Anomaly Detection from Log Files Using Data Mining Techniques

Article *in* Lecture Notes in Electrical Engineering · January 2015

DOI: 10.1007/978-3-662-46578-3_53

2 authors, including:

Breier Jakub
Nanyang Technological University
45 PUBLICATIONS **103** CITATIONS

SEE PROFILE

A Dynamic Rule Creation Based Anomaly Detection Method for Identifying Security Breaches in Log Records

Jakub Breier*[1] and Jana Branišová†[2]

[1]Physical Analysis and Cryptographic Engineering,
Nanyang Technological University,
Singapore
[2]Faculty of Informatics and Information Technologies,
Slovak University of Technology, Bratislava,
Slovakia

Abstract

Evidence of security breaches can be found in log files, created by various network devices in order to provide information about their operation. Huge amount of data contained within these files usually prevents to analyze them manually, therefore it is necessary to utilize automatic methods capable of revealing potential attacks.

In this paper we propose a method for anomaly detection in log files, based on data mining techniques for dynamic rule creation. To support parallel processing, we employ Apache Hadoop framework, providing distributed storage and distributed processing of data. Outcomes of our testing show potential to discover new types of breaches and plausible error rates below 10%. Also, rule generation and anomaly detection speeds are competitive to currently used algorithms, such as FP-Growth and Apriori.

1 Introduction

Information systems and computer networks provide information about their state and operation in the form of log records. These records are composed of log entries containing information related to a specific event, which can be related to security [6]. Potential security breaches can be revealed by analyzing log files and looking for anomalies that occurred at a certain time

*jakub.breier@gmail.com
†branisovaj@gmail.com

during the device operation. Organizations today utilize complex information systems producing large amounts of log messages, making it infeasible to analyze them manually. Automatized methods, if implemented correctly, can help us to aim at important records, revealing malicious code, non-privileged system access or resources usage, policy breaches and also identify the source of these activities [13].

As stated in NIST SP800-92 [6], organizations should establish processes for log management, incorporate these processes into their policies and clearly define the goals and requirements for log management.

There can be various types of log messages, commonly inspected ones usually originate from operating systems, network devices, web servers and various applications using the network. SANS Institute published a report stating six most critical report categories that should be collected and analyzed in order to identify possible breaches[1]:

1. Authentication and Authorization Reports
2. Systems and Data Change Reports
3. Network Activity Reports
4. Resource Access Reports
5. Malware Activity Reports
6. Failure and Critical Error Reports

In our work we focus on the third category, but the method can be easily extended to other categories as well. Common types of security software capable of capturing these log are antimalware software, IDS/IPS, remote access software, web proxies, vulnerability management software, authentication servers, routers, firewalls, and network quarantine servers[6]. To test our method we use a set from IDS and a set of Snort logs created by analyzing this set. We use anomaly detection, with the help of data mining techniques.

Our approach utilizes a dynamic rule creation method for detecting possible breaches. A human intervention is minimized by detecting anomalies from the normal system behavior and also, it is possible to discover new types of breaches. A problem with large data sets is effectively handled by Apache Hadoop framework, which we have implemented using its *MapReduce* method. Hadoop-based algorithm processes data faster compared to algorithm using tree-based structure and the framework makes it easy to add several computing nodes for parallel log analysis.

The rest of the paper is organized as follows. Related work is stated in Section 2. Section 3 describes possibilities of anomaly detection, providing

[1] http://www.sans.edu/research/security-laboratory/article/sixtoplogcategories

overview of current methods in this field. Section 4 presents the design of our solution, followed by Section 5, where we state results of the testing. Finally, Section 6 concludes this work.

2 Related Work

There are several works proposing usage of data mining methods in log file analysis process or in detecting security threats in general.

One of the first approaches utilizing data mining techniques for intrusion detection was proposed by Lee and Stolfo in 1998 [7]. They implemented two algorithms for detecting algorithms, the association rules algorithm and the frequent episodes algorithm. They showed that by analyzing audit data it is possible to discover patterns of intrusions.

Schultz et al. [9] proposed a method for detecting malicious executables using data mining algorithms. They have used several standard data mining techniques in order to detect previously undetectable malicious executables. They found out that some methods, like Multi-Label Naive Bayes Classification can achieve very good results in comparison to standard signature-based methods.

Grace, Maheswari and Nagamalai [5] used data mining methods for analyzing web log files, for the purposes of getting more information about users. In their work they described log file formats, types and contents and provided an overview of web usage mining processes.

Frei and Rennhard [3] used a different approach to search for anomalies in log files. They created the Histogram Matrix, a log file visualization technique that helps security administrators to spot anomalies. Their approach works on every textual log file. It is based on a fact that human brain is efficient in detecting patterns when inspecting images, so the log file is visualized in a form that it is possible to observe deviations from normal behavior.

Fu et al. [4] proposed a technique for anomaly detection in unstructured system logs that does not require any application specific knowledge. They also included a method to extract log keys from free text messages. Their false positive rate using Hadoop was around 13% and using SILK around 24%.

Makanju, Zincir-Heywood and Milios [8] proposed a hybrid log alert detection scheme, using both anomaly and signature-based detection methods.

3 Detection Methods

According to Siddiqui [10], there are three main detection methods that are used for monitoring malicious activities: scanning, activity monitoring and integrity check. *Scanning* is the most widely used detection method, based

on searching for pre-defined strings in files. Advanced version of scanning includes heuristic scanning which searches for unusual commands or instructions in a program. *Activity monitoring* simply monitors a file execution and observes its behavior. Usually, APIs, system calls and hybrid data sources are monitored. Finally, *integrity checking* creates a cryptographic checksum for chosen files and periodically checks for integrity changes.

According to OWASP[2], there are four main types of information in every log record: when, where, who and what. It is possible to further process the log data and identify anomalies or correlations among different device logs only if all the four attributes are present.

Data mining is relatively new approach for detecting malicious actions in the system. It uses statistical and machine learning algorithms on a set of features derived from standard and non-standard behavior. It consists of two phases: data collection and application of detection method on collected data. These two phases sometimes overlap in a way that selection of particular detection method can affect a data collection.

Data can be analyzed either statically or dynamically. Dynamic analysis observes a program or a system during the execution, therefore it is precise but time consuming. Static analysis uses reverse-engineering techniques. It determines behavior by observing program structure, functionality or types of operation. This technique is faster and it does not need as much computational power as dynamic analysis, but we get only approximation of reality. We can also use hybrid analysis - first, a static analysis is used and if it does not achieve correct results, dynamic analysis is applied as well.

3.1 Detection Strategies

Every detection method includes a record analysis based on selected criteria. There are three categories of data detection:

- *Anomaly detection* - first, a system profile, functioning properly according to specification, is created. Then the method compares gathered data in order to check if a non-standard behavior can be labeled as an attack [12].

- *Misuse detection* - when using this method, a system profile containing samples of various malicious programs, is created. Then, an algorithm tries to find a match with this profile. It can complement the anomaly detection method, but it is unable to recognize new or unknown attack types.

- *Hybrid detection* - is a combination of last two strategies. There is no need of creating a profile, this detection strategy creates a data classifiers based on data from both safe and unsafe sources.

[2]https://www.owasp.org

3.2 Knowledge Discovery in Databases

Data mining is the analysis step of the Knowledge Discovery in Databases (KDD) [2]. The main goal of KDD is to discover information in large data sets. KDD consists of following stages:

- Data cleaning
- Data integration - combines multiple data sources
- Data transformation - data is transformed or consolidated in a form appropriate for data mining
- Data mining - applies data analysis and discovery algorithms that produce patterns over the data
- Pattern evaluation - identifies important patterns representing certain level of knowledge
- Knowledge interpretation - uses visualization and presentation techniques for knowledge representation

3.3 Data Mining

Data mining is a process of (semi-)automatic knowledge extraction. It is possible to apply this process on data in various forms, e. g. quantitative form, text form or multimedia form. The main advantage of data mining is a fast way of information gathering. When considering logging problem, data mining is applied after standard log analysis in order to provide outcome of better quality. In this work we use two main data mining algorithm types:

- *Classification* is a process of data mapping to previously defined categories. These categories have certain properties so that by examining the data it can be identified with some degree of accuracy to which category it belongs. The goal of text classification is to find an approximation of unknown function $\Phi : D \times C \rightarrow \{true, false\}$, where D is a set of text documents and $C = \{c_1, c_2, \ldots, c_{|C|}\}$ is a set of predefined categories. Function Φ holds true for $< d_i, c_j >$ if the document d_i belongs to category c_j. Function $D \times C \rightarrow \{true, false\}$ which approximates Φ is known as a *classifier*.

  There are several algorithms that could be used for this purpose. Tavallae et al. [11] made a comparative analysis of several algorithms for the log file data mining. According to their results, decision tree algorithms and Multi-Layer Perception method that makes a model of artificial neural network for positive feedback gained success rate of 90%. Naive Bayesian a posteriori classification had success rate of 80% and Support Vector Machine method achieved 60%.

- *Clustering* is a method that maps data into groups. These groups are not defined in the beginning, therefore we do not know their properties. A group, called the *cluster*, is defined by the data it contains. In the end, instances belonging to one cluster are very similar. This data mining algorithm can therefore help to reveal the natural structure of the analyzed data.

  If there are elements that do not belong to any cluster, we can usually observe significant differences with comparison to other data belonging to some cluster. These elements can reveal a non-standard system behavior, an anomaly. Such elements are called the *outliers*. Thanks to a capability of this algorithm to discover outliers, we can detect new or modified attack behavior that cannot be recognized by standard detection methods.

3.4 Evaluation of Data Mining Algorithms

In order to determine how well the result corresponds to reality, it is necessary to evaluate the implemented data mining algorithm. For each type of algorithm, a different evaluation method is used, because the properties of algorithms differ greatly. For example one algorithm can be very efficient in processing numerical data sets, other can be useful in sorting text documents into classes. This part provided an overview of an evaluation of two previously described data mining algorithm types. These methods can be also used for comparison of particular algorithms in the case of the same data set usage:

- *Classification evaluation:* evaluation of these algorithms can be done in several ways. Usually, an approach based on measuring *true positives, false positives, true negatives,* and *false negatives*, is used. A Receiver operating characteristic (ROC) can be used for this purpose, which is a curve created by plotting the *true positive rate* against the *false positive rate*. ROC for the ideal case and for a random guess is depicted in Figure 1.

- *Clustering evaluation:* for clustering algorithms, there are two main evaluation techniques. *Internal evaluation* means that the result is evaluated based on the data that was clustered itself. As examples of internal evaluation can be two methods – Davies-Bouldin index and Dunn index. On the other hand, *external evaluation* is based on the data that was not used for clustering, like class labels and external benchmarks. Examples of such methods are Rand meaure, F-measure, and Jaccard index.

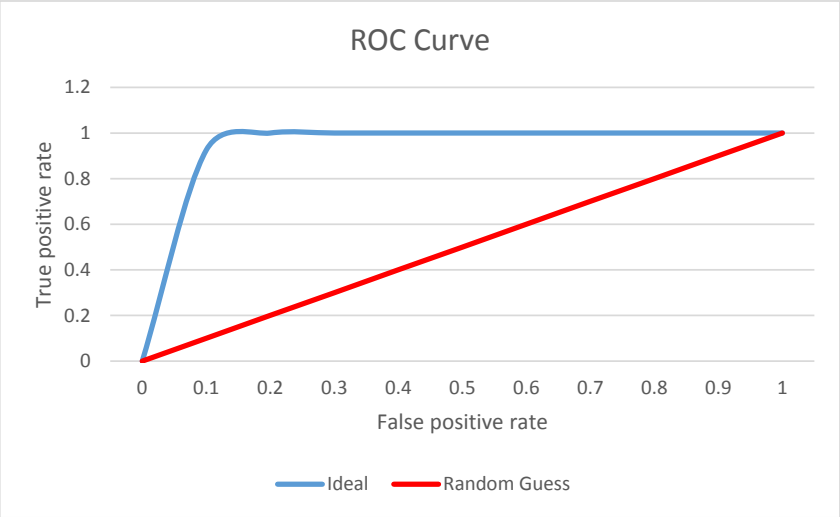


Figure 1: Receiver operating characteristic curve.

3.5 Parallel Processing

A huge amount of log data is generated every day and it is presumed that this amount will grow over time. Therefore, it is necessary to improve the process of the log analysis and make it more effective. We have chosen Apache Hadoop[3] technology for this purpose with the *MapReduce* [1] programming model. *MapReduce* is used for processing large data sets by using two functions. *Map* function processes the data and generates a list in the key-value form. *Reduce* function can be then used by the user for joining all the values with the same key. Hadoop architecture is based on distributing the data on every node in the system. The resulting model is simple, because MapReduce handles the parallelism, so the user does not have to take care about load balancing, network performance or fault tolerance. Hadoop Distributed File System can then effectively store and use the data on multiple nodes.

3.6 Static and Dynamic Rules

There are two general approaches for rule creation. A *static rule-based correlation* is based on static rules, defined by a user before the analysis. Another approach is a *dynamic rule creation*, using an algorithmic data mining

[3]http://hadoop.apache.org/

techniques to define rules. We will briefly describe both approaches in this subsection.

3.6.1 Static Rule-Based Correlation

First, we need to create a scenario of a situation we want to simulate. If, for example, there is a process running on one device and another process running on a different device at the same time and the combination of both induces a security breach, we have to design rules that model this scenario. On top of these rules we have to implement a correlation rules deciding whether the situation we are dealing with is an attack, or not. Rules can contain many parameters, e.g. time frame, pattern repeat, service type, port. Algorithm then checks the data from log files and finds the attack scenarios.

The main advantage of such approach is an ability to discover attacks by analyzing correlations and therefore, to reveal breaches which could be hard to detect. There are specific languages enabling creation of rules and also tools that provide effective and easy-to-use rule creation. For companies, it is easier to buy packages of predefined rules than employing many system analysts capable of creating organization-specific rules.

The main disadvantage of this approach is its high price, especially for the maintenance. Modelling of each attack scenario is a non-trivial task, there are many possibilities of executing the same attack type and sometimes those are non-deterministic. Also, attacks are evolving and new types are being created every day. That means, it is necessary to create new rules over time and even then there is still a chance there are some undiscovered unspecified attacks that can easily happen without noticing. There are companies providing Intrusion Detection Systems and Security Information and Event Management systems, together with periodic maintenance.

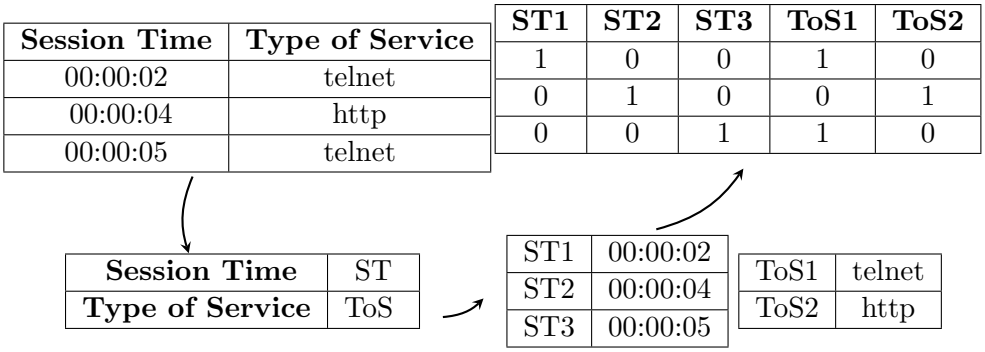
3.6.2 Dynamic Rule Creation

This approach is being utilized for anomaly detection for less than twenty years. Generated rules are usually in an *if-then* form. First, an algorithm creates patterns that can be further processed into a set of rules specifying which action should be taken.

Methods based on dynamic rule creation can solve the problem with a necessity of continual breach patterns update by searching for potential breaches that are not yet known. For example, in [9] authors implemented a data mining algorithm based on a *cross-validation* technique, achieving more than twice as better breach detection compared to current static pattern based methods.

The main disadvantage is a complexity of analyzing a high dimensional data, such as log files. There are algorithms capable of handling such data,

Table 1: Binary Transformation Example.



but usually the space and computational complexity is high. Therefore, it is necessary to reduce data dimensions as much as possible.

4 Design

The main idea for the design of our solution is to minimize false positives and false negatives and to make the anomaly identification process faster.

The steps of the algorithm are following. First, a testing phase is performed and rules are made from the testing data set. The outcome of this phase is an anomaly profile that will be used to detect anomalies in network devices log files. For creating rules, log file is divided into blocks instead of rows. A block is identified by the starting time, session duration and type of a service. We will use a term 'transaction' for particular block. This approach allows us to create a rule based on several log files from different devices or systems, so that one transaction can contain information from various sources. For creating uniform data sets, which can be processed by different algorithms, each transaction is transformed in a binary string form. For a spatial recognition of a log record in transaction, each record in the original log file will be given a new attribute - transaction ID. Creation of transactions and rules are depicted in Fig. 2.

For the detection program it is necessary to be able to process various log file formats, therefore we decided to use configuration files which will help to determine each attribute position.

4.1 Data Transformation

Data transformation includes creation of a new data set that contains the binary data only. The advantage of such data representation is ability to process it with various algorithms for association rules creation. Example of such a transformation is depicted in Table 1.

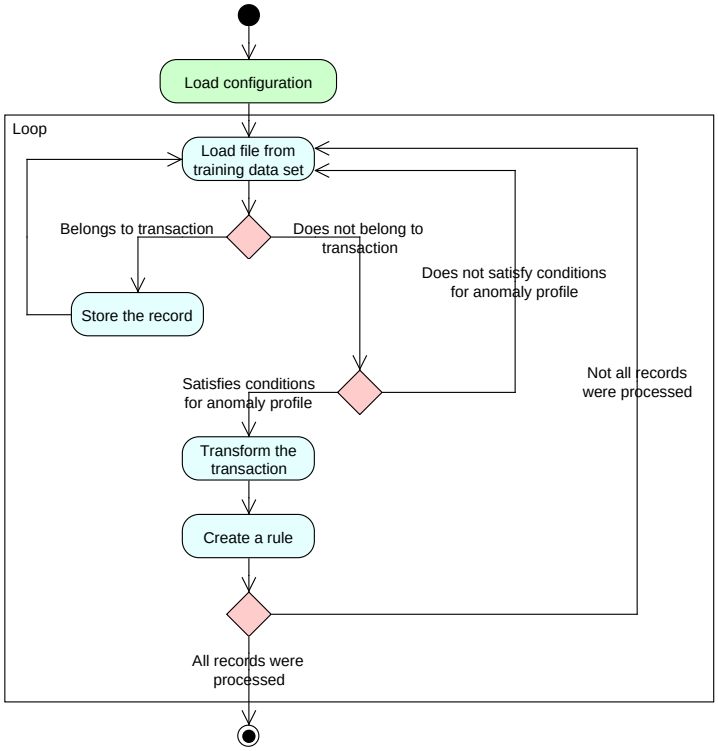


Figure 2: Transaction and Rule Creation.

To avoid a problem with large dimension number by using binary representation of log records, we propose a data reduction. This reduction is achieved by inserting values into categories and using an interval representative instead of a scalar or time value. Binary string contains a numerical value of 1 for values which are present in the record and a numerical value of 0 otherwise. However, some of the values are unable to reduce, such as IP addresses or ports.

4.2 Transaction and Rule Creation

Transaction and rule creation algorithm works as follows. It loads each record from log files line by line and stores them in the same block if they were created in the same time division, within the same session and if the IP addresses and ports are identical. If they can be identified as related, a transaction is created. Then it is decided whether this transaction fulfills conditions to be included in the anomaly profile. If yes, a new rule is created, if no, this transaction is ignored for the further rule creation process.

A rule contains attributes in a binary form that are defined in config-

uration file. It always contains some basic attributes related to time and session parameters and also a device ID, from which particular log record originates.

The anomaly finding algorithm first loads a set of previously created rules from the database. Then it sequentially processes the log files intended for analysis and creates transactions from these files. This transaction is then compared with the set of rules and if it is identified as an anomaly, it is stored for further observations.

