EXPERIMENTAL ANALYSIS WITH BEHAVIOR RELIANCE INSIDER THREAT DETECTION MODEL

K. Venkateswara Rao\textsuperscript{1}, T. Uma Devi\textsuperscript{2}

\textsuperscript{1}Research Scholar, GIS-GITAM (Deemed to be University), Visakhapatnam, India.
\textsuperscript{2}Associate Professor, GIS-GITAM (Deemed to be University), Visakhapatnam, India.
\textsuperscript{1}venkateswara.gnt@gmail.com, \textsuperscript{2}uma@gitam.edu

Abstract

Malicious insiders are executing the severe attacks on cloud by misusing their privileges, which leads to the irreversible damages and loss of reputation. As the malicious insiders are authorized and integral part of the cloud, detecting and obstructing them to prevent the cloud from malicious attacks, became the complex and instantly focusable research aspect. An efficient “Insider Threat Detection Model” was proposed using the behavior reliance anomaly detection process. This paper elucidates Behavior Reliance Insider Threat Detection Model (BRITDM) implementation process and an empirical study was also conducted on the proposed model.

Amazon AWS modeled log file input records were used as input to detect the insider activities, using the proposed Behavior Reliance Anomaly Detection (BRAD) four layer architecture. Detailed user and admin activities were collected from the cloud log files that are represented in JSON format. JSQL Parser used for the query knowledge extraction and to create XML Tree. SVM classifier is trained with Compact Prediction Tree (CPT) structures knowledge starts with the comparison of admin executed activity query knowledge against the respective CPT structures of design level activity base, to determine whether the executed admin activity is malicious or not according to the BRAD four layered architecture. Cloud BRITDM processed 30 input records and resulted 5 as unique activities, 5 as abnormal, 2 as unintended suspicious activities and one as intended insider threat and reaming are normal activities. Experimental results shown the proposed BRITDM performed well in identifying the unique, abnormal, and suspicious and threats from insider activities.

Keywords: ITDM, BRAD Process flow, Anomaly Detection, Malicious Insider Threat Detection
I. Introduction

Malicious insiders are the authenticated members of a cloud organization, with the intention to execute the malevolent activities on cloud application platforms to tamper the data. Recent global surveys from IBM X-Force [VIII], Inside Threat Report [IX] and Ponemon Study [III] revealed that the victim organizations of malicious insiders are increasing day to day due to lack of monitoring, negligence and credential theft. According to them, 60% of the attacks were carried out by the insiders of an organization, which are executed in a series of suspicious activities. Among these insider attacks, 51% are unintentional or accidental, 47% are malicious or intended and 2% were in ambiguity to determine. Classification of malicious insiders and inadvertent insiders became a crucial task and the relevant strategies are suffering from the high value of false negatives in results.

In 2018, approximately $15.3 million has been spent, on several insider threat management strategies to protect their applications and data from insider attacks [X]. Another important factor to notice is, the insider activity is not a single step rather than it’s a series of abnormal steps in execution, for which it take minimum 30 to 70 days of time to complete [III]. Identity Spoofing, Data Tampering and disclosure of sensitive private information to rivals are the considerable severe damages happen to cloud due to failure in preventing the insider attacks. Majority of these insider attacks are planned and executed by the high privileged IT users or administrators of the same system [X], as they are well aware about the security flaws and escaping strategies of that system.

Admin job role is a superior job role, essentially designed with enormous privileges to ensure the satisfaction of software application clients and smooth running of software applications. Data Administrators, Application Administrators, Process Administrators, Hardware Administrators and Network Administrators are the frequently audible admin names. Like other application environments, the cloud environment is also equipped with the respective admin roles and each admin role is allocated to many employees. Installation, software configuration, verification, maintenance, complaints addressing and troubleshooting are the main responsibilities of administrators. Because there is no technical superior than admin and to avoid the barriers for him in trouble shooting, administrators are having the enormous privileges. These privileges are intentionally misused by some inside administrators to attack on systems, proven by the aforementioned attack incidents of identified in this paper. Identifying such hidden malicious entities and preventing the target clouds became a hot research topic today.

In order to mitigate the insider attacks CERT[IV] proposed a standard malicious insider threat detection abstract model is "prevent-detect-respond" (PDR model). Prevent phase suggests the policies, standards and laws to empower the cloud system to resist from insider activities. Detection phase mainly focus on identifying the hidden malicious Insiders of the system with the help of efficient monitoring system. Finally the respond phase is designed to perform the incident response activities, such as previous state restoration, damage reduction and operation rollbacks etc. By following the PDR model, the former research Scholars proposed the Rule-based
In 2019, Mr.K.Venkateswara Rao & Dr. T. UmaDevi proposed an efficient and reliable "Insider Threat Detection Model (ITDM)" to prevent the malicious insider attacks on cloud environment. This ITDM architecture is designed by following the CERT proposed PDR model. Like PDR model, the three modules of proposed ITDM also designed as comprehensive insider threat management architecture, to encounter the most sophisticated malevolent activities by insiders. The main goal of the ITDM architecture is identifying the insider attacks as early possible based on log file data analysis, preventing the cloud from malicious insider attacks and eliminating the result false positives with less processing burden.

To accomplish the log file data analysis, the main module of ITDM is appointed with BRAD model, which is transformed to “Behavior Reliance Insider Threat Detection Model (BRITDM)” with the capabilities of malicious activities detection and malicious insider’s identification. BRITDM analyses the series of an employee executed activities from log files and determines whether that employee is malicious insider or not. Unlike the former research models, BRITDM proposed a four layered architecture to precisely differentiate the malicious insiders from inadvertent employees. Compact prediction tree (CPTs), Algebraic query structures, Behavioral pattern mining, and anomaly detection algorithms are playing a vital role in BRITDM contained four layered threat detection architecture. Unique admin activity identification, abnormality detection, suspiciousness forecasting and threat confirmation are the four layers of proposed BRAD layered architecture.

To perform the experiments with proposed BRITDM, it has been implemented by cutting edge JAVA tools and technologies. Amazon AWS modeled log file input records were used as input to detect the insider activities using the proposed BRAD four layer architecture. Experimental results shown the proposed BRITDM performed well identifying the unique, abnormal, and suspicious and threat from insider activities.

Section-2 explains the BRITDM development associated technologies in brief, section-3 explains the BRITDM implementation flow process in detail and finally Section-4 conducts the empirical study of the research proposed work.

II. Tools and Technologies

This section is framed to explore the tools and technologies, which have been used with the projected BRITDM process. Although many tools and technologies were used at various levels only, the prominent ones are discussed in this section.

JSON: Java Script Object Notation (JSON) is a platform independent and light weight data exchanging format, which helps in transferring the data in the form of objects from one application to another. JSON data format is supported by most of the programming languages and platforms today, hence it is termed as universal data exchanging format with interoperability. Popular data generation tools like cloud logs, sensor outputs, web service responses are designed to be delivered in JSON.
format to support the global access. Present cloud vendors like AWS, Google, Microsoft and IBM are updated their log formats to JSON as part of the NIST SP800-92 guidelines. JSON supports individual objects, records, structures, lists and arrays officially.

JSON is less verbose, structural and simplified key-value based data exchange format, proven as the best data transmission model when compared with XML. As its structure annoys the model of independent plain objects, transformation from “JSON to Objects (of other OOP languages)” and “Objects to JSON” became feasible. Due to the flexibility and speed in data exchange, most of the process required data exchange needs are accomplishing today with the help of JSON only.

**Jackson**: Jackson library [XI] is the most popular and high speed JSON-JAVA and JAVA-JSON parsing technology, which is implemented with state-of-the-art JAVA technologies. Apart from JSON-JAVA conversion, it is also capable enough to parse various formats like XML, YAML and CSV to JSON. Configuration less auto mapping and Annotation based mapping features of Jackson made the mapping of “JSON to POJO” (Plain Ordinary Java Object) process feasible. To process the JSON formatted data, first it must be converted to respective programming language (i.e. JAVA) objects.

In BRITDM, the input log file is a JSON file, which consists of the data in the form of JSON array structure. Similarly while parsing the admin associated SQL queries using “Custom SQL Parser”, the parsed output is returned in XML format. In these both cases the Jackson parser is used to convert the Input log JSON data to Java and Custom SQL Parser XML output to JSON. Although some other JSON parsers like GSON and Moshi are available in market .,This paper deals based on Jackson parser as it supports the multi-dimensional conversion and auto mapping of JSON at high speed, compared to GSON, Moshi and other parsers.

**Custom SQL Parser**: Each record in the cloud input log file represents an activity executed by the admin. Activity means executing the SQL query on target database to accomplish the required data manipulation process. My research is able to thoroughly analyze and extract the knowledge from the executed SQL queries, it became possible to classify the normal and abnormal activities of an admin. Classifying normal and abnormal activities of an admin by thoroughly analyzing and extracting knowledge from the executed SQL queries is a research challenge. For this the SQL queries must be decoded and enwrapped with required metadata, to understand the information about the effecting target tables, columns, operations and conditions etc.

In order to parse the database vendor specific SQL queries to respective XML tree structures, a custom SQL parser has been designed using open source SQL parsers [XII]. This custom parser splits the SQL query statement into individual constructs and identifies the relations and dependencies among the constructs. After parsing the SQL statements, it returns the XML output with enwrapped metadata.
III. Behavior Reliance Insider Threat Detection Model (BRITDM)

Mr. K. Venkateswara Rao & Dr. T. Uma Devi proposed [XIII, XIV] Insider Threat Detection Model (ITDM) with Behavior Reliance Anomaly Detection (BRAD) process is finally unified as Behavior Reliance Insider Threat Detection Model (BRITDM). This BRITDM processes the input logs with different tools and technologies to identify the intended malicious behavior. This section describes the detailed flow of the proposed BRITDM, with respective process flow diagram as shown in fig.1.

![Fig. 1: BRITDM process flow diagram](image_url)

In cloud, all user/admin activities are recorded with detailed activity information on log files in a structural manner. Each log file contains several dozens of log records and each log record represents an activity by cloud user/admin, which is framed in a Java Script Object Notation (JSON) format. The proposed BRITDM utilizes these log records...
files as input data to analyze the behavior of cloud insiders. After receiving the input log file K, with a set activity records \( R_1, R_2, R_3 \ldots R_n \), is transformed into a java objects list D with objects \( P_1, P_2, P_3 \ldots P_n \), using the Jackson API Mappers as \( \{ K_{R_1} \rightarrow D_{P_1}, K_{R_2} \rightarrow D_{P_2}, \ldots K_{R_n} \rightarrow D_{P_n} \} \), to enable the input data records processing programmatically. Most popular and highly featured Jackson API [XI] is selected to complete this JSON to JAVA transformation process smoothly.

A new set called Admin Level Activity set, \( D_c = \{ A_{d_1}, A_{d_2}, \ldots A_{d_n} \} \) is created using each log record \( (P_x) \) by categorizing each admin_id property from the mapped object set D. This admin level clustering process helps us to validate each admin activities individually to find the anomalies. After clustering, each admin records and the corresponding executed SQL query statements will be extracted in an iterative manner for next phase query validations. These extracted SQL query statements are the main sources for validating the admin executed activity anomalies. Unfortunately the core syntax of SQL statements could be digested by the respective dialects only, but not by any others. To understand and classify the admin SQL statement by using supervised training classifiers (i.e. SVM), the statement must be decoded and knowledge should be extracted to relational algebraic structures. For this operation, the popular open source “JSQL Parser [XII]” has been used with a wide range of customization, to perform the query knowledge extraction. This decoding process creates an XML activity tree for each SQL statement and returns XML tree in Relational Algebraic format with a detailed query constructs information. This XML activity tree consists of SQL statement type, target tables, target columns, parameters, constraints, views, synonyms and sequences related information in extensible markup tags model. To avoid the processing of high verbose XML tags in query tree knowledge extraction process, this XML activity tree will be transformed to simple JSON format (explained in section-II) using JSON libraries, to speed up the knowledge extraction process. Now the JSON objects are ready for the admin executed query knowledge extraction process. Using JSON libraries this query knowledge will be extracted from JSON and placed into respective java object properties for future classifications.

K. Venkateswara Rao & Dr. T. Uma Devi [XIV] designed a SVM classifier model for future classifications with Activity based CPT structures. Each CPT structures contain the knowledge in the form of Prediction Tree (PT), Inverted Index (II) and Lookup Table(LT). The trained SVM classifier with CPT structures knowledge starts the comparison of admin executed activity query knowledge against the respective CPT structures of design level activity base is validated to determine whether the executed admin activity is malicious or not according to the BRAD proposed architecture. In this malicious activity detection process, first all queries of each admin will be examined for finding unique activities. These unique activities are treated as anomalies, when compared to the activities by the same user. Later the security measures and error codes regarding to the executed admin activities are validated to detect abnormal operations. For this client IP addresses, authentication policies, connecting user_agents and other network information is validated against actual expected information. Any failures at this stage will be announced as abnormal activities which will be sent to the suspicious detection process. Here the SVM
classifier will compare the admin executed query knowledge against the CPT provided Inverted Indexes to conclude the nature of admin activity using the three different possible cases.

Case 1: Activities based on Data Definition Language (DDL) but violation of admin role privileges is defined as **unintended suspicious activity** which may not reflect the effect on existing data currently but may effect in future.

Case 2: Activities based on Data Manipulation Language (DML) but violation of admin role privileges is defined as an **intended insider threat activity** that effects data in either read or write format.

Case 3: Activities in any other format that do not violate admin job privileges is defined as **valid activity** with some considerable warning to avoid future threats.

IV. **Empirical Study**

This section explores the implementation and experimental research details of the proposed BRITDM project. BRITDM execution processes sequential view snippets helps in understanding the input, output and the working nature of BRITDM.

**Environmental Setup:** In order to construct the BRITDM framework, several tools and technologies have been used as explored in section-II. Java Enterprise Edition-8 is used as main coding technology with its compatible java third party vendor licensed libraries like JSON, Jackson, Junit and jsqparser are utilized in this project. Eclipse IDE 2019-12 (R) is used for development of project, whereas Apache Tomcat 8.0 is utilized for the deployment of BRITDM project. JDK 1.8 (U-232) is installed as core java engine to execute java technology programs. All these software’s are alone existed on an Intel – i5 processor with RAM 8GB& Hard Disk 500 GB, which is running with a Windows-10 operating system.

Fig 2.1 gives the extracted log records, Fig 2.2 list the Java objects obtained from the JSON log records, Fig. 2.3 shows the clusters identified by BRITDM model based on admin_id property, Fig 2.4 list the activities of each cluster and finally Fig 2.5 shows the generated XML activity tree based on the knowledge extracted from queries.

Later these decoded clusters are transformed into Json object using JSON libraries, Figure 3.1 shows the created Json object of an admin activity executed SQL query. Similarly all records relevant XML activity trees are transformed to Json objects and these Json Objects will be transformed to Java objects using Jackson again. Now the processed query knowledge is available with each java object, which is extracted and shown in figure 3.2. At this level each query related target tables, columns, parameters, functions, operation types, access privileges and admin information is available to classifiers. Now the SVM classifier identifies the anomalies as unique activities and error prone activities as abnormal as shown in figure 3.3, with respective error messages even. Finally the SVM classifier validated all unique and abnormal activities against preloaded activity base contained Compact Prediction Trees (CPT). Any admin activity violates his/her job role privileges would be classified as either unintended insider suspicious activity or intended malicious insider threat as shown in figure 3.4

Copyright reserved © J. Mech. Cont.& Math. Sci.
K. Venkateswara Rao et al
In this way, BRITDM processed all 30 input records and resulted 5 as unique activities, 5 as abnormal, 2 as unintended suspicious activities and one as intended insider threat and remaining are normal activities.

