User Behaviour Risk Evaluation in Zero Trust Architecture Environment

Wawan Yunanto
Dept. of Computer Science and Information Engineering
National Taiwan University of Science and Technology
Taipei, R.O.C.
d10615811@mail.ntust.edu.tw

Hsing-Kuo Pao
Dept. of Computer Science and Information Engineering
National Taiwan University of Science and Technology
Taipei, R.O.C.
pao@mail.ntust.edu.tw

2022 IEEE 8th World Forum on Internet of Things (WF-IoT) | 978-1-6654-9153-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/WF-IOT54382.2022.10152197

***Abstract*—Our study proposes user access evaluation framework within the trustworthy organization network. We design a mechanism on how to measure user access risk on every interaction with the servers. It is an extensive anomaly detection problem where the already trusted user is continuously being observed while interacting with organization resources. Moreover, alongside with network perimeter based authentication, this study provides an improved access control to evaluate the risk of every user behavior in day-to-day operation. To portray user interaction with organization server, our work focus on user behavior risk evaluation using log entries recorded by Apache Web Server. We perform the log analysis process and develop an unsupervised recurrent deep neural net architecture which is widely used in recent log data anomaly detection for high performance computing and cloud computing infrastructure. This approach may perform well in previous use cases but its never been arranged for Apache Access log data. Unlike other log analysis datasets, ours has no label defined by domain expert, so we propose an alternative way to carry out and evaluate the experiments for User anomaly behaviour detection. Our method yields a good result on small log data and has a decent performance on large data. This is a very promising outcome in the early stage of User Behaviour Risk Evaluation.**

***Index Terms*—zero trust architecture, user behaviour, log analysis, anomaly detection**

I. Introduction

Zero Trust Architecture (ZTA) plays a significant role in today's trusted online working environment. The main concept behind the zero trust security model is "never trust, always verify," which means that devices/users should not be trusted by default, even if they are connected to a permissioned network such as a corporate LAN and even if they were previously verified.

Although the term "zero trust" has been used by IT industry since early 2000s, its principles and definitions have only been published in 2020 by US. Department of Commerce [1]. In the document, ZTA is defined as an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources. A year later, US government published a Presidential Document to improve cybersecurity across federal governments to keep pace with dynamic and sophisticated cyber threats [2]. They urge all federal governments to modernize cybersecurity by adopting security best practice, which is Zero Trust Architecture.

Let's have a look in todays work environment, there has been several new ways how workers interact with company servers. Such as Bring Your Own Device, Cloud Computing, and Remote Apps (mostly when we were Working From Home during the covid-19 pandemic). Network-based perimeters, such as password authentication, MAC or IP address authorization, would not be enough. These protection technologies that are based on static rules and policies might prevent us from threats originating from outside the organization, but unable to prevent computer security threats from malicious insiders. So, one of important thing in ZTA is how to control the behavior of users, assets, and resources within the organization.

One of the insider party in an organization that we need to monitor is user activities. Conventional client-server interaction use network-based perimeters to authorize user access. Once a user is authorized by the server equipped with Policy Decision Point, he/she will always be trusted to access the organization resources every time. However, several studies have shown that humans are considered the most vulnerable to security threats. It is estimated that 95% of cybersecurity breaches and network attacks are due to human errors [3]. Some human unsafe behaviours that compromise cyber and network security include, but not limited to, accessing untrusted sources, oversharing information on social media, falling victim to phishing, and not updating softwares [4].

In ZTA, the rule to authorize user access must be dynamic, we never trust every access made by the same user, even when this user has been verified before. Our defense system will focus on each access by monitoring the user behavior before he/she being authorized.

This study proposed a new approach on how to justify user behaviour risk after being authorized by the network perimeters. In our industry cooperation, our team build an application to portray the ZTA environment and the user behavior information was drawn from the server access log. The Policy Decision Point will dynamically analyze the risk of each access by evaluating the behavior of the connected user. The output is a score entitled Behaviour Trust Score between 0 and 1 to justify how risky a connection could be. Since this study still in its early stage, we focus on the performance of our initial model that underlies our scoring method.

The rest of this paper is organized as follows: Section 2 gives a brief introduction of our current development in the field log analysis and anomaly detection. Section 3 describes the proposed framework and the problem that we try to accomplished. Next, Section 4 explain the dataset and the experiment result. Section 5 summarizes the discussion from the previous sections and advises future works.

II. Related Work

A. Unstructured System Log

Over the years, logs have been the crucial component in system development, maintenance, and security. Since IT infrastructures are widely implemented in almost every business operation, the growth of system runtime environment recorded in logs are enormous. To extract useful information from logs efficiently and effectively, several research focuses on AI (artificial intelligence) powered log analytic [5] [6]. However, there is a significant gap between research academic and industry implementation due to the lack of public dataset for a spesific system including its annotation from domain experts. There are several organized system log repositories to facilitate and benchmark research in this field, such as LogHub. LogHub provides 17 real-world log datasets collected from a wide range of systems, including distributed systems, supercomputers, operating systems, mobile systems, server applications, and standalone software [7]. Unfortunately, Apache Access log is beyond the scope of LogHub, so we use datasets originated from other sources as we explain later on.

B. System Log Data Analysis

Our work is highly related to System Log Data Analysis. This research area cover the complete steps from how to preprocess raw log data, to how to analyze this data to fulfill industrial needs. Log entries were recorded as a text so that the raw log data file become unstructured. In order to utilize it further, this raw file needs to be parsed and filtered. Log parsing is a preprocessing strategy on how to convert unstructured log entry into structured log before being analyzed. For a specific log format on a specific system, parsing can be done by setting up rules or using regular expression, but rule will be useless if we use it for other system logs. Regular expression is suitable for most Apache Access log since the unstructured form already follow specific rules determined by domain experts and sometime we even do not need to filter out. However, researchers prefer a more general approach, a strategy that can be used for broader collection of logs format. Drain [8] and Spell [9] are the examples of well-known log parsing strategies which utilize dataset from LogHub repository.

C. System Log Anomaly Detection

In log data, there is information about production server activities in day-to-day operation. We consider this as server behavior. And from this behavior we can monitor our operational system to maintain its performance during its working time. Any unusual behavior can be detected from its log data

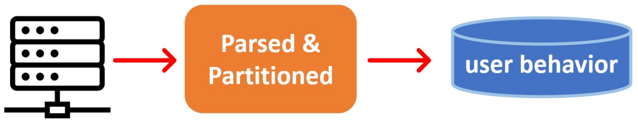


Fig. 1. The User Behaviour Risk Evaluation Pipeline.

as an early warning before the system completely fails, and to prevent company loss from a malfunctioned system. Most of industrial needs in log data analysis is anomaly detection, including our own User Behavior Risk Evaluation.

Malicious behaviours can be detected by some constraints written in the log entry since this activity often leave a technical footprint in the server. So it is not that difficult to distinguish a normal and a bad user access conventionally. Nevertheