Applying Autonomic Intrusion Detection on Web Applications

EDUARDO ALVES FERREIRA[1]
RODRIGO FERNANDES DE MELLO[1]

USP - Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Avenida Trabalhador São-carlense, 400 13.566-590 - São Carlos - SP - Brazil
[1](eaf,mello)@icmc.usp.br

Abstract. The characterization of system behavior is a commonly considered approach when performing intrusion detection. Such approach is limited when the observed context is unstructured, that is, context characterization is not a trivial task. In order to tackle this issue, this paper considers the use of single-pass clustering techniques to quantize unstructured data, generating time series where novelty detection techniques can be employed to detect intrusion incidents. We evaluate this approach using public system characterization datasets, and the outputs of a web application in a simulated environment. We observed that our approach is capable of aggregating context information into time series in order to represent the behavior of applications with fairly enough precision to detect attacks.

Keywords: Intrusion detection, Web applications.

(Received November 2nd, 2011 / Accepted March 6th, 2012)

1 Introduction

The characterization of process behavior is commonly considered to compose autonomic intrusion detection solutions. In order to implement this approach, one must first extract and characterize the normal behavior of an application (e.g. when it is working in an isolated network, in a simulated environment or during acceptance or stress testing sessions). Later on, the system can be monitored in production environments, where observed novelties are handled as possible attack incidents.

To characterize the behavior of an application, two basic approaches are described in related works: The first one uses model-based techniques to evaluate application outputs, ignoring any temporal dependency among events. The second approach considers only the temporal dependency, among a limited number of observations. The major limitations of those approaches are the fact that each one ignores one aspect of the application, and also the fact that proposed solutions are closely tied to a specific application domain, hindering the proposal of a fully autonomous solution. A third approach, based on clustering and time series analysis, is not only capable of addressing both aspects of an application (i.e. the multi-dimensionality of system observations and the temporal dependency among objects), but it can also be independent of the monitored data model. Most works that follow this third approach present two major shortcomings, though: the first one is that they are based in batch approaches, using algorithms with high computational complexity, which is not acceptable in a real scenario. The second limitation is that most of those works still use context-specific functions to process the system behavior data, what restricts their use on a truly autonomous solution.

In this paper, we evaluate a variation of this third approach, that deals with these two limitations, i.e. processing the dataset in a single sweep (or single pass), and using functions that are not tied to a specific application domain. We perform three sets of experiments. In the first set, we evaluate the precision of generic functions relative to domain-specific functions, when all algorithms are applied in a single-sweep fashion.

These experiments are executed on sequences of system calls, and indicate that the generic functions exhibit equivalent precision to the context-aware ones, and that single-sweep clustering is capable of characterizing the behavior of the application with enough precision to distinguish normal from attack sessions. The second set of experiments evaluates the scalability of the technique, when applying our proposal to a dataset with more than 5 million operating system audit records, and we confirm that the technique can deal with production-scale data.

The third set of experiments evaluates the main proposal of this work, when we apply the autonomic technique to the outputs of a three-tiered Web application. We designed a simulation environment using open-source Web applications and a relational database, where test scripts were executed to collect data flows that represent the complex behavior of a dynamic Web application. The experiments confirm that attack sessions can be differentiated from normal ones, corroborating the main hypothesis of this work.

This paper is divided in the following sections: Section 2 presents some related work in intrusion detection via system characterization; Section 3 presents the proposed approach, describing the intrusion detection technique, the datasets and the simulation environment. Experimental and simulation results are presented and discussed in Section 4, and Section 5 presents the concluding remarks.

2 Related Work

Two basic approaches are usually employed to characterize the behavior of an application. The first one uses model-based techniques to evaluate system outputs, not considering any temporal dependency among them. For instance, Eskin et al.[7] employ clustering techniques to characterize unlabeled system observations and use outlier detection to point out anomalies. Kruegel et al.[11], on the other hand, use a multi-model detector-based technique to model a range of characteristics (e.g. string length and string character distribution) present in system call parameters. Any observation that does not fit in this model is considered an intrusion.

The second approach models the behavior of applications considering the temporal dependency among observations. Every application procedure generates a sequence of objects (also called events) from a limited set when performing operations, and, in this case, the sequence of those objects defines the application behavior. Forrest et al.[8] use a sliding window to observe the sequence of system calls performed by an application, while Albertini and de Mello[1] and Pereira and de Mello[15] estimate the Entropy of Markov Chains, generated from the sequence of observations, to indicate novelties.

A third approach, proposed by Zanero and Savaresi[20], is capable of addressing both aspects of an application: the multi-dimensionality of system observations and the temporal dependency among objects. In that work, network packets are firstly submitted to a clustering algorithm (K-Means [12], SOM [10] and a hierarchical agglomerative algorithm are evaluated), and the output of this stage is analyzed through a sliding window. That approach is also employed by Maggi et al.[13] to evaluate the system call parameters of several UNIX applications, in which a hierarchical clustering technique is applied in the first stage, and Markov Chains are used to evaluate the probability of the observed behavior belonging to a normal set.

While being useful to characterize the behavior of network softwares and presenting enough precision to distinguish normal from anomalous behavior, all those techniques present two major drawbacks: first, the model-based techniques demand the definition of an application-specific modeling, which might be an obstacle to employ such technique to new domains or provide autonomic solutions. Another limitation lies on the issue that, usually, system characterization data is generated on-the-fly (as applications are executed on production environments), producing huge amounts of data, which cannot usually be stored on a secondary memory device. It is therefore interesting to consider machine learning techniques which are capable of processing datasets in a single-sweep stage, having the minimal dependency on the observed context. With this setting, we need algorithms that not only have near-linear time complexity, but also operate on datasets in a single sweep, i.e. perform on every observation at the time they are produced (or very close to that), keeping only a small amount of data in memory.

3 The Proposed Approach

In this paper, we employ single-sweep clustering and novelty detection algorithms to quantize application behavior and, therefore, perform autonomic intrusion detection in unstructured contexts. The quantization stage applies a clustering algorith to the dataset, allowing the mapping from dataset inputs (i.e. complex, multi-dimensional objects) to simple numeric and sequential identifiers. This allows us to transform a sequence of complex objects into a sequence of numeric identifiers, that represent the temporal behavior of the application. This sequence is then submitted to novelty detection algorithms to point out intrusion incidents. Our approach

is based on the architecture presented in Figure 1. Such architecture is composed of three stages: the first one (a) extracts the events from a system under the production environment (e.g. system call data, CPU and memory usage, log or network messages), providing the data or domain model; such data model is input to the next stage (b), which considers a distance function to characterize the similarity among objects (also named observations or events), an optional merge function (which is responsible for representing a multidimensional object into a single object), and also a clustering algorithm that receives objects as input and finally produces time series representing the system behavior; such time series is input to the last stage (c), that applies a novelty detection algorithm on it in order to distinguish attack events (novelties) from the normal behavior.

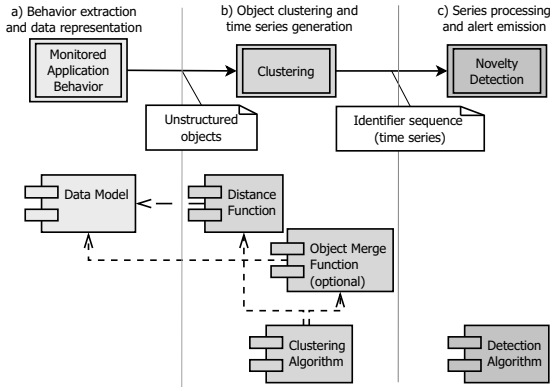

Figure 1: Clustering and Novelty Detection Architecture.

We evaluate the use of single-pass clustering techniques to quantize unstructured data, allowing the characterization of complex application behavior with little knowledge about its structure. By using this approach, we group similar application behavior observations, overlooking small variations that are expected on the normal operation, while attempting to distinguish objects that are observed only during attacks. By grouping similar observations of unstructured system outputs, and assigning identifiers to every cluster, we then generate time series where novelty detection techniques can be applied to. We also evaluate the detection precision when generic distance functions are used, since they are needed to compose a truly autonomous solution, independent of the application context.

Based on previously conducted experiments, we chose, for clustering, the Greedy Leader-Follower Adaptive algorithm [4], because it presented acceptable performance and precision, providing an adaptive solution that does not strongly depend on parameters definition, and also because it does not demand the definition of a point merging function. This algorithm maintains a set of centroids in memory, to represent the most usual dataset observations. A parameter θ is assigned to every centroid, to represent its coverage area. The algorithm sweeps the dataset and, for every object, computes the distance from the object to every centroid. The centroid with the smallest distance to the observed object is considered the winner. If the distance in between the object and the winning centroid is smaller than θ, then the object is associated with that centroid, otherwise a new centroid is created at the position of the observed object. Every time a new centroid is created, the number of existing centroids is verified: if the number of centroids exceeds a parameter k, an operation is executed to reduce such number. In order to perform this reduction, the algorithm looks for the two closest centroids, and merges them into a single one. Parameter θ of the composed centroid is then incremented to accommodate objects of both, and the initial θ is incremented to better adapt to the observed dataset.

Based on Albertini and de Mello[1] and also on the previously conducted experiments, we considered the Shannon's Entropy [17] of time series as novelty detection algorithm. Each observation in the sequence is represented as a state on a Markov Chain, and the probability of transitions are estimated from sequences of time series. The Entropy value is computed with Equation 1, where C is the set of all groups generated from the clustering stage, and $p(i, j)$ is the transition probability from group i to j. This transition probability is estimated from the time series observations (i.e. the number of transitions from state i to state j divided by the lenght of the time series). The amount of novelty in the series at a specific time is given by the derivative of the Entropy with respect to time. The most important aspect of this algorithm to this work is its linear computational complexity (given that the number of distinct identifiers is constant and must be kep to a reduced value, not only for performance issues but also to avoid overfitting), which is an important aspect to make our proposal scalable.

$$H = -\sum_{i \epsilon C} \sum_{j \epsilon C} p(i,j) log_2(p(i,j)) \qquad (1)$$

4 Experiments

Being defined the algorithms to perform clustering and novelty detection, we had to choose the datasets to conduct experiments on. Several system characterization datasets are presented or described in the literature. By far, the most widely used is the "1998 and 1999 DARPA intrusion detection", which contains network traffic and

Solaris BSM output captured during several weeks on a simulated environment [6]. However, this dataset presents several flaws [14] and it does not contain all information needed to characterize the behavior of specific applications. Therefore, we only considered the Solaris BSM output as dataset to assess the scalability of our solution. On the other hand, we evaluated the precision of our approach through other experiments.

Most application characterization datasets are composed of sequences of system calls. For instance, Warrender et al.[19] and Kruegel et al.[11] present and describe datasets with hundreds of thousands of system calls extracted from production systems. However, either the system call parameters are not present or the datasets are not published because they contain confidential information, which is a common issue faced when data is extracted from production systems. By contrast, Twycross and Aickelin[18] provide a dataset of system characterization which contains system calls and their parameters for server applications, in which the behavior was simulated using sessions from public datasets. To build this dataset, a public domain dataset of FTP protocol sessions was used to simulate 55 sessions of normal usage, while attack sessions were manually executed. Although this dataset is shorter than others (containing around $85,000$ system calls, when they usually have around a million objects), it is more appropriate for this work since it is public and contains the value of system call parameters. This dataset is used in the first set of experiments, in which we evaluate the precision of the proposal and its ability to work in a single-sweep fashion, with generic distance functions.

Finally, to collect data that represents the behavior of a web application, we assembled an environment composed of an HTTP server in which web applications are deployed, and a relational database that is used by those applications. To simulate users' behavior, we defined scripts for a test automation tool that simulate normal and also attack requests. For the attack simulation, we have chosen the `CVE - 2010 - 2908` vulnerability, a SQL-injection flaw that affects an integration module of a popular Web-based Content Management System. In order to exploit this vulnerability, we assembled a malicious request tampering one of the parameters of the web application, which is not validated by the application and is concatenated to a SQL query, allowing the execution of arbitrary SQL statements by the attacker.

This vulnerability was selected due to three main aspects. First, because it is a SQL injection vulnerability conducted by HTTP requests, which is the main focus of this work. The second aspect is because the vulnerable applications are very popular LAMP ("Linux, Apache, MySQL and PHP", an open source platform for Web applications) applications, freely distributed on the Internet, which simplifies the assembly of a realistic simulation environment and reinforces the importance of the attack. Finally, the third aspect is the fact that traces from these attacks can be observed not only in the user requests and the database commands, but also in the Web Service integration tier in between applications, which allows monitoring an additional attack vector. Considering this setup, we can monitor the HTTP requests issued by users, the Web Service messages exchanged in between applications, and SQL commands executed in the database. We executed 90 sessions of normal usage, 5 sessions of SQL-Injection attacks, and 5 sessions of anomalous behavior. The anomalous behavior represents user sessions executed after confidential information was extracted via the SQL-Injection attacks.

All three datasets used in the experiments are modeled as sequences of objects that represent either normal or attack behavior (the Web dataset is in fact composed of three datasets: one of SQL statements, one of HTTP requests and a last one of Web Service requests).

We executed three sets of experiments to evaluate different aspects of our proposal. In the first set, we evaluated the precision of the technique when single-sweep clustering is used, and we compared the results of the detection when generic and specific distance functions are used. In this first set, we used a system call dataset, and compared the results of three distance functions: SSD (simple syscall distance), NCD (normalized compression distance [5]) and CSD (complex syscall distance). CSD is a context-specific distance, only applicable to system calls (i.e. to a specific domain). It is based on a hierarchical approach, where only system calls under the same type are clustered together. After verifying the type, the distance in between the objects is calculated based on the value of the parameters and the return value. The first step to calculate the distance requires the computation of the distance in between every attribute value: for numeric (i.e. integer, date and network address parameters) values, the distance is given by $\frac{|a-b|}{max(a,b)}$. Nominal attributes have distance equals to 1 when values are different and 0 when they are equal. For character strings, we used the normalized padded Hamming distance [9]. We end up having a vector of distances in range $[0, 1]$, which is combined into a single value using the Euclidean distance.

SSD distance, on the other hand, represents a system call as a tuple, composed of its type, concatenated to pa-

rameters and return value. The distance in between objects is then given by the Euclidean distance in between tuples. This distance can be considered as generic as it is applicable to any object that could be represented as a tuple. Finally, since NCD can be applied to any array of bytes, we serialized system call objects to calculate this distance. The experimental results show that single-sweep clustering is capable of defining series that represent the application behavior with enough precision to differentiate attacks from normal scenarios, and also that the generic distance functions show equivalent precision to the context-specific functions, what motivates its adoption on later experiments to compose a fully autonomous intrusion detection application.

The second set of experiments evaluates the scalability of the technique, using a dataset with more that 5 million Solaris BSM records [6]. Every record is composed of a sequence of tokens, and we used the Euclidean distance in between tokens as the distance between records. We observed that the proposed approach can scale to a production-sized dataset, and attack sessions can be successfully differentiated from the normal ones.

The third set of experiments evaluates the proposal on the data collected from the Web application simulation. We generated three datasets, named HTTP, XML-RPC and SQL, containing user HTTP requests, Web Service messages and SQL commands executed in the database. NCD distance is used for all experiments in this set.

On all experiments that depend on NCD distance, the `pickle` serialization algorithm available in `Python` programming language [16] was used, and compressions were performed using `gzip` with 1 as compression level. All evaluated objects were in the limit of $32KBytes$, which is required for the NCD distance stability when using the `gzip` algorithm [3].

The effectiveness of the proposal is measured using receiver operating characteristic (ROC) curves generated from the novelty levels observed on the novelty detection stage. A ROC curve is a curve where the x axis represents the sensitivity, i.e. the true positive rate (TPR), and the y axis represents the value of 1 minus the specificity, i.e. the false positive rate (FPR), for a classifier when its discrimination threshold is varied. To obtain the curve, we assumed that the normal and the attack novelty indexes generate two normal distributions, and estimate their means and standard deviations based on the observed data.

The area under the ROC curve (AuC) is used as an index of the intrusion detection capability. The area under the ROC curve was used as indicator of classificator accuracy due to the fact that this index is not biased [2] when the number of observations is unbalanced (what happens for anomaly detection datasets). On next sections, results from the characterization stage are presented.

4.1 System call experiments

Figures 2 and 3 presents the comparison of some of the ROC curves obtained when generic distances (SSD and NCD) are used, in comparison to one of the results from the syscall-specific distance (CSD). We present a curve from the CSD experiment where $k = 10$ (where k is the number of centroids in the clustering stage), because it represents an experiment in which the area under the curve is near the average observed on all CSD experiments. On this subset of the results, we observe that SSD outperformed the context-specific distance (which is noticeable by the curves closer to the vertex $(0, 1)$ and higher value of area under this curve), while the NCD experiments present slightly inferior results.

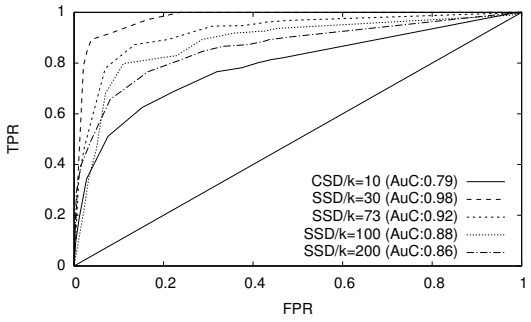

Figure 2: Simple Distance.

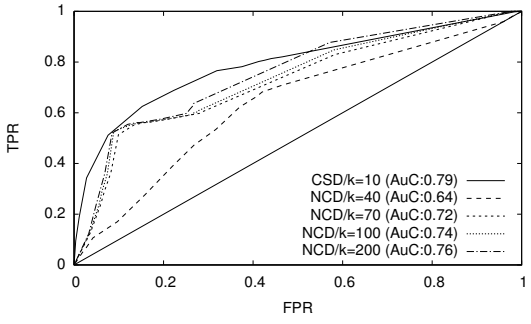

Figure 3: NCD Distance.

Figure 4 presents the value of the area under the ROC curves in relation to the parameter k of the clustering experiment (i.e. the number of groups), for all system call experiments. To compare the average per-

formance for all experiments, we executed two hypothesis tests considering the observed mean and standard deviation for the context-specific distance and the two generic distances.

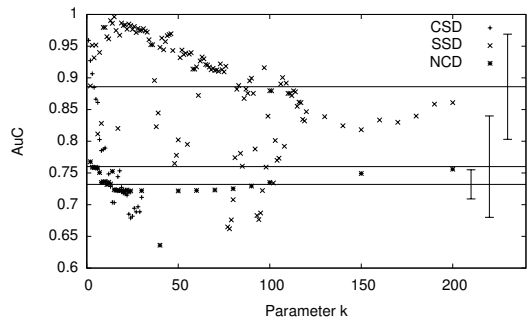

Figure 4: Area under ROC curve for each k parameter. The horizontal line presents the mean value, while the vertical bars present the standard deviation.

The first test compares the results of CSD and NCD distances. We employ an unpaired, two-tailed two-sample Welch's t-test to evaluate whether the observed difference in between means is statistically significant. The NCD experiments present an average area under the ROC curve of 0.73 with standard deviation of 0.02, while CSD experiments present slightly superior results, with 0.76 as average and 0.08 as standard deviation. We formulate two hypothesis:

- Null hypothesis($H0$): $\mu_{csd} = \mu_{ncd}$
- Alternate hypothesis($H1$): $\mu_{csd} \neq \mu_{ncd}$

The observed mean difference represents a *t score* of -1.85, and with 33 degrees of freedom we observe a probability of 0.07 of this difference being observed by chance, i.e. it is not sufficient to reject the null hypothesis with $\alpha < 0.05$. Considering this, we cannot say that the context-specific distance has presented superior results when compared to NCD, so we assume that both techniques presented equivalent results.

The second statistical test compares the result of SSD experiments to CSD ones. Again, we employ an unpaired, two-tailed, two-sample Welch's t-test to evaluate whether the observed difference in between results is statistically significant. SSD experiments presented an average area under the curve of 0.886 with 0.08 standard deviation. Thus, we formulate the hypothesis:

- Null hypothesis($H0$): $\mu_{csd} = \mu_{ssd}$
- Alternate hypothesis($H1$): $\mu_{csd} \neq \mu_{ssd}$

The observed mean difference represents a *t score* of -7.7. With 44 degrees of freedom, we have a probability on the magnitude of 10^{-9} of observing this mean difference by chance, so it is safe to reject the null hypothesis, and hence we can say that the generic distance has presented superior results to the context-specific one.

These experimental results supported the hypothesis that techniques that do not consider context information are also capable of characterizing the behavior of an application, and that single-sweep clustering techniques can successfully be applied on the first step of the characterization task. It is therefore feasible to differentiate attacks from normal sessions through a single sweep on the dataset, using an autonomic approach that is independent of the underlying data model that describes the application behavior.

4.2 BSM logs experiments

After evaluating the precision of our proposal through experiments using the system call dataset, we were motivated to assess the scalability of it, i.e. check if it is capable of discriminating normal from attack sessions in a dataset with a greater order of magnitude. In order to perform this evaluation, we executed experiments on a subset of the DARPA IDEval BSM dataset, and results are presented in Figure 5.

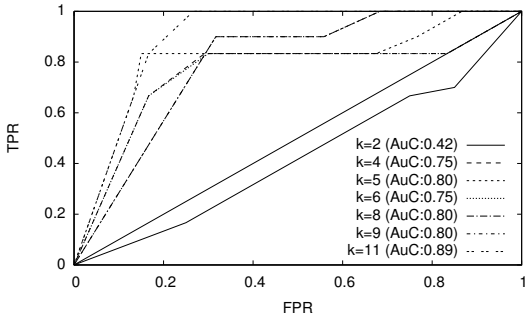

Figure 5: ROC curves for DARPA IDEval BSM dataset.

When parameter k ranges from 2 to 14, we observe areas under the ROC curve in between 0.43 and 0.89, with an average of 0.64. To evaluate the effectiveness of the proposal, we need to check if the value of this average is significantly above 0.5, i.e. if the technique has greater differentiation capability than a random classifier for most of the experiments. To evaluate that, we performed an one-tailed, one-sample Student's t test, because we had a reduced number of observations and wanted to verify whether the average value is above a threshold. Thus, we formulated two hypothesis:

- Null hypothesis($H0$): $\mu \leq 0,5$
- Alternate hypothesis($H1$): $\mu > 0,5$

Having an average of 0.64 with a standard deviation equals to 0.165, the probability of observing an average below 0.5 is 0.005, when we consider 12 degrees of freedom. Thus, it is safe to reject the null hypothesis even with $\alpha < 0.01$, so we confirmed that the technique can be successfully applied to a production-scale dataset.

4.3 Web experiments

Finally, we present the intrusion detection results when testing our proposal on the dataset extracted from a Web application. Figure 6 presents some of the ROC curves observed when the technique is applied to the dataset of HTTP user requests. The most effective experiments occur when parameter k ranges from 4 to 19: in this dataset, we observed that when $k \leq 3$, the clustering stage is not capable of generating time series that characterize the dataset with enough precision to differentiate normal from attack sessions. On the other hand, when $k \geq 20$, we start observing overfitting situations, i.e. when the added information interferes in detection. When $4 \leq k \leq 19$, we observe an average area under the ROC curve of 0.68.

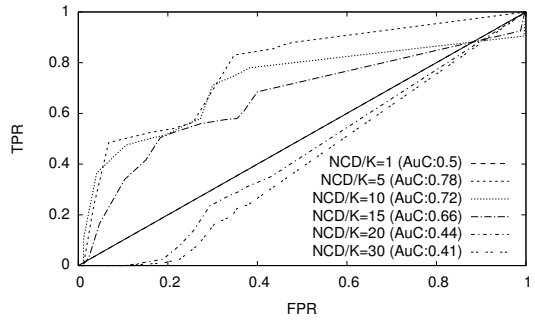

Figure 6: ROC curves for HTTP dataset.

To evaluate if the technique can effectively differentiate normal from attack sessions, we again perform an one-tailed, one-sample Student's t test to evaluate whether the average area under the curve is above 0.5. Thus, we formulated the following hypothesis:

- Null hypothesis($H0$): $\mu \leq 0,5$
- Alternate hypothesis($H1$): $\mu > 0,5$

When all test cases are considered, i.e. for all $1 \leq k \leq 30$, we observed an average area under the curve of 0.56 with standard deviation equals to 0.14. Considering 29 degrees of freedom, we have a probability of 0.008 of observing an average area under the curve below 0.5, and hence it is safe to reject the null hypothesis even for $\alpha < 0.01$. By analyzing the results, we confirm that the technique is effective to detect intrusions in this context, as long as we have good clustering results (i.e. avoiding excessive generalizations and overfitting situations).