Fig. 2: Snippets of BRITDM input log record to admin query decoding process
V. Conclusion

In order to identify the malicious insiders from cloud log files, we proposed a Behavior Reliance Insider Threat Detection Model earlier. This paper explores the implementation details and experimental analysis of proposed BRITDM framework in detail. As part of designing the BRITDM framework, each phase level input-

Fig. 3: Snippets of BRITDM decoded JSON to Threat result declaration

Copyright reserved © J. Mech. Cont. & Math. Sci.
K. Venkateswara Rao et al
process-outputs are explained using a detailed process flow diagram. This flow diagram represents the process from input selection to threat detection using several phases of framework. In order to implement the BRITDM framework Json Libraries, Jackson API and jsql-parser were mainly used as third party libraries. Each phase of design output is displayed as view snippets in this paper for easy understanding. The comprehensive project code flow and milestones were discussed in this paper along with experimental analysis. Empirical study confirms that the proposed BRITDM achieved all the targets and identified the malicious insider threat with high accuracy.

References

I. AWS CloudTrail: User Guide by Amazon AWS. Version-1, 2020, https://docs.aws.amazon.com/awscloudtrail/latest/userguide/awscloudtrail-ug.pdf

II. Bray, T. (2014). The JavaScript Object Notation (JSON) Data Interchange Format. RFC, 7158, 1-16

III. Cost of Insider Threats: Global Organizations,” https://www.observeit.com/ponemon-report-cost-of-insider-threats”

IV. Dawn Cappelli, Andrew Moore and Randall Trzeciak “The CERT Guide to Insider Threats”, Addison-Wesley, 2012 Pearson Education, Inc. http://ptgmedia.pearsoncmg.com/images/9780321812575/samplepages/9780321812575.pdf

V. Eberle, William & Holder, Lawrence & Graves, Jeffrey. (2010). Insider Threat Detection Using a Graph-Based Approach. Journal of Applied Security Research. 6. 10.1080/19361610.2011.529413.

VI. Greitzer, F. L., & Hohimer, R. E. (2011). Modeling human behavior to anticipate insider attacks. Journal of Strategic Security, 4(2), 25

VII. Gueniche T., Fournier-Viger P., Raman R., Tseng V.S. (2015) CPT+: Decreasing the Time/Space Complexity of the Compact Prediction Tree. In: Cao T., Lim EP., Zhou ZH., Ho TB., Cheung D., Motoda H. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2015. Lecture Notes in Computer Science, vol 9078. Springer, Cham.

VIII. IBM X-Force Threat Intelligence Index Report “https://www.ibm.com/security/data-breach/threat-intelligence”

IX. Isaac Kohen, “2018 Crowd Research Partners ’Insider Threat Report’: hopes and fears revealed”, 29 NOVEMBER 2017. http://crowdresearchpartners.com/wp-content/uploads/2017/07/Insider-Threat-Report-2018.pdf

X. Insider Threat Statistics for 2019: Facts and Figures :”https://www.ekransystem.com/en/blog/insider-threat-statistics-facts-and-figures”
XI. Jackson Project Home @github “https://github.com/FasterXML/jackson”

XII. Java Sql Parser, “http://jsqlparser.sourceforge.net/”.

XIII. K. Venkateswara Rao, Dr. T. Uma Devi “Architecture of Insider Threat Detection Model to Counter the Malicious Insider Threats on Cloud”, JASC: Journal of Applied Science and Computations - Volume 5, Issue 10, October/2018.

XIV. K. Venkateswara Rao, Dr. T. Uma Devi “Behavior Reliance Anomaly Detection with Customized Compact Prediction Trees” International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume-8 Issue-8, June 2019 https://www.ijitee.org/download/volume-8-issue-8/

XV. Kandias, Miltiadis & Virvilis, Nikos & Gritzalis, Dimitris. (2013). “The Insider Threat in Cloud Computing”. 6983. 93-103. 10.1007/978-3-642-41476-3_8.

XVI. P. Chattopadhyay, L. Wang and Y. Tan, "Scenario-Based Insider Threat Detection From Cyber Activities," in IEEE Transactions on Computational Social Systems, vol. 5, no. 3, pp. 660-675, Sept. 2018.

XVII. S. Ceri and G. Gottlob, "Translating SQL Into Relational Algebra: Optimization, Semantics, and Equivalence of SQL Queries," in IEEE Transactions on Software Engineering, vol. SE-11, no. 4, pp. 324-345, April 1985.