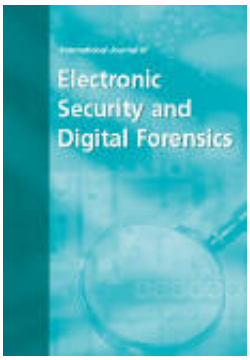


International Journal of Electronic Security and Digital Forensics

ISSN online: 1751-9128 - ISSN print: 1751-911X
https://www.inderscience.com/ijesdf

Security in database management system using machine learning

M. Deepa, J. Dhilipan

DOI: 10.1504/IJESDF.2024.10057609

Article History:

Received:	24 November 2022
Last revised:	17 April 2023
Accepted:	20 April 2023
Published online:	12 January 2024

Security in database management system using machine learning

M. Deepa* and J. Dhilipan

Department of Computer Science and Applications,
Faculty of Science and Humanities,
SRM Institute of Science and Technology,
Ramapuram-600089, Chennai, Tamil Nadu, India
Email: dm8027@srmist.edu.in
Email: hod.mca.rmp@srmist.edu.in
*Corresponding author

Abstract: The term 'database security' refers to the collection of rules, tools, and processes that have been developed to maintain and protect the databases' confidentiality, integrity, and accessibility. The use of machine learning to improve database management security is becoming more common. The fundamental goal of employing machine learning in security is to make the process of malware detection more actionable, scalable, and successful than conventional techniques, which need the participation of humans. This may be accomplished by making the process more automated. The process entails overcoming problems posed by machine learning, which need to be managed in an effective, logical, and theoretical manner. Machine learning algorithm is applied in the critical paths of the tuner. The optimum configuration for the proposed system yields a throughput boost of between 22% and 35% and a latency reduction of around 60%. The method is robust to various attacks.

Keywords: database security'; security techniques; database threats; integrity; machine learning.

Reference to this paper should be made as follows: Deepa, M. and Dhilipan, J. (2024) 'Security in database management system using machine learning', *Int. J. Electronic Security and Digital Forensics*, Vol. 16, No. 1, pp.124–133.

Biographical notes: M. Deepa received her PG degree in Masters of Computer Application from Madras University, India, in 2006. She has completed her MPhil degree and currently pursuing her Doctorate degree from SRM Institute of Science and Technology. Her research areas include intrusion detection attacks, database security, network security, cyber security, deep learning and machine learning. She has participated in various international conference and published papers in various journals.

J. Dhilipan is a Professor and Head at the Department of MCA, SRMIST, Ramapuram. He completed his MSc, MBA, and MPhil and completed his Doctorate in 2014. His area of interest is cloud computing, machine learning, e-commerce and data analysis. He published 35 research articles in reputed journals and conference proceedings.

1 Introduction

The concept of cloud computing refers to the provision of computing resources to users via a network, specifically the internet. Cloud service providers make use of the network for communication purposes, as well as for the exchange of data that is both private and confidential. The cloud is vulnerable to a wide variety of attacks because of its distributed nature. These attacks include denial of service (DoS), exploits, generic, reconnaissance, shellcode, zero-day attack, advanced persistent threat (APT), worms, and many more (Sahi et al., 2017). Worms are another type of attack that can target it. When such an attack is launched, the amount of resources that are being consumed by the server quickly begins to increase. The autoscaling feature allows the cloud manager to begin the process of adding additional resources to this server. If the attack is not identified and its effects are not mitigated, the process of resource allocation will continue with a significant loss. In addition, a single physical server can host the operation of several virtual machines (VMs) simultaneously (Roopa and Selvakumarraja, 2018). When a VM is attacked, it can have repercussions for other VMs running on a single server. These assaults are being carried out for the primary reasons of extortion and protest. Cloud security, coupled with the possibility of inflicting serious harm to the cloud, is a big worry for the most current attack models. This is because of the cloud's reliance on distributed computing. As a consequence of this, a cyber-security risk management (CSRM) strategy is required in order to handle these issues, reduce the chance of risks that are connected with each other, and guarantee preparation for cyber resilience. It refers to the procedure of identifying potential threats to a network's cyber security and developing countermeasures to combat those threats (Roopa and Selvakumarraja, 2017). Recent events have resulted in an increase in the severity of these assaults on the cloud.

The majority of CSRM plans make use of basic deep neural network models and machine learning (ML) models to identify cyberattacks on hosts and network systems in order to defend themselves against these assaults (Challa et al., 2018). An example of an artificial neural network would be a straightforward deep neural network with just one or two hidden layers (HL) (ANN). Nevertheless, a deep network will have a great number of HLs in addition to a variety of diverse designs (Verma and Dey, 2015). The vast majority of researchers make extensive use of a technique called deep learning (DL), which is named for its capacity to simulate the natural processes that occur in the human brain via in-depth studies of computational processes (L-Ghamdi, 2021). In addition, the use of this technology has been shown to have exceptionally high rates of false-positive results, in addition to having problems recognising the most recent assault types (Cui, 2021).

Conventional ML approaches have shown a number of drawbacks, including lengthy training time requirements, poor detection accuracy, and an alarmingly high proportion of false positives (Kanimozhi and Prem Jacob, 2019). As a consequence of this, the majority of artificial intelligence (AI) systems have shown typical adversary agnostic behaviour, which means that they do not acknowledge the potential of a hostile assault. As a result, there is a significant possibility of a hostile backdoor poisoning assault being carried out. Inaccurate identification of attacks and data breaches were also caused by a lack of tagged samples (Annarelli et al., 2020). In light of the aforementioned predicament, the research presented here suggests using HT-RLSTM for the purpose of providing intelligent cyber security defences and protections.

The organisation of this paper is as follows: Section 2 provides a survey of the associated works regarding the proposed methodology. The methodology that has been

proposed is explained in Section 3. Section 4 describes the illustration of the results and discussion of the proposed methodology, which is based on performance metrics. The last part of the paper is Section 5, which discusses potential future research.

2 Related works

Tian et al. (2019) built a web attack detection system (ADS) that made advantage of the potential benefits of URL analysis. The technology, which was developed to operate on edge devices, was created with the intention of identifying harmful behaviour on the internet. In the edge of things (EoT) model, the cloud was able to successfully handle a wide range of problems. It was decided to make use of a large number of concurrent deep models in order to both improve the system's overall stability and simplify the process of keeping the system up to date. The experiment was carried out on a platform that had two parallel deep models, and a large number of datasets were utilised to evaluate the performance of the platform in comparison to other platforms. The results of the testing demonstrated that the system was effective in identifying malicious web requests, with an accuracy of 99.41%, a true positive rate (TPR) of 98.91%, and a detection rate of normal requests of 99.55% (DRN). The strategy, on the other hand, proved ineffective when it came to defending computers against the most recent cyber security threats.

Agarwal et al. (2021) created a P-estimation detection system that was capable of efficiently detecting assaults. This was carried out using a number of trained DL-LSTM models and was centred on the logs from the web server. After deriving an estimate of the attack percentage, the data from that model was utilised to implement an appropriate detection strategy. This strategy takes into consideration the dynamic nature of websites, which means that the degree of popularity enjoyed by specific web pages may alter over the course of time as a consequence of the constant updating of detection models. The FNR and FPR that could be achieved with this technology were 0.0059% and 0%, respectively, and it had the capacity of detecting attacks with an intensity as low as 2%. In addition, this article contained strategies for mitigating their effects and assigning blame in order to recognise and prevent such assaults. Nevertheless, the method required a significant number of training samples. In this way, large computational difficulties were also found for the identification of APT attacks.

The method of auto-encoder-based DL was used in the study that was conducted by Abdullayeva et al. (2021). A high classification rate was achieved with this method thanks to the identification of intricate relationships between the features. In addition, the approach made the task of classifying enormous volumes of data more simpler by cutting down on the amount of data stored in the encoder. An unsupervised analysis of the useful characteristics that were derived from network traffic data was carried out after the deployment of an auto-encoder neural network at the beginning of the process. Following that, the SoftMax regression layer was added to the top layer of the newly formed auto-encoder network for the categorisation of APT attacks. The statistics indicated that the method was correct 98.32% of the time. On the other hand, it was difficult to stop opponents from stealing or physically altering the evidence.