The second set of experiments on Web datasets evaluates the effectiveness of our proposal to detect attacks in the Web Service messages. Figures 7 and 8 presents some of the ROC curves observed on experiments. We evaluated two situations: in the first, we attempted to detect both the SQL-Injection attacks and anomalous user navigation as attacks, as we did on the other Web experiments. On the second situation, we tried to detect only the SQL-Injection attack. When we compared the ROC curves in Figures 7 and 8, we notice that the technique is not capable of detecting anomalous situations as attacks. This happens because the anomalous situations do not depend on the Web Service functionality, and hence no anomaly is observed on this situation.

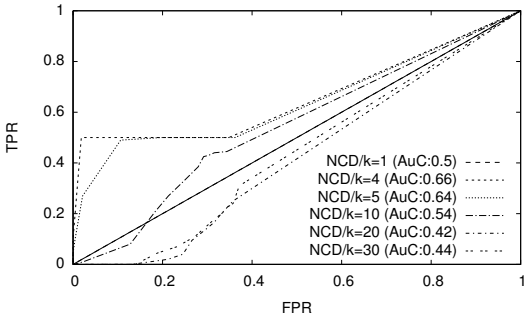

Figure 7: Anomalies and attacks.

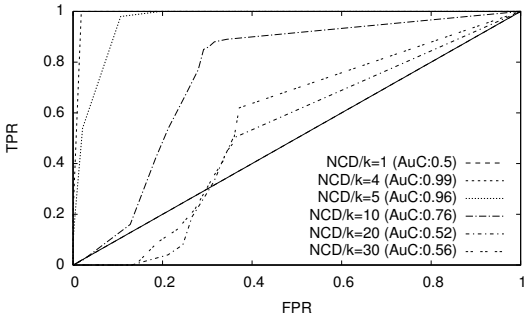

Figure 8: Attacks only.

When we evaluate the effectiveness of the technique considering both attacks and anomalies, we observe that a two-tailed one-sample Student's t-test is not conclu-

sive to state that the technique is superior to a random classifier. On the other hand, if we consider only the SQL injection test cases as attacks, having $1 \leq k \leq 30$ we observe an average area under the ROC curve of 0.64 with standard deviation equals to 0.16. Considering 29 degrees of freedom, a one-tailed one-sample Student's t test yields a probability of less that 10^{-4} of observing this area difference by chance. Thus, it is safe to reject the null hypothesis even with $\alpha < 0.01$, which indicates that the technique was effective to detect attacks on this vector, even though it was not able to detect anomalies (which is expected, since no attack trace is present on that vector in this situation). Also, as for the previous experiments, we observe in Figures 7 and 8 a reduction on the effectiveness when we have overfitting situations on the clustering stage.

In the last experiment with the Web application, we applied our proposal to the dataset with SQL commands executed on the database. Figure 9 presents some of the ROC curves obtained on this experiment.

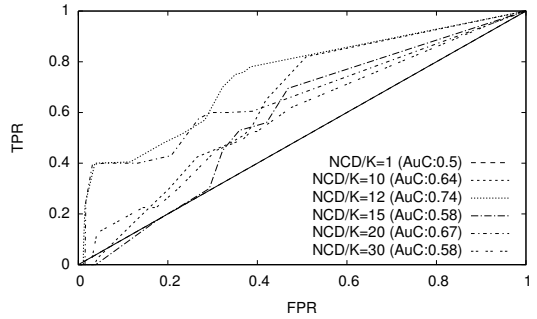

Figure 9: ROC curves for the SQL dataset.

With $1 \leq k \leq 30$, we observed an average area under the ROC curve of 0.54 with standard deviation equals to 0.13. Considering 29 degrees of freedom, an one-tailed, one-sample Student's t test yields a probability of 0.03 of this result being significant, so the null hypothesis can be rejected only for $\alpha < 0.05$. Thus, we noticed that we have a much smaller differentiation capacity for this dataset than in previous situations, what happens due to the greater amount of data being is presented (since more objects are present on every session, as for each request several SQL statements are executed on the database), and also to a leniency observed on the NCD distance when greater amount of shared data is present on the dataset observations (since each SQL statement is a relatively large object). A possible workaround to deal with this issue would be by reducing the length of every section, or by using a more specific distance function only on this point of the whole intrusion detection application, sacrificing part of the autonomy of the solution in favor of better precision.

