related task with the aid of data mining techniques. The main purpose was to simplify some tasks which must be performed by a human analyst. In order to avoid analysis of network connection payloads, we focused on a relevant subset of features which can be extracted from the packets header, which is very cost effective and it is expected to be useful to classify normal versus suspicious network activity.

From the chosen methods and algorithms, we conclude that data decision trees yielded the best results, both numerically and considering its decision process representation -- decision trees are very similar, structuraly, to expert systems, therefore can be easily understood by human analysts and modified to include more knowledge. We consider this feature to be very important since it allows the human analyst to identify not only what can be an attack but also why, i.e. which features and values were used in the classification.

A traditional machine learning algorithm, K-nearest neighbors, also yielded interesting results, but the implementation used is computationally very expensive, therefore should be improved before real application since one of the requirements for the solution of our problem is that the processing time must be acceptable to allow a timely answer time from the analyst.

Future work on this research will be directed to the improvement of the KNN algorithm, possibly using a hybrid supervised/unsupervised learning schema using clustering as pre-processing step, and application of visualization and identification of trends using time-stamped decision trees.

ACKNOWLEDGEMENTS

The authors would like to thank Benicio Pereira de Carvalho Filho for authorizing the use of network dumps from Brazilian Institute for Space Research (INPE) DMZ.

REFERENCES

1. GNU Project, *grep*, http://gnu.org/software/grep
2. M. J. Ranum, *Artificial Ignorance: How-To Guide*, http://www.ranum.com/security/computer_security/papers/ai/index.html, 1997.
3. R. Rehman, *Intrusion Detection with SNORT: Advanced IDS Techniques using SNORT, Apache, MySQL, PHP and ACID*, Prentice Hall PTR, 2003.
4. H. S. Javitz and A. Valdes, "The SRI IDES statistical anomaly detector", *Proc. of the Symposium on Research in Security and Privacy*, 316 – 326, Oakland, CA, May 1991.
5. U. M. Fayyad and G. Piatetsky-Shapiro and P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, MIT Press, 1996.
6. L. Ertoz and E. Eilertson and A. Lazarevic and P. Tan and J. Srivastava and V. Kumar and P. Dokas, "The minds – minnesota intrusion detection system", Next Generation Data Mining, 2004.
7. D. Barbará and J. Couto and S. Jajodia and N. Wu, "Adam: a testbed for exploring the use of data mining in intrusion detection", *ACM SIGMOD Record*, **v. 30**, n. 4, 15 – 24, December 2001.
8. L. Mé, "Gassata, a genetic algorithm as an alternative tool for security audit trail analysis", First International Workshop on the Recent Advances in Intrusion Detection (RAID), September 1998.
9. L. Spitzner, *Honeypots – Tracking Hackers*, Addison-Wesley, 2003.
10. The Honeynet Project, *Know Your Enemy – Learning About Security Threats*, Addison-Wesley, 2004.
11. P. Adriaans, and D. Zantinge, *Data Mining*, Addison-Wesley, 1996.
12. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1998.
13. L. V. Fausett, *Fundamentals of Neural Networks*, Prentice Hall, 1994.
14. J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.
15. E. Frank and M. Hall and L. Trigg, http://www.cs.waikato.ac.nz/~ml/weka/index.html
16. KDD Cup 1999, http://kdd.ics.uci.edu/~kdd/databases/kddcup99/kddcup99.html