Using a strategy known as previously-selected-server-first (PSSF), AdiMaheswara A method that is both dynamic and scalable for the placement of VM was developed. In this technique, the number of movements that could be computed was restricted. The acquired set of movements served as the basis for the construction of the greedy baseline design.

When a new VM placement request was being processed, the highest priority was given to previously hosted servers or VMs that were already being hosted by a user who had a profile in PSSF that was comparable to the new request. This ensured that the new VM could be placed as quickly as possible. The performance of this method was analysed using 20 distinct VMs that were hosted on four distinct servers. In terms of both hit rate and loss rate, as well as resource loss, the newly developed method demonstrated significantly higher levels of efficiency when compared to previously utilised strategies for the placement of VM. The accuracy of this method, on the other hand, did not meet my expectations.

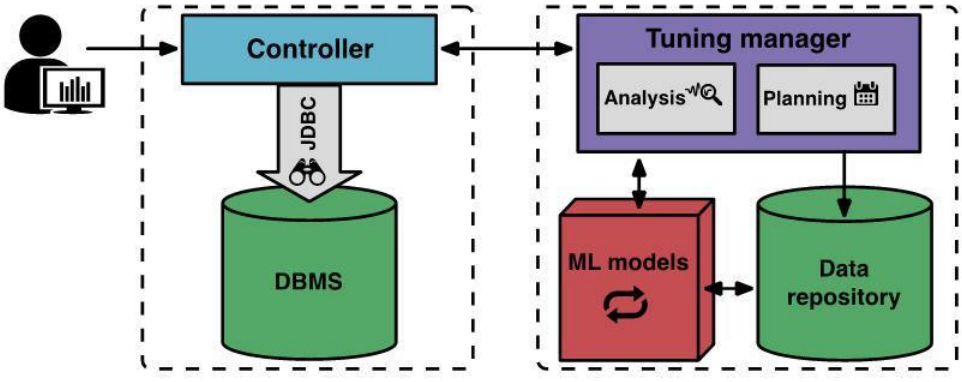
Kushwah and Ranga (2020) recently presented the most up-to-date method for detecting distributed DoS attacks in a cloud computing environment. A voting extreme learning machine was the component that made up this method. Both the NSL-KDD and the ISCX intrusion detection datasets were utilised so that the experiments could be evaluated. According to the results of the tests, the detection rate of the system for the assaults was 92.11% with the ISCX dataset and 99.18% with the NSL-KDD dataset respectively. The developed system's performance, which was centred on back-propagation ANN, was compared to the performance of other systems. In order to train the ANN, we made use of black hole optimisation, random forest, extreme learning machine, and Adaboost. The method, on the other hand, had some shortcomings due to an unacceptable level of network overhead, most notably with regard to latency.

The researchers Yesin et al. (2021a) present a method for assessing the safety of relational databases that is based on an improved version of the Clements-Hoffman theoretical model. When calculating the level of security, an integral quantitative metric is used. When security precautions are in place, this metric is the inverse of the overall residual risk that is connected to the potential for threats to be implemented in connection to a database item. This study focuses on the main strategies that have been used to ensure the integrity of data and permanently stored database modules in response to the recommendations of the Clark-Wilson model. The authors provide a method to guarantee the reliability of both the programmes and the data contained in databases. This mechanism is underpinned by the principles of relational database theory, the row level Security technology, the potential of the contemporary blockchain model, and the capabilities of the database management system (DBMS) that is implemented on the platform that databases with the universal basis of relations are used on. Because of the application of this mechanism, it is ensured that the saved data and programmes will not be corrupted, altered, or distorted while being kept intact.

3 Research methodology

Database management systems, often known as DBMSs, are the most essential part of any programme that handles large amounts of data. They are able to deal with enormous volumes of data and demanding workloads. The issue is that there are hundreds of configuration 'knobs' that control things like how often data should be sent to storage and how much RAM should be used for caches. This makes them tough to maintain. Tuning tasks are often delegated to experts by organisations, but the excessively high costs of specialists prevent many from doing so.

Figure 1 The ML pipeline (see online version for colours)

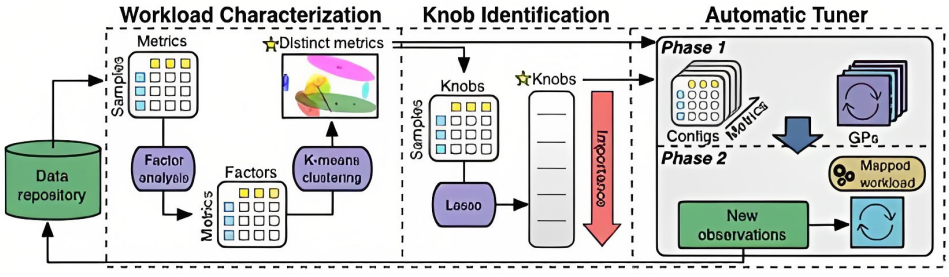


The technology that is being presented is able to automatically determine suitable settings for the configuration knobs of a DBMS. The purpose of this paper is to make it simpler for anybody to implement a DBMS, even individuals who have no prior experience managing databases. It is unique among DBMS configuration tools in that it applies the expertise obtained from adjusting earlier DBMS deployments to the process of tuning new DBMS installations. As a result, the time and resources needed to fine-tune a new DBMS deployment are significantly reduced. In order to do this, the system under consideration keeps a store of tuning data that was compiled from earlier sessions of tuning. The behaviour of the DBMS when configured in various ways is then captured using ML models built using this data. The suggested system makes use of these models as a basis for directing experimentation with new applications, making recommendations for settings that enhance performance toward a particular goal.

We go over all the components that make up the ML pipeline of the proposed system and demonstrate how those components interact with one another to fine-tune the configuration of a DBMS. The results of the proposed system's optimum configuration are then compared to those of configurations selected by database administrators (DBAs) and other automated tuning tools to see how effectively the system can tune MySQL and Postgres.

The next graphic provides an illustration of how data is handled as it travels through the ML pipeline of the proposed system. The repository that has been suggested for this system contains all of the observations. The suggested system starts the process of characterisation of the workload by sending observations to the workload element. This component selects a more limited range of DBMS metrics that are the most effective at capturing the variability in performance as well as the differentiating features for various workloads. The next step is for the knob identification element to produce a ranked group of the knobs that have the greatest impact on the performance of the DBMS. After that, each and every piece of data is delivered to the automatic tuner by the proposed system. This component uses the most comparable workload data from its data repository to generate more optimal system configurations by mapping the workload of the target DBMS to that workload.

Figure 2 ML pipeline (see online version for colours)



Workload characterisation

The proposed technique uses the internal runtime metrics of the DBMS to characterise the behaviour of a workload. These metrics provide a true picture of a task since they adequately capture the behaviour shown by a work while it is being carried out. Many of the metrics, however, are redundant; some of them are the same measurement recorded in different units, while others represent separate DBMS components with unique but closely related values. Some of these metrics are documented in the table below. It is essential to get rid of duplicate measurements since doing so will make ML models that make use of such measures simpler. We organise the DBMS's metrics into clusters according to the correlation patterns between them. After that, we choose one metric to be indicative of each cluster, especially the one that is located in the most central position inside the cluster. These measurements are used by subsequent parts of the ML process.

Knob identification

there may be hundreds of knobs on a DBMS, but only a subset of those knobs affect the performance of the DBMS. The proposed system employs a well-known feature-selection technique called Lasso to identify which knobs significantly affect the system's performance. By employing this technique to the information contained in its repository, the proposed system establishes the hierarchy of importance of the DBMS's knobs.

The proposed system then has to determine how many of the knobs to use in order to make configuration recommendations. The amount of time required to optimise the suggested system is significantly increased when an excessive number of them are used. Using an insufficient number of them may make it impossible for the proposed system to locate the optimal configuration. The proposed system takes an incremental approach to automating this process in order to achieve this goal. The total number of knobs needed in the tuning procedure eventually increases. This approach allows the suggested system to study and fine-tune the configuration for a small number of the system's most crucial knobs before extending its scope to include other factors.

Automatic tuner

Following the end of each observation period, the automated tuning component does a two-step analysis to decide which configuration the suggested system should suggest. The system starts out by looking at the performance information for the metrics that were

chosen to make up the workload characterisation component. In doing so, it is able to identify which workload from a prior tuning session most closely resembles the workload of the target DBMS. To identify whether workloads respond similarly to different knob settings, the metrics from the present session are compared to the data from earlier workloads.

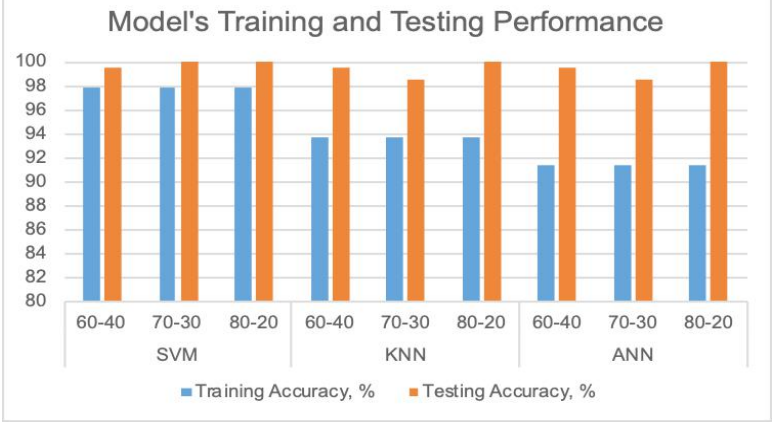
After that, the algorithm selects another knob setup to test out. In addition to the data from the repository entry that corresponds to the workload that is the most similar, it then applies a statistical model to the data it has already collected. Because of this model, the proposed system is able to make accurate predictions on how effectively the DBMS would function with each of the available configurations. The suggested method optimises the following configuration by making a trade-off between exploration and exploitation.

4 Results and discussion

Because runtime performance is not a primary concern for the workload characterisation and knob recognition elements, we used scikit-learn to implement the ML algorithms that correspond to those components. These algorithms are executed in the background as processes, and they incorporate new data in the repository of the proposed system whenever it is made available.

The ML algorithms are located on the crucial route for the Automatic Tuner. They are run after each observation time, including fresh data in order for the proposed system to choose a knob configuration to test out next, and they do this after each observation period. TensorFlow was used in the implementation of these algorithms since we were concerned about how well they performed. We combined the controller of the proposed system with the OLTP-Bench benchmarking framework in order to gather information on the hardware, knob settings, and runtime performance metrics of the DBMS. Figure 3 displays the performance metrics of various types of ML algorithms

Figure 3 Accuracy comparison of various ML algorithms (see online version for colours)



5 Experiment design

We used the ideal configuration selected by the suggested method to evaluate the effectiveness of MySQL and Postgres with the following:

- Default: the settings that the DBMS makes accessible.
- Tuning script: the configuration that an open-source tuning advisor tool had generated.
- DBA: The configuration that was selected manually by a DBA.
- RDS: The deployment of the configuration, which is tailored for the DBMS and uses the same EC2 instance type. Amazon RD is in charge of this setup.

The Amazon EC2 Spot Instances served as the testing ground for all of our procedures. Each experiment was carried out on two separate instances, one of which served as the controller for the proposed system, and the other as the target DBMS deployment. Big and m3.xlarge are the appropriate instance types for this instance size. We used the M4 rifle. The tuning manager and data repository for the suggested system were installed on a local server with 20 processor cores and 128 gigabytes of RAM. We used the TPC-C workload, which is the industry standard for evaluating workloads for such systems, to assess the performance of online transaction processing (OLTP) systems.