In every experiment presented in this work, we noticed reasonably low levels of area under the curve in comparison to related works (while most works present indexes above 0.9, we obtained indexes ranging from 0.6 to 0.9 [8, 13, 18]). However, we must point out that we currently do not aim at obtaining the highest precision at all, but we intend to propose an approach that can be easily applied to any application context, so that we can characterize several aspects of a working system in an autonomic fashion. Throughout our experiments, we showed the effectiveness of our proposal (expressed by an average area under the curve above 0.5 regardless of any parameter values) instead of showing how precise a solution can be when considering ideal parameters and context-specific data functions. We considered that the definition of these dependencies for every specific application would hinder the adoption of any proposal on a real scenario, and we believe that, as showed in previous works [18], a completely autonomic solution should be based on the collection of data from several tiers of a working system, so we need techniques that require the least supervision as possible to build a whole intrusion detection application.

5 Conclusions

In this paper, we proposed a variation of a commonly employed technique for novelty detection that can be applied to any context, in a single sweep of a dataset. In three groups of experiments, we have evaluated the precision and scalability of our approach, and finally, we applied it on a web-application scenario. All those experiments have confirmed that generic distance functions can be used to characterize the behavior of applications, and also that a single-sweep approach can indeed be used to perform autonomic intrusion detection on different domains.

A shortcoming we observed in this work is the fact that data was collected from simulations of real environments, and not from production environments. As observed by McHugh[14], the issue on how to obtain this kind of information, without bias and not compromising confidential information, is still an open question. As future work, we intend to evaluate other datastream clustering algorithms, in order to increase the precision of the technique, and also to research more realistic simulation environments.

References

[1] Albertini, M. K. and de Mello, R. F. A self-organizing neural network for detecting novelties.

In *ACM Symposium on Applied Computing*, pages 1–5, 2007.

[2] Bradley, A. P. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[3] Cebrián, M., Alfonseca, M., , and Ortega, A. Common pitfalls using the normalized compression distance: What to watch out for in a compressor. *Communications in Information and Systems*, pages 367–384, 2005.

[4] Charikar, M., Chekuri, C., Feder, T., and Motwani, R. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 626–635, 1997.

[5] Cilibrasi, R. and Vitányi, P. M. B. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[6] Cunningham, R. K., Lippmann, R. P., Fried, D. J., Garfinkel, S. L., Graf, I., Kendall, K. R., Webster, S. E., Wyschogrod, D., and Zissman, M. A. Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation. In *Proceedings of Third Conference and Workshop on Intrusion Detection and Response*, pages 1–5, 1999.

[7] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security*, pages 1–20, 2002.

[8] Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 120–128, 1996.

[9] Hamming, R. Error detecting and error correcting codes. *Bell System Technical Journal*, pages 147–160, 1950.

[10] Kohonen, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[11] Kruegel, C., Mutz, D., Valeur, F., and Vigna, G. On the detection of anomalous system call arguments. In *Proceedings of the 8th European Symposium on Research in Computer Security*, pages 326–343, 2003.

[12] MacQueen, J. B. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[13] Maggi, F., Matteucci, M., and Zanero, S. Detecting intrusions through system call sequence and argument analysis. *IEEE Transactions on Dependable and Secure Computing*, 99(1):1–15, 2009.

[14] 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 System Security*, 3(4):262–294, 2000.

[15] Pereira, C. M. M. and de Mello, R. F. Behavioral study of unix commands in a faulty environment. In *Proceedings of the 8th International Conference on Dependable, Autonomic and Secure Computing*, pages 1–6, 2009.

[16] Python. Python v2.7 documentation, 2010. `http://docs.python.org/`.

[17] Shannon, C. E. A mathematical theory of communication. *Bell System Technical Journal*, 27(1):379–423, 1948.

[18] Twycross, J. and Aickelin, U. Information fusion in the immune system. *Information Fusion*, 11(1):35–44, 2010.

[19] Warrender, C., Forrest, S., and Pearlmutter, B. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on security and Privacy*, pages 133–145, 1999.

[20] Zanero, S. and Savaresi, S. M. Unsupervised learning techniques for an intrusion detection system. In *ACM Symposium on Applied Computing - SAC 2004*, pages 412–419, 2004.