4.3 Processing of Large Data Sets

As stated in Section 3.5, we decided to use Hadoop technology with MapReduce programming model to process large data sets. Hadoop enables us to easily add processing nodes of independent device types. After program starts, *JobTracker* and *TaskTracker* processes are started, which are responsible for *Job* coordination and for execution of *Map* and *Reduce* methods. *JobTracker* is a coordinator process, there exists only one instance of this process and it manages *TaskTracker* processes which are instantiated on every node. MapReduce model is depicted in Fig. 3, however in our case, only one *Reduce* method instance is used. First, a file is loaded from Hadoop Distributed File System (HDFS) and it is divided into several parts. Each *Map* method accepts data from particular part line by line. It then processes the line and stores them until a rule is created (if all the conditions are met). The rule is then further processed by the *Reduce* method, which identifies redundant rules and if the rule is unique, it is written into the HDFS.

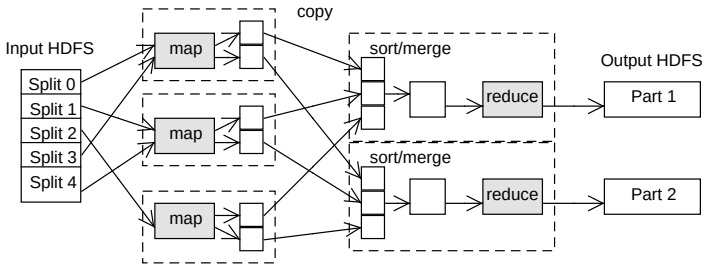


Figure 3: MapReduce Algorithm.

To allow nodes to access same files in the same time, but without loading them onto the each node separately, a Hadoop library for distributed cache is used[4].

[4]https://hadoop.apache.org/docs/r2.2.0/api/org/apache/hadoop/filecache/DistributedCache.html

5 Testing

Our anomaly detection method was implemented in Java programming language. For testing, a following setup was used:

- CPU: Intel Core i7-4500U (1.8-3.0 GHz, 4MB cache, 2 cores, 4 threads)
- RAM: 8 GB
- Operating system: Ubuntu 12.04

For testing purposes, two data sets were used: 1998 DARPA Intrusion Detection Evaluation Set[5], that was created by monitoring a system for two weeks, and Snort logs, created by analyzing the DARPA data set[6]. Snort logs contain information, if the attack was performed, or not. Based on that, we were able to determine if our anomaly detection method was able to successfully identify an intrusion, or not.

Testing was performed on a log records set of a size of 442 181 records. This set was made by merging DARPA and Snort data sets. We have split this data set into ten subsets for cross-validation purposes. In cross-validation, set is divided into subsets with similar sizes and each subset is used as many times as is the number of subsets. In each testing, one subset is used as a test set and the other subsets are used as training sets. For our validation, we split the main data set in a way that each subset contained log records from every day when monitoring was performed.

5.1 Data Transformation

After data sets merging, it was necessary to determine how many unique values are present in each table column. These values are stated in Table 2.

As we have already stated, high-dimensional data increases memory requirements of anomaly detection algorithm. It is possible to reduce some of the attribute values so that it can still be able to detect anomalies on a reduced set. We can analyze the 'Session Time' attribute and a process of reducing its values into intervals. These intervals are stated in Table 3.

Since the majority of records has a session time value 00:00:01, it was decided to take this value as a standalone interval. The same holds for value 00:00:02. Two other intervals cover longer time sessions, but since there are not many values present in each of these intervals, it was possible to make the reduction. Therefore, after reduction it was possible to change the range of values from 884 to 4 in this case, which enables significantly faster data processing.

[5]http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/
[6]https://www.snort.org

Table 2: Occurrence of Unique Values in Merged Data Set.

Attribute	Unique Values	Min	Max
Date	10	07/20/1998	07/31/1998
Time	63 299	00:00:00	23:59:59
Session Time	884	00:00:01	17:50:36
Service	4664	n/a	n/a
Source Port	38 637	-	65 404
Destination Port	7887	-	33 442
Source IP	1 565	000.000.000.000	209.154.098.104
Destination IP	2 640	012.005.231.198	212.053.065.245
Attack Occurred	2	0	1
Attack Type	47	n/a	n/a
Alert	61	n/a	n/a

Table 3: Intervals with Highest Number of Occurences.

Session Time	00:00:01	00:00:02	(00:00:02,01:00:00>	(01:00:00,18:00:00)
Occurrences	795 421	13 873	7 987	759

5.2 True Positive and False Positive Rates

After applying generated rules and observing anomalies among transactions, we have to check how many anomalies have been identified correctly. It is desirable to identify all the real anomalies and not to label normal communication as an anomaly at the same time. Therefore, we have checked our results using *true positive rate (TPR)* and *false positive rate (FPR)* formulas to get outcomes of our algorithms implemented in Java and Hadoop. These formulas are stated in Equations 1 and 2, where FP = false positives, FN = false negatives, TP = true positives, TN = true negatives.

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

Table 4 shows TPR and FPR for Java and Hadoop implementations after cross-validation for ten testing sets. Results show that Hadoop MapReduce technology helps to reduce false positives by 1% and increases true positives by 3%.

Table 4: True Positive and False Positive Rates for Java and Hadoop Implementations.

	Java		**Hadoop**	
	TPR	**FPR**	**TPR**	**FPR**
Set 1	0.846989	0.083069	0.855534	0.073358
Set 2	0.840526	0.072856	0.843952	0.06012
Set 3	0.616683	0.080336	0.843952	0.060124
Set 4	0.788934	0.068232	0.843952	0.060124
Set 5	0.866599	0.084820	0.869818	0.073270
Set 6	0.861434	0.080436	0.863636	0.072067
Set 7	0.834294	0.076534	0.838545	0.063471
Set 8	0.856624	0.074708	0.859672	0.063386
Set 9	0.840459	0.073108	0.841219	0.058621
Set 10	0.677185	0.078954	0.675759	0.068844
Overall	**0.802972**	**0.077305**	**0.833604**	**0.065339**

5.3 Error Rate

Overall error rate of the algorithm can be determined by Equation 3.

$$Error\ rate = \frac{FP + FN}{TP + FP + FN + TN} \tag{3}$$

The anomaly detection algorithm was implemented both in Java and in Hadoop. Table 5 shows values for both implementations. The table shows us that Hadoop implementation has around 1% lower error rate than Java implementation. Table results are also depicted in Figure 4.

5.4 Processing Speed

Important factor in anomaly detection is both speed of rules generation and speed of data processing. We compared our algorithm with two other anomaly detection algorithms, Apriori and FP-Growth. Apriori algorithm serves as a base for several rule-creation methods. Its disadvantage is that it needs to process the data set several times. FP-Growth algorithm uses tree-based storages for storing intermediate values. We used Weka libraries[7] for these algorithms implementations. Testing results are stated in Table 6. We can see that Hadoop implementation was the fastest among the tested algorithms. Therefore we can conclude that parallelization can bring very good results in terms of speed into the rule generation process.

Speed of data set processing for anomaly detection is stated in Table 7. We were comparing standard implementation in Java and implementation

[7] http://www.cs.waikato.ac.nz/ml/weka/

Table 5: Error Rate for Each Subset Using Java and Hadoop Implementations.

	Java	**Hadoop**
Set 1	0.095	0.087
Set 2	0.087	0.077
Set 3	0.144	0.142
Set 4	0.091	0.083
Set 5	0.093	0.083
Set 6	0.090	0.083
Set 7	0.090	0.080
Set 8	0.086	0.077
Set 9	0.087	0.077
Set 10	0.123	0.121
Overall	**0.098576**	**0.091465**

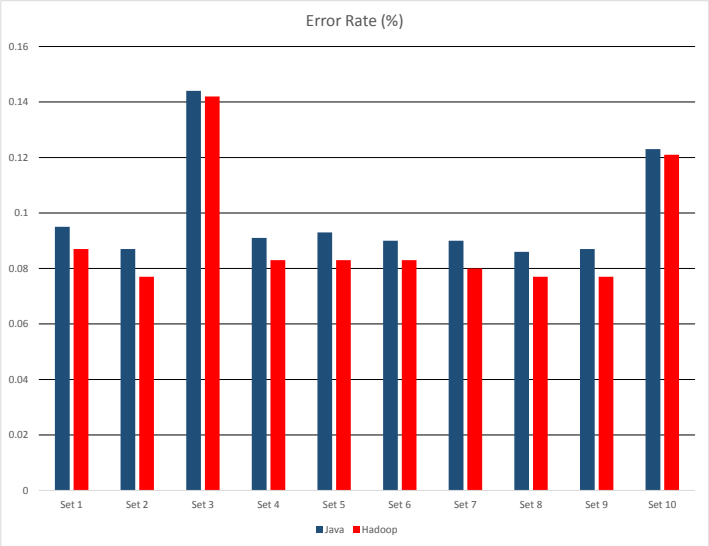


Figure 4: Error Rate for Different Subsets Using Java and Hadoop Implementations.

in Hadoop. Tests were performed on three data sets of sizes 10, 50, and 500 GB. As we can see, Hadoop can speed up this process more than ten times, even by using a single node. The Hadoop configuration was set to pseudo-distributed operation, which allowed it to run on a single-node. It

Table 6: Comparison of Rule Generation Speed.

Algorithm	Java Implementation	Hadoop Implementation	Apriori	FP-Growth
Time (s)	163.1	15.6	226	93

Table 7: Comparison of Anomaly Detection Speed.

Data Size	10 GB	50 GB	500 GB
Number of Records (in millions)	84	423	851
Java Implementation Time (s)	32 622	164 050	330 152
Hadoop Implementation Time (s)	3 164	13 531	29 042

is, of course, possible to add more nodes in order to improve throughput. We have tested a 10 GB data set on a three-node cluster, one node was configured as a master+slave, the other two nodes were configured as slaves only. Running time was 2040s, which gives us approximately 1.55 times better throughput than using a single-node.

6 Conclusion

In this work we have analyzed data mining methods for anomaly detection and proposed an approach for discovering security breaches in log records. Such approach generates rules dynamically from certain patterns in sample files and is able to learn new types of attacks, while minimizing the need of human actions.

Our implementation utilizes Apache Hadoop framework for distributed storage and distributed processing of data, allowing computations to run in parallel on several nodes and therefore, speeding up the whole process. Compared to our Java implementation, single-node cluster Hadoop implementation performs more than ten times faster. Another optimization was done in transformation of data into binary form, making it more efficient to analyze particular transactions.

For the future work, we would like to investigate possibilities of identifying correlations among several network devices with automatized methods.

Acknowledgement. This work was supported by VEGA 1/0722/12 grant entitled “Security in distributed computer systems and mobile computer networks.”

References

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[2] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Advances in Knowledge Discovery and Data Mining. chapter From Data Mining to Knowledge Discovery: An Overview, pages 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996.

[3] A Frei and M. Rennhard. Histogram matrix: Log file visualization for anomaly detection. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 610–617, March 2008.

[4] Q. Fu, J.-G. Lou, Y. Wang, and J. Li. Execution anomaly detection in distributed systems through unstructured log analysis. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, pages 149–158, Washington, DC, USA, 2009. IEEE Computer Society.

[5] L.K.J. Grace, V. Maheswari, and D. Nagamalai. Web log data analysis and mining. In Natarajan Meghanathan, BrajeshKumar Kaushik, and Dhinaharan Nagamalai, editors, *Advanced Computing*, volume 133 of *Communications in Computer and Information Science*, pages 459–469. Springer Berlin Heidelberg, 2011.

[6] Karen Kent and Murugiah P. Souppaya. Sp 800-92. guide to computer security log management. Technical report, Gaithersburg, MD, United States, 2006.

[7] Wenke Lee and Salvatore J Stolfo. Data mining approaches for intrusion detection. In *Usenix Security*, 1998.

[8] A Makanju, A.N. Zincir-Heywood, and E.E. Milios. Investigating event log analysis with minimum apriori information. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pages 962–968, May 2013.

[9] M.G. Schultz, E. Eskin, E. Zadok, and S.J. Stolfo. Data mining methods for detection of new malicious executables. In *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, pages 38–49, 2001.

[10] M. A. Siddiqui. *Data mining methods for malware detection*. ProQuest, 2011.

[11] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, pages 53–58, Piscataway, NJ, USA, 2009. IEEE Press.

[12] R. Venkatesan. A Survey on Intrusion Detection using Data Mining Techniques. *International Journal of Computers & Distributed Systems*, 2(1), 2012.

[13] R. Winding, T. Wright, and M. Chapple. System Anomaly Detection: Mining Firewall Logs. In *Securecomm and Workshops, 2006*, pages 1–5, Aug 2006.