6 Evaluation

In this experiment, we used MySQL and Postgres as our databases. We measured the latency and throughput of both of these databases. The findings are presented in the graphs that follow. The 'worst case' time it takes for a transaction to be completed is shown by the first graph, which shows the latency that corresponds to the 99th percentile. The results for throughput are shown in Figure 4 and Figure 5, which were calculated by adding up the usual number of transactions completed per second, are shown in the second graph.

Figure 4 Latency time and throughput time (see online version for colours)

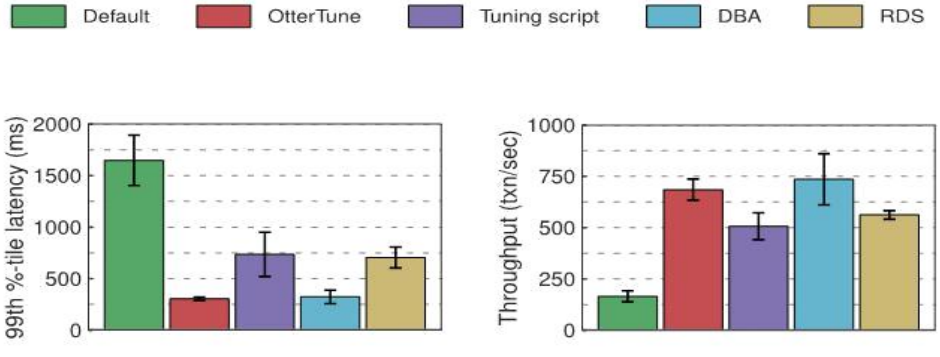
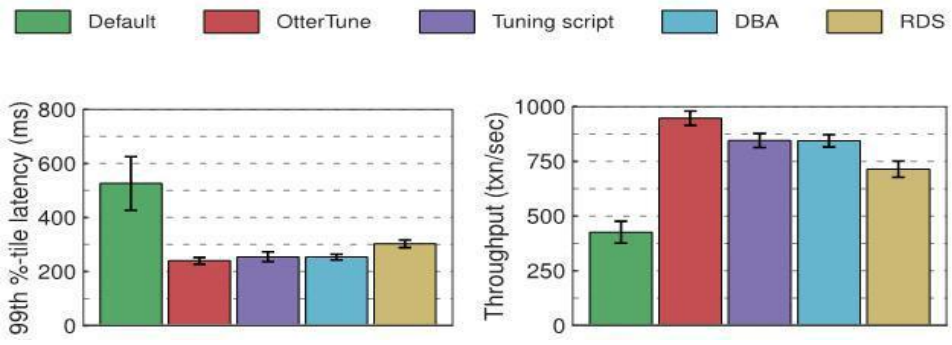


Figure 5 Latency time and throughput time (see online version for colours)



7 MySQL results

The solution that is being suggested may also create a configuration that is almost as excellent as the one chosen by the DBA. MySQL's overall performance for the TPC-C workload can be significantly altered by turning only a few of its knobs. Each of these knobs can benefit from the optimal settings that are provided by the configurations that are produced by the proposed system and the DBA. Because it offers a setting that is less than ideal for one of the knobs, RDS performs marginally less well. Because the tuning script simply changes one knob, its performance is at its lowest point.

8 Postgres results

The configurations produced by the proposed system, the tuning tool, the DBA, and RDS all provide improvements in latency that are equivalent to those made by Postgres' default settings. This is most likely due to the cost involved by the repeated network round trips required to communicate between the OLTP-Bench client and the DBMS. With the configuration recommended by the proposed system, Postgres performs around 12% better in terms of throughput than it does with the settings picked by the DBA and the tuning script, and about 32% better than RDS. Similar to MySQL, Postgres' performance may be significantly changed by adjusting a few knobs. The configurations produced by the proposed system, the DBA, the tuning script, and the RDS were used to change these knobs; for the most part, these configurations offered values that were suitable.

9 Conclusions

Finding appropriate values for the configuration knobs of a DBMS can now be done quickly and easily with the help of the approach that has been suggested. It recycles training data obtained from earlier tuning sessions so that it may tune fresh installations of the DBMS. Tuning time is considerably cut down on account of the fact that the suggested system does not need the generation of an initial dataset in order to train its ML

models. In order to support the growing number of DBaaS installations, the proposed system will soon be able to automatically identify the hardware capabilities of the target DBMS without requiring remote access. These deployments don't allow for remote access to the machine that hosts the DBMS. Existing works are notoriously difficult to complete in a timely manner. Therefore, the method that has been provided is still strong for an ADS and for attacks that only have a short amount of time.

References

Abdullayeva, F.J. (2021) 'Advanced persistent threat attack detection method in cloud computing based on autoencoder and softmax regression algorithm', *Array*, 1 July, Vol. 10, pp.1–11.

Agarwal, A., Prasad, A., Rustogi, R. and Mishra, S. (2021) 'Detection and mitigation of fraudulent resource consumption attacks in cloud using deep learning approach', *J. Inf. Secur. Appl.*, 1 February, Vol. 56, pp.1–14.

Annarelli, A., Nonino, F. and Palombi, G. (2020) 'Understanding the management of cyber resilient systems', *Comput. Ind. Eng.*, 1 November, Vol. 149, pp.1–18.

Challa, S., Das, A.K., Gope, P., Kumar, N., Wu, F. AND Vasilakos, A.V. (2018) Design and analysis of authenticated key agreement scheme in cloud-assisted cyber-physical systems', *Future Gener. Comput. Syst.*, 1 July, Vol. 108, pp.1–25.

Cui, H. (2021) 'Handoff control strategy of cyber physical systems under dynamic data attack', *Comput. Commun.*, 1 October, Vol. 178, pp.183–190.

Kanimozhi, V. and Prem Jacob, T. (2019) 'Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing', *ICT Express*, 4 April, Vol. 5, pp.211–214.

Kushwah, G.S. and Ranga, V. (2020) 'Voting extreme learning machine based distributed denial of service attack detection in cloud computing', *J. Inf. Secur. Appl.*, 1 August, Vol. 53, pp.1–12.

L-Ghamdi, M.I.A. (2021) 'Effects of knowledge of cyber security on prevention of attacks', *Mater. Today Proc.*, in press.

Roopa, M. and Selvakumarraja, R.S. (2017) 'An intelligent network algorithm for enhanced security in a mobile ad hoc network', *International Journal of Networking and Virtual Organisations*, Vol. 17, Nos. 2/3, pp.1470–9503.

Roopa, M. and Selvakumarraja, R.S. (2018) 'Intelligent intrusion detection and prevention system using smart multi-instance multi- learning protocol for tactical mobile ad hoc networks', *KSII Transactions on Internet and Information Systems*, June, Vol. 12, No. 6, pp.2895–2921, ISSN: 1976-7277.

Sahi, A., Lai, D., Li, Y. and Diykh, M. (2017) 'An efficient DDoS TCP flood attack detection and prevention system in a cloud environment', *IEEE Access*, 6 April, Vol. 5, pp.6036–6048.

Tian, Z., Luo, C., Qiu, J., Du, X. and Guizani, M. (2019) 'A distributed deep learning system for web attack detection on edge devices', *IEEE Trans. Ind. Inform.*, Vol. 16, No. 3, pp.1963–1971.

Verma, C. and Dey, S. (2015) 'Methods to obtain training videos for fully automated application-specific classification', *IEEE Access*, 27 July, Vol. 3, pp.1188–1205.

Yesin, V., Karpinski, M., Yesina, M., Vilihura, V. and Rajba, S. (2021a) 'Technique for evaluating the security of relational databases based on the enhanced Clements-Hoffman model', *Appl. Sci.*, Vol. 11, No. 23, p.11175.