Journal of Computer Science

Original Research Paper

Periodic Service Behavior Strain Analysis-Based Intrusion Detection in Cloud

[1]S. Priya and [2]R. S. Ponmagal

[1]Department of Computer Science and Engineering, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India
[2]Department of Computing Technologies, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

Article history
Received: 16-08-2023
Revised: 24-10-2023
Accepted: 17-11-2023

Corresponding Author:
R. S. Ponmagal
Department of Computing Technologies, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India
Email: rsponmagal@gmail.com

Abstract: The problem of intrusion detection in cloud environments has been well studied. The presence of adversaries would challenge data security in the cloud by generating intrusion attacks towards the cloud data and should be mitigated for the development of the cloud environment. In mitigating intrusion attacks, there exist several techniques in the literature. The method uses different features like frequency of access, payload details, protocol mapping, etc. However, the methods need to improve to achieve the expected performance in detecting intrusion attacks. An efficient Periodic Service Behavior Strain Analysis (PSBSA) is presented to handle this issue. Unlike earlier methods, the PSBSA model analyzes the behavior of users in various time frames like historical, recent, and current spans. The model focused on identifying intrusion attacks in several constraints, not just considering the current nature. The performance of intrusion detection can be improved by viewing the user's behavior in historical, present, and recent timespan. Unlike other approaches, the proposed PSBSA model considers the user's behavior at different times in measuring the user's trust towards intrusion detection. Accordingly, the proposed PSBSA model analyzes the behavior of users under various situations. It examines the behavior in accessing the services at historical, current, and recent times. The method performs Historical Strain Analysis (HSA) Current Strain Analysis (CSA) and Recent Strain Analysis (RSA). HSA analysis is performed according to the historical data, CSA is performed based on the current access data and RSA is performed with the recent access data. The model estimates various legitimacy support values on each analysis to conclude the trust of any user. According to the support values, intrusion detection has been performed. The proposed PSBSA model introduces higher accuracy in intrusion detection in a cloud environment.

Keywords: Cloud, Cloud Security, Energy Efficiency, Intrusion Detection, Behavior Analysis

Introduction

The recent development in information technology has changed the shape of the informatics world. The organization has shifted its data to the cloud in order to maintain it. The cloud environment has become the most supporting entity for organizations to maintain and access their data without any hazards. The cloud service providers provide various services to access the data in the cloud and the registered users can access them with the required privilege. This supports the organizations from the difficulty of maintaining dedicated data servers, maintaining integrity, and investing higher costs.

In reality, the cloud is a loosely coupled environment and the service provider knows nothing about the client who accesses the service. Earlier, the user had been verified by a Third-Party Auditor (TPA) in restricting the service access and many challenges were identified, like leakage of data and so on. As with any environment, the cloud faces various security threats regarding data and service. Different threats can be named, which target the services and data. Among them, intrusion attacks are the most dominant ones targeting the service. The presence of such a threat must be detected to maintain the Quality of Service (QoS) of the environment because it would affect

the service throughput and introduce latency which would degrade the overall QoS performance. Any service deployed in the cloud has been designed to work on a specific goal and an intrusion attack is a malicious activity that targets the service and data. Any service has its nature or protocol, which takes clear input and returns the specific output. Any user generates an intrusion attack, whether from an insider or outsider. Whoever it is, they target the service throughput and data behind it. If the malicious user succeeds with the attack, they can steal the data and degrade the service integrity.

Cloud security has been viewed on different levels as data and service. Regarding data level security, separate access restriction approaches restrict malicious access. Also, various data encryption approaches are available to encode the data so the dedicated user can decode it. Similarly, other access restriction approaches are recommended by different researchers in terms of service level security. This article focuses on detecting intrusion attacks and discusses an efficient method to handle the issue.

The intrusion attacks can be detected in several ways by enforcing rigid access restriction algorithms. Many access restriction approaches restrict the access of malicious users according to their profile-which verifies whether the user concerned has access to the service but suffers from identifying the threat when the user submits malformed data to the service even though he has access to the service. Similarly, key-based approaches are used, which verify the key submitted before giving access to the service. This approach suffers from poor performance as the user would submit malicious data after clearing the trust check. Further, attribute-based techniques verify the access grant for different attributes accessed by the service and so on. However, the performance of the methods differs for each of them based on the process of measuring trust values. Because the trust-based approach verifies the user's trust based on different constraints, this article presents a novel time-orient service-centric behavioral strain analysis toward effective intrusion detection.

Problem Statement

The presence of malicious users in the environment would produce an intrusion attack. The malicious user would be an outsider or insider but they are capable of producing an intrusion attack. As the service provided would be accessed by various users and to identify the presence of intrusion attacks, it is necessary to monitor how the user accesses the service. By enforcing the adequate access restriction approaches, the user will be filtered in the initial stage, but when the service fails, it is necessary to find the intrusion attack and identify who generates the attack, or it is essential to conclude the presence of an intrusion attack. Energy efficiency is the primary concern in determining intrusion attacks suitable for loosely coupled service-orient environments. For example, to perform intrusion detection in mobile networks that access services through different devices, energy efficiency becomes a dominant entity that must be considered to improve intrusion detection.

Contribution

With the consideration to improve the performance of detecting intrusion attacks in the cloud, a novel model is described in this study. By analyzing the behavior of the user, you can identify the presence of an intrusion attack directly. You cannot directly say that the user has been involved in an intrusion attack straightaway. Even a genuine user would generate the request with incomplete features due to some mistakes in keyboarding. All these encourage the analysis of the behavior of the user to be performed in identifying the presence of an intrusion attack. Considering all these, the proposed PSBSA model analyzes the behavior in various strains to conclude the existence of intrusion attacks. Unlike earlier methods, the PSBSA model analyzes the behavior of users in various time frames like historical, current, and recent, which helps to achieve higher accuracy in intrusion detection. The model analyses the user behavior in the historical strain as HAS performs Recent Strain Analysis (RSA) and performs Current Strain Analysis (CSA) towards effective intrusion detection. Integrating the proposed model with the environment improves service performance and overall QoS performance has been hiked. The detailed approach is presented in this section.

Several articles have analyzed and discussed the problem of intrusion detection in the cloud. This section discusses some of the more popular and effective approaches for the problem considered.

An Artificial Intelligence (AI) based intrusion detection scheme is presented in Shrivastav and Dhawan (2018) which uses the firefly algorithm to optimize the data and uses a support vector machine to classify multi-class problems. A sequential minimal optimization technique is presented for intrusion detection, which uses k-means clustering for grouping similar records and applies Sequential Minimal Optimization (SMO) for classification (Chandra *et al*., 2019). An Intelligent Intrusion Detection Framework (IIDF) is presented in Urmila and Balasubramanian (2019) and uses header data and payload information in detecting intrusion attacks. An active learning-based threat detection approach is presented by Ramaiah *et al*. (2018) which continuously monitors the systems for their activities and applies artificial intelligence to detect the intrusion attack. The method uses the feedback values of different administrators in detecting the attack. A snort-based intrusion detection system is presented in Olanrewaju *et al*. (2018) which uses a set of rules to define the threats and

based on that, the method collects the sensor details to perform threat detection. The technique works towards various categories of attacks. The process uses multiple traffic information in classifying the threats. A deep learning model for an Intrusion Detection System (IDS) is presented by Almi'ani *et al*. (2018) focusing on unlearned attacks. The method continuously monitors the behavior of nodes to perform intrusion detection. A self-organized map-based intrusion detection scheme is presented by Ren *et al.* (2019) which uses hierarchical agglomerative clustering and sensitivity and consumption time in detection. A hybrid optimization technique with machine learning has been shown by Divyasree and Sherly (2018) to detect intrusion detection. The DO_IDS (Deep Organized Intrusion Detection System) algorithm uses isolation forest to eliminate the outliers and the genetic algorithm is used in feature selection. Also, the random forest has been used as a classifier. An ensemble Core Vector Machine (CVM) based intrusion detection scheme is presented in Vinayakumar *et al*. (2019) which uses ensemble Core Vector Machine (CVM) towards intrusion detection and is capable of detecting different threats. A neural network ensemble-based intrusion detection scheme is presented in Ludwig (2019) capable of detecting various threats.

A machine learning-based hybrid intrusion detection system is presented by Aljamal *et al*. (2019) which works based on monitored events. The possibilities are a set of network activities raised by various network activities. The method groups the events using the K means algorithm and performs a Support Vector Machine (SVM) for classification.

A trust-based game theoretical model is presented by Abusitta *et al*. (2018) for intrusion detection in the cloud. The method works according to the game theory and works collaboratively to perform intrusion detection. A behavior-based network intrusion detection scheme is presented by Ghanshala *et al*. (2018) which uses statistical learning and works according to the traffic behavior of different nodes in identifying the threat. A Cooperative Distributed Intrusion Detection System (C-DIDS) is presented by Ghribi *et al*. (2018) which uses the rates and delay features in real-time detection.

A self-adaptive genetic algorithm-based deep neural network IDS is presented by Chiba *et al*. (2019) which trains the network and uses a genetic algorithm in feature selection. The services provided by any network are accessed by various users who are eligible to face attacks at low rates. Low-rate attacks are handled and mitigated using the multi-threshold mechanism (Baskar *et al*., 2021).

A Virtual Machine Introspection (VMI) based security system (VM guard) is presented in Mishra *et al*. (2018) to support cloud systems that extract the features using term frequency and inverse document frequency approaches. The random forest has been used for classification. A Deep Blockchain Framework (DBF) is presented for intrusion detection by Alkadi *et al*. (2020) which uses smart contracts and blockchain in privacy preservation. The method uses a Bidirectional Long Short-Term Memory (BiLSTM) deep learning algorithm in threat detection.

A Vehicular-Edge Computing (VEC) based approach is presented by Mourad *et al*. (2020) for detecting intrusion attacks in vehicular networks. A collaborative IDS is presented by Tan *et al*. (2014) to handle vulnerabilities in accessing big data. Mishra *et al*. (2021) present Virtual Machine Introspection (VMI) to monitor granular attacks in system calls.

An efficient approach is designed to handle a variety of attacks (Nadeem *et al*., 2021) which monitors the devices for attacks and the rate of attacks. According to that, threat detection is performed and addresses the problem. An auto Isolation Forest (IF) based approach is presented in Sadaf and Sultana (2020) which uses deep learning Auto Encoder algorithm in feature extraction and isolation forest for feature selection. Further, the method performs binary classification according to the features selected.

A fuzziness-based semi-supervised learning scheme is presented for intrusion detection by Gao *et al*. (2018), which generates the ensembles using the labeled data. The fuzziness-based approach is used in classification. A Dew Computing as a Service (DaaS) for intelligent intrusion detection in Edge of Thing (EoT) ecosystems is presented by Singh *et al*. (2020). The method uses deep belief networks in classification. A secure approach to isolate malicious nodes according to the actual time transmission features is presented by Fatani *et al*. (2021).

Kasongo (2021) a Convolution Neural Network (CNN) based feature extraction scheme is presented, which also adapts short Search Optimization (TSO) for feature selection. The Transient Search Optimization Differential Evolution (TSODE) method uses a differential evolution algorithm in classification. A multi-level intrusion detection scheme is presented by Mishra *et al.* (2021) which performs intrusion detection in multiple levels to perform classification. Abdel-Basset *et al.* (2021) a genetic algorithm-based IDS for Industrial Internet of Things (IIoT) is presented and uses random forest for classification. The genetic algorithm is used for feature selection. In Sun and Grishman (2022) an introspection-based security approach called VMS held is was given, which uses the behavior of runtime processes. The method extracts the features of system calls and uses PSO and random forest classifiers in detecting the intrusion attack. An Intelligent Forensics System-based Deep learning model (Deep IFS) is presented for intrusion attacks in IIoT traffic. A residual connection between layers is designed to prevent information loss. The method performs intrusion

detection by distributing the traffic data across fog nodes and sharing it between the nodes. A Lexicalized Dependency Path (LDP) based training representation is presented, which handles the sparsity problem by using entity types and subtypes to refine the model. Supervised learning is used in classification.

All the methods discussed above suffer from poor performance in detecting intrusion attacks in a cloud environment. Accordingly, this article presents a novel Periodic Service Behavior Analysis (PSBSA) model for intrusion detection. The model differs from other approaches as it measures the user's trust by analyzing the user's behavior in different time domains like historical, recent, and current timestamps. This helps the proposed approach in improving the accuracy of intrusion detection.

Materials and Methods

The proposed Periodic Service Behavior Strain model (PSBSA) fetches the cloud access traces and performs preprocessing. The preprocessing involves identifying the features and eliminating the traces with noisy and missing features. Further, the traces are split into different time stamps and historical, recent, and current strains are analyzed. The method performs Historical Strain Analysis (HSA), Current Strain Analysis (CSA), and Recent Strain Analysis (RSA). HSA analysis is performed according to the historical data, CSA is performed based on the current access data and RSA is performed with the recent access data. Each strain analysis returns a legitimate score using which the method computes the legitimate score for the user. Based on the value of the legitimate score, the technique performs intrusion detection. The detailed approach is discussed in this section.

The architecture of the proposed PSBSA model has been pictured in Fig. 1 and the steps involved in the analysis are detailed in this section.

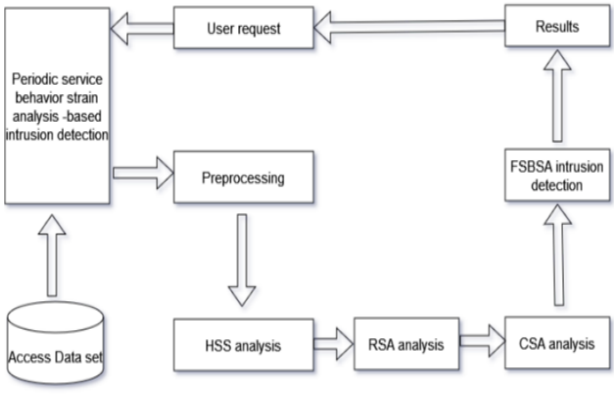

Fig. 1: Architecture of PSBSA intrusion detection model

Preprocessing

The cloud trace belongs to access performed by various users and is used in preprocessing. The service handler has fetched the trace, which finds the set of features present in entire traces. Further, each trace has been checked for all features identified. The traces which have missing features are removed from the trace set. Second, the method determines the set of all-time stamps in months, years, and weeks. The traces are split into several sub-sets according to the time stamps identified. The traces split based on the time stamp have been used to perform further analysis towards intrusion detection.

Consider the cloud access trace *CaT* contains a set of traces and then the method finds a set of all features in all the traces as follows:

$$Feature\ Set\ F\ Set\ =\ size(CaT)Fset \cup\ (\emptyset(Features \in CaT(i)) \cap Fset)\ i = 1 \quad (1)$$

Now, each trace *Ti* belongs to *CaT* is verified for the containment of features present in *F Set* as follows:

$$Flag = size(F\ Set)\ if Ti \in Fset(i) then\ ?\ 1:0\ i = 1 \quad (2)$$

The above equation checks the access trace *Ti* for the containment of all the features of *the F set*, if the feature is not present in the trace *Ti*, then the equation returns 0 otherwise, it will return a value of 1. so that the flag value has been used as the result of verification towards the selection of trace. The flag value has been used to verify whether the feature is available in the trace. If the flag is 0, the trace does not contain the feature. Otherwise, it has the feature. This supports noise removal in preprocessing.

If the value returned by Eq. (2) is 0, then the trace *Ti* will be removed from the access trace set *CaT*.

Further, the set of time stamps present in the trace *CaT* is identified as follows:

$$Time\ stamp\ set\ Ts = \mathrm{T} \cup (\emptyset(Time\ stamp \in CaT(i)) \cap Ts \quad (3)$$

//The time stamp is identified according to the number of months, years, weeks, and days. You can split the trace into any time stamp according to available traces.

Now, the traces *CaT* has been split into several time stamp trace *TsT* as follows:

$$Initialize\ time\ stamp\ set.TsT\ = size(Ts)\ Initialize(set(Ts(i))\ i = 1 \quad (4)$$

Finally, split the traces according to time stamp as follows: For example, the traces belonging to time stamp Ts_i are identified as follows:

$$TsT(Tsi)\ =\ Size(CaT)\ TsT\ \cup (CaT(i).\ Time\ Stamp = Tsi)\ i = 1 \quad (5)$$

The time stamp traces generated are used to analyze various strains to perform intrusion detection. The traces contain the time of generation, which has been used to split the traces under multiple timestamps. Also, the entire duration has been divided into months, weeks, and years. This is how the traces are separated according to the time stamp. The cloud access trace would contain several features like the time of access, the service accessed, the status of the service, a payload of data submitted, the completeness of the service, the user who accessed the service, and so on. The method first identifies what the features present in the trace overall are. Accordingly, the traces with incomplete features are removed and the traces are split according to the time they were created to support the analysis in various forms. To keep the research, the traces are divided under different time stamps when they are made. So, by reading the time stamp traces belonging to various time stamps, the method performs other analyses to measure the user's trust towards intrusion detection.

Historic Strain Analysis (HAS)

The historic strain analysis algorithm measures the Historical Legitimacy Support (HLS) based on the behavior of the user in accessing the service with genuine. To calculate the value of HLS, the method considers the service access with two constraints: One is honest in following the protocol and the other is how the user supports the service growth. According to this, the method first measures the value of Protocol Support (PS) based on the number of times the user accessed the service and the number of times the user followed the protocol. Similarly, according to corollary two, the Service Growth Support (SGS) is measured based on the number of times the service is accessed and several times the service is completed. Using these two values, the method computes the value of HLS to support intrusion detection.

Pseudocode

Give : Time stamp trace *TST*, Time stamp *set Ts*
Obtai : HLS
Start
Read *TST, Ts*
For each timestamp Ts_i:

$$Compute\ Protocol\ Support\ Ps\ = \frac{Size(TST(Tsi))\ Count(TST(Tsi(i)).ProtocolState==1)\ i=1}{size(TsT(Tsi))} \quad (6)$$

$$Compute\ Service\ Growth\ Support\ (SGS)\ = \frac{Size(TST(Tsi))\ Count(TST(Tsi(i)).CompleteState==1)\ i=1}{size(TsT(Tsi))} \quad (7)$$

End:

$$Compute\ HLS = \frac{Size(Ts)\sum_{i=1}^{n} Ts(i).Ps}{size(Ts)} \times \frac{Size(Ts)\ \sum_{i=1}^{n} Ts(i).SGS}{size(Ts)} \quad (8)$$

Stop

The historic strain analysis approach measures the protocol support and service growth support the user delivered at various time stamps to measure the value of HLS. As the traces are split under different time stamps and based on the total time stamps identified, the method finds the traces produced at the specific time stamp. For the traces of timestamp, the process computes Protocol Support (PS) and Service Growth Support (SGS) to support the measurement of HLS. Protocol support is the measure that represents the trust of the user in following the service protocol. Service growth support is the value measured based on the number of times the user completed the service for a specific number of service accesses. The historic strain analysis algorithm computes the value of Protocol Support (PS) and Service Growth Support (SGS) to add the value of HLS for the user. According to the value of HLS, the method would perform intrusion detection.

Recent Strain Analysis (RSA)

The recent strain analysis algorithm considers the frequency of service access made by any user at various and current time stamps. Also, the analysis finds the payload support at different time stamps and recent time stamps. Similarly, the method considers the protocol strain in other and current time stamps. The method computes the Frequency Strain Value (FSV) based on the value of the frequency of access at historical and recent times. Similarly, the Payload Strain Value (PSV) is measured based on the mean payload at historical times and the mean payload at current times. The Frequency Strain Value (FSV) is the measure that represents the access frequency of the user towards the service and the number of times it has been going malicious. It is measured based on the number of times the user accessed the service in recent times for the overall number of times and several times it has been identified as malicious at the current time and the number of times it has been identified as malicious in general times.

Similarly, the Payload Strain Value (PSV) is measured based on the number of times the user followed the payload metrics in the recent time for the number of times the user tracked in overall time. The number of times it has been identified as malicious in the current time with several times it has been identified as malicious in overall time. Using these values, the method computes the value of Recent Legitimate Support (RLS) to support intrusion detection.

Pseudocode

$$Compute\ Frequency\ Strain\ Value\ FSV = \frac{\frac{Size(TST(TsR))\ Count(TST(TsR(i)).Service==S)\ i=1}{size(TsT(TsR))}}{\frac{Size(TST(Tsi))\ Count(TST(Tsi(i)).Service==S)\ i=1}{size(TsT(Tsi))}} \times \frac{\frac{Size(TST(TsR))\ Count(TST(TsR(i)).Service==S\ \&\&\ State==Malicious)\ i=1}{size(TsT(TsiR))}}{\frac{Size(TST(Tsi))\ Count(TST(Tsi(i)).Service==S\ \&\&\ State==Malicious)\ i=1}{size(TsT(Tsi))}} \quad (9)$$

where, *TSR* is the recent trace set, the sub-set of the trace set includes traces of little time stamps generated in recent times.

Compute Payload Strain value PSV:

$$PSV = \frac{\frac{Size(TST(TsR))\ Sum(TST(TsR(i)).Payload)\ i=1}{size(TsT(TsR))}}{\frac{Size(TST(Tsi))\ Sum(TST(Tsi(i)).Payload)\ i=1}{size(TsT(Tsi))}} \times \frac{\frac{Size(TST(TsR))\ Suim(TST(TsR(i)).Payload==S\ \&\&\ State==Malicious)\ i=1}{size(TsT(TsiR))}}{\frac{SumSize(TST(Tsi))\ (TST(Tsi(i)).Service==S\ \&\&\ State==Malicious)\ i=1}{size(TsT(Tsi))}} \quad (10)$$

End:

$$Compute\ RLS\ = \frac{Size(Ts)\ \sum_{i=1}^{n} Ts(i).FSV\ i=1}{size(Ts)} \times \frac{Size(Ts)\ \sum_{i=1}^{n} Ts(i).PSV}{size(Ts)} \quad (11)$$

Stop

The recent strain analysis algorithm measures the user's trust in submitting proper payload to the service point by computing the payload strain value and the value frequency strain value FSV to represent the user's trust in the frequency of access. By analyzing a user's trust through recent strain analysis, the user behavior in recent times can be identified and how the user behaves in recent times can be monitored. This would help to determine the malicious user most concretely. As the method computes the Payload Strain Value (PSV) and Frequency Strain Value (FSV) to measure the value of RLS, the presence of intrusion detection can be identified effectively.

Current Strain Analysis (CSA)

The current strain analysis algorithm monitors the user's behavior toward the legitimacy of accessing the service based on the user's access profile and the access grant for the user. According to that, the method computes Grant Strain Value (GSV). The GSV value is measured based on the average contribution on service access identified from the current time trace. The average grant placed on service access with the overall trace. Similarly, the method computes the user's Malformed Strain Value (MSV) towards the service in historical and current times. The value of MSV is measured by calculating the average malicious request produced by the user at the present trace with the average malicious access created by the user at the overall time trace. Using both these values, the method computes the value of CSA to support the detection of intrusion attacks.

Pseudocode

Given : Time stamp Trace *TsT*, Time stamp set *Ts*
Obtain: CLS
Start

$$Compute\ Grant\ Strain\ Value\ (GSV) = \frac{\frac{Size(RTS)\ \sum_{i=1}^{n} RTS(i).Service==S\ \&\&\ RTS(i).Grant==False}{Size(RTS)}}{\frac{Size(TsT)\ \sum_{i=1}^{n} TST(i).Service==S\ \&\&\ TST(i).Grant==False}{Size(TST)}} \quad (12)$$

$$Compute\ Malicious\ Strain\ Value\ (MSV) = \frac{\frac{Size(RTS)\ \sum_{i=1}^{n} RTS(i).Service==S\ \&\&\ RTS(i).State==Malicious}{Size(RTS)}}{\frac{Size(TsT)\ \sum_{i=1}^{n} TST(i).Service==S\ \&\&\ TST(i).State==Malicious}{Size(TST)}} \quad (13)$$

$$Compute\ CLS = GSV \times MSV \quad (14)$$

Stop

The current strain analysis algorithm measures *GSV* and *MSV* values according to the strain values computed over malicious access and the classified values of the service access. The value of *GSV* is calculated based on the number of grants given to the user and the number of features accessed by the user. Similarly, the value of *MSV* is measured according to the number of malicious requests generated by the user among the number of malicious requests identified. Using both values, the method computes the value of *CLS* to support intrusion detection. By analyzing the user's trust in the current time stamp, like the current week, current month, and current year, the performance of intrusion detection can be identified effectively. It shows how the user behaves at present, which speaks to the user's trust in the most concrete way.

FSBSA Intrusion Detection

The proposed model performs intrusion detection by performing strain analysis at the different time stamps. To achieve this, the access trace has been used and preprocessed initially. Further, the method performs historical strain analysis, recent strain analysis, and current strain analysis. Each analysis returns a legitimate score to support intrusion detection. The HAS algorithm returns the value of HLS. The RSA algorithm returns RLS and the CSA algorithm returns CLS. Using these values, the method computes the value of Legitimacy Support (LS) based on the classification of user requests. The Legitimacy Support (LS) represents the user's trustworthiness in accessing the service and it has been measured by computing LS based on the results of various strain analyses.

Pseudocode

Give : Cloud access Trace *CaT*, service request *SR*
Obtain : Boolean
Start
Read *CaT* and *SR*.

Service Requested sreq	=	Request ∈ *SR*
Time Stamp Trace *TST*	=	perform preprocessing (*CaT*)
HLS	=	Perform historical strain analysis (*TST*, time stamp set *Ts*)
RLS	=	Perform recent strain analysis (*TST*, *TS*)
CLS	=	Perform current strain analysis (*TST*, *TS*)

$$Compute\ LS = \frac{RLS}{HLS} \times CLS \quad (15)$$

If *LS*>*Th*, then
Return true
Else
Return false
End
Stop

The above-discussed algorithm performs various strain analyses to compute different legitimacy support values. Based on the values obtained, the method computes the legitimate support to perform intrusion detection. The threshold value is set according to the frequency of the threat identified and has been adjusted in a dynamic manner. The threshold for decision-making is computed dynamically according to the rate of threat determined. If there is a higher threat ratio, then the value of the threshold will be higher. Otherwise, it will result in a moderate level.

Results and Discussion

The proposed Periodic Service Behavior Strain Analysis (PSBSA) model-based intrusion detection scheme has been implemented and evaluated for its performance under various constraints. The model has been implemented with Python and for the performance evaluation, the data set collected in Microsoft Azure has been used. The model has used the access trace maintained by different cloud models and environments to perform the analysis. The performance of the model has been measured on various parameters and compared with the results of other approaches. This section details the results obtained and the parameters considered for the evaluation presented in Table 1.

The constraints considered for the evaluation of the proposed approach are Table 1. The methods are evaluated for their performance on the following factors.

Table 1: Evaluation details

Parameter	Value
Tool used	Python
Data set	Amazon
Platform	Microsoft azure
Total records	1 million
No of services	100
No of users	5000

Table 2: Analysis of intrusion detection accuracy

	Intrusion detection accuracy %		
No of records	3 Lakhs	5 Lakhs	10 Lakhs
GA-RF	69	74	79
VMShield	73	77	82
Deep-IFS	79	85	87
PSBSA	87	93	98

Intrusion Detection Accuracy

The method's accuracy in finding the presence of an attack is measured according to the total number of attacks generated and several instances detected successfully. Also, it has been measured based on the True Positive (TP) and True Negative (TN) classifications. It has been measured as follows:

$$IDA = \frac{TP + TN}{Total\ Attacks} \times 100 \quad (16)$$

The performance in intrusion detection is measured for various approaches according to the traces available. In each test case, the performance is computed and compared with multiple methods in Table 2. The proposed PSBSA approach achieves higher intrusion detection accuracy than other techniques. The increasing number of traces would greatly help intrusion detection performance. The performance of intrusion detection is directly proportional to the number of traces in the set.

The performance of various methods at detecting intrusion attacks is measured with multiple records in the data set; it's shown in Fig. 2. The proposed PSBSA model achieves higher performance in intrusion detection than other techniques. As the proposed PSBSA model performs intrusion detection by analyzing the behavior of users in different time stamps in accessing the services, the PSBSA model achieves higher intrusion detection accuracy.

False Ratio in Intrusion Detection

The false ratio in intrusion detection is the measure that shows the false classification of different requests. It is measured according to the number of True Negative (TN) and False Positive (FP) classifications made by the algorithm. It has been calculated as follows:

$$FRID = \frac{FP + TN}{Total\ Attacks} \times 100 \quad (17)$$

The performance of methods in false classification or false ratio in detecting the intrusion detection attack is measured by the availability of a different number of traces in the data set. In each test case, the performance is measured for various approaches and compared in Table 3. The proposed PSBSA has produced less false ratio performance in intrusion detection than other approaches. The false detection ratio gets reduced with the increased number of traces in the set.

The performance of various methods at a false ratio in intrusion detection is measured and compared in Fig. 3. The proposed PSBS model introduces less false detection than other techniques.

Time Complexity

The time complexity of the intrusion detection scheme is measured according to the total time taken by the approach in detecting the intrusion attack.

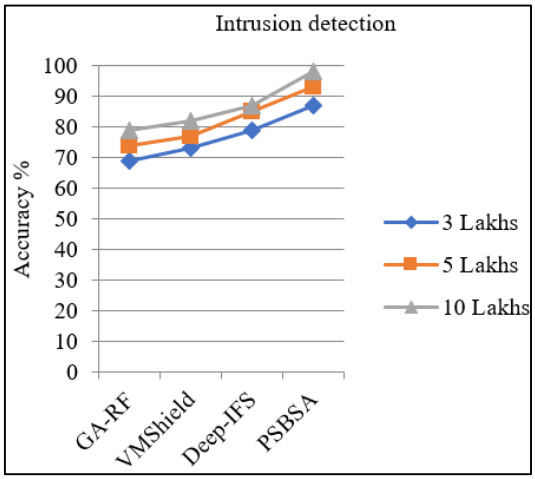

Fig. 2: Analysis of intrusion detection accuracy

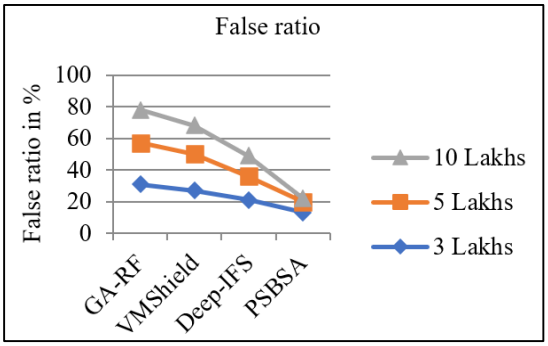

Fig. 3: Analysis of false ratio in intrusion detection

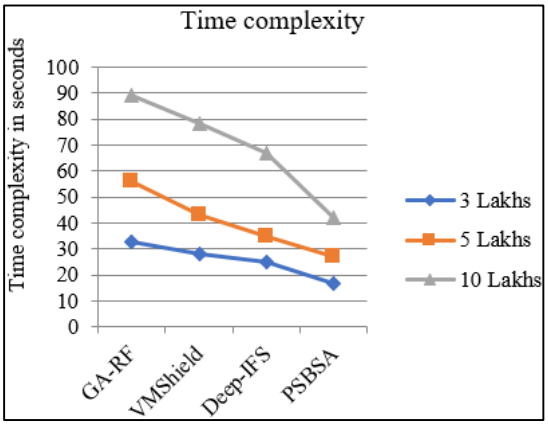

Fig. 4: Analysis of time complexity in intrusion detection

Table 3: Analysis of false ratio in intrusion detection

	False ratio in intrusion detection %		
No of records	3 Lakhs	5 Lakhs	10 Lakhs
GA-RF	31	26	21
VMShield	27	23	18
Deep-IFS	21	15	13
PSBSA	13	7	2

Table 4: Analysis of time complexity in intrusion detection

	Time complexity in intrusion detection in seconds		
No of records	3 Lakhs	5 Lakhs	10 Lakhs
GA-RF	33	56	89
VMShield	28	43	78
Deep-IFS	25	35	67
PSBSA	17	27	42

The value of time complexity introduced by various approaches in intrusion detection has been measured and presented in Table 4. The PSBSA model introduces less time complexity in all the cases. The increasing number of traces does not significantly increase the time complexity and only has a negligible effect.

The performance of time complexity in intrusion detection has been measured for different methods with different records calculated and presented in Fig. 4. The proposed PSBSA model introduces less time complexity than other approaches.

Conclusion

This study presented a novel Periodic Service Behavior Strain Analysis (PSBSA) towards intrusion detection in a cloud environment. The proposed model analyzes the user's behavior in accessing the service at different historical, recent, and current times. According to the analysis values, the method computes the value of legitimate support for the user. Based on the importance of legitimate support, the process performs intrusion detection. The inclusion of behavior analysis in various time stamp support the proposed PSBSA model to achieve higher performance in intrusion detection by up to 97% with the least time complexity.

Acknowledgment

Thank you to the publisher for their support in the publication of this research article. We are grateful for the resources and platform provided by the publisher, which have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team in reviewing and editing our work, and we are thankful for the opportunity to contribute to the field of research through this publication.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

S. Priya: Behavior strain analysis, interpretation of data drafted the article.

R. S. Ponmagal: Reviewed article critically for significant intellectual content and gave final approval of the version to be submitted and any revised version.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and that no ethical issues are involved.

Conflicts of Interest

The authors declare that they have no conflicts of interest to report regarding the present study.

References

Abdel-Basset, M., Chang, V., Hawash, H., Chakrabortty, R. K., & Ryan, M. (2020). Deep-IFS: Intrusion detection approach for industrial internet of things traffic in fog environment. *IEEE Transactions on Industrial Informatics*, *17*(11), 7704-7715. https://doi.org/10.1109/TII.2020.3025755

Abusitta, A., Bellaiche, M., & Dagenais, M. (2018). A trust-based game theoretical model for cooperative intrusion detection in multi-cloud environments. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 1-8. IEEE. https://doi.org/10.1109/ICIN.2018.8401625

Aljamal, I., Tekeoğlu, A., Bekiroglu, K., & Sengupta, S. (2019). Hybrid intrusion detection system using machine learning techniques in cloud computing environments. In *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)* (84-89). IEEE. https://doi.org/10.1109/SERA.2019.8886794

Alkadi, O., Moustafa, N., Turnbull, B., & Choo, K. K. R. (2020). A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks. *IEEE Internet of Things Journal*, *8*(12), 9463-9472. https://doi.org/10.1109/JIOT.2020.2996590

Almi'ani, M., Ghazleh, A. A. Rahayfeh A. A., & Razaque, A. (2018). Intelligent intrusion detection system using clustered self-organized map in *Proc. 5th International Conference on Software Defined Systems (SDS)*. https://doi.org/10.1109/SDS.2018.8370435

Baskar, M., Ramkumar, J., Karthikeyan, C., Anbarasu, V., Balaji, A., & Arulananth, T. S. (2021). Low rate DDoS mitigation using real-time multi threshold traffic monitoring system. *Journal of Ambient Intelligence and Humanized Computing*, 1-9. https://doi.org/10.1007/s12652-020-02744-y

Chandra, A., Khatri, S. K., & Simon, R. (2019). Filter-based attribute selection approach for intrusion detection using k-means clustering and sequential minimal optimization techniq. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 740-745. IEEE. https://doi.org/10.1109/AICAI.2019.8701373

Chiba, Z., Abghour, N., Moussaid, K., El Omri, A., & Rida, M. (2019). A clever approach to develop an efficient deep neural network based IDS for cloud environments using a self-adaptive genetic algorithm. In *2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*, 1-9. IEEE. https://doi.org/10.1109/COMMNET.2019.8742390

Divyasree, T. H., & Sherly, K. K. (2018). A network intrusion detection system based on ensemble CVM using efficient feature selection approach. *Procedia Computer Science*, *143*, 442-449. https://doi.org/10.1016/j.procs.2018.10.416

Fatani, A., Abd Elaziz, M., Dahou, A., Al-Qaness, M. A., & Lu, S. (2021). IoT intrusion detection system using deep learning and enhanced transient search optimization. *IEEE Access*, *9*, 123448-123464. https://doi.org/10.1109/ACCESS.2021.3109081

Gao, Y., Liu, Y., Jin, Y., Chen, J., & Wu, H. (2018). A novel semi-supervised learning approach for network intrusion detection on cloud-based robotic system. *IEEE Access*, *6*, 50927-50938. https://doi.org/10.1109/ACCESS.2018.2868171

Ghanshala, K. K., Mishra, P., Joshi, R. C., & Sharma, S. (2018). BNID: a behavior-based network intrusion detection at network-layer in cloud environment. In *2018 1st International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 100-105. IEEE. https://doi.org/10.1109/ICSCCC.2018.8703265

Ghribi, S., Makhlouf, A. M., & Zarai, F. (2018). C-dids: A cooperative and distributed intrusion detection system in cloud environment. In *2018 14th International Wireless Communications and Mobile Computing Conference (IWCMC)* 267-272. IEEE. https://doi.org/10.1109/IWCMC.2018.8450478

Kasongo, S. M. (2021). An advanced intrusion detection system for IIoT based on GA and tree-based algorithms. *IEEE Access*, *9*, 113199-113212. https://doi.org/10.1109/ACCESS.2021.3104113

Ludwig, S. A. (2019). Applying a neural network ensemble to intrusion detection. *Journal of Artificial Intelligence and Soft Computing Research*, *9*(3), 177-188. https://doi.org/10.2478/jaiscr-2019-0002

Mishra, P., Aggarwal, P., Vidyarthi, A., Singh, P., Khan, B., Alhelou, H. H., & Siano, P. (2021). VMShield: Memory introspection-based malware detection to secure cloud-based services against stealthy attacks. *IEEE Transactions on Industrial Informatics*, *17*(10), 6754-6764. https://doi.org/10.1109/TII.2020.3048791

Mishra, P., Varadharajan, V., Pilli, E. S., & Tupakula, U. (2018). Vmguard: A vmi-based security architecture for intrusion detection in cloud environment. *IEEE Transactions on Cloud Computing*, *8*(3), 957-971. https://doi.org/10.1109/TCC.2018.2829202

Mourad, A., Tout, H., Wahab, O. A., Otrok, H., & Dbouk, T. (2020). Ad hoc vehicular fog enabling cooperative low-latency intrusion detection. *IEEE Internet of Things Journal*, *8*(2), 829-843. https://doi.org/10.1109/JIOT.2020.3008488

Nadeem, M., Arshad, A., Riaz, S., Band, S. S., & Mosavi, A. (2021). Intercept the cloud network from brute force and DDoS attacks via intrusion detection and prevention system. *IEEE Access*, *9*, 152300-152309. https://doi.org/10.1109/ACCESS.2021.3126535

Olanrewaju, R. F., Khan, B. U. I., Najeeb, A. R., Zahir, K. N., & Hussain, S. (2018). Snort-based smart and swift intrusion detection system. *Indian Journal of Science and Technology*, *11*(4), 1-9.

Ramaiah, C. H., Charan, D. A., Akhil, R. S., & Kumar, P. P. (2018). Secure Automated Threat Detection and Prevention (SATDP). *Int J Eng Technol*, *7*, 86-9. https://doi.org/10.14419/ijet.v7i2.20.11760

Ren, J., Guo, J., Qian, W., Yuan, H., Hao, X., & Jingjing, H. (2019). Building an effective intrusion detection system by using hybrid data optimization based on machine learning algorithms. *Security and Communication Networks*, *2019*. https://doi.org/10.1155/2019/7130868

Sadaf, K., & Sultana, J. (2020). Intrusion detection based on autoencoder and isolation forest in fog computing. *IEEE Access*, *8*, 167059-167068. https://doi.org/10.1109/ACCESS.2020.3022855

Shrivastav, S., & Dhawan, G. (2018). Detection of intrusion detection system in cloud using artificial intelligence. *Studies (IJACMS)*, *3*(2). http://www.ijacms.com/submittedFiles/4e880228-faf9-4bd7-bd71-d82d94f40d8c.pdf

Singh, P., Kaur, A., Aujla, G. S., Batth, R. S., & Kanhere, S. (2020). Daas: Dew computing as a service for intelligent intrusion detection in edge-of-things ecosystem. *IEEE Internet of Things Journal*, *8*(16), 12569-12577. https://doi.org/10.1109/JIOT.2020.3029248

Sun, H., & Grishman, R. (2022). Lexicalized dependency paths based supervised learning for relation extraction. *Computer Systems Science and Engineering*, *43*(3). https://doi.org/10.32604/csse.2022.030759

Tan, Z., Nagar, U. T., He, X., Nanda, P., Liu, R. P., Wang, S., & Hu, J. (2014). Enhancing big data security with collaborative intrusion detection. *IEEE Cloud Computing*, *1*(3), 27-33. https://doi.org/10.1109/MCC.2014.53

Urmila, T. S., & Balasubramanian, R. (2019). Dynamic multi-layered intrusion identification and recognition using artificial intelligence framework. *International Journal of Computer Science and Information Security (IJCSIS)*, *17*(2).

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, *7*, 41525-41550. https://doi.org/10.1109/ACCESS.2019.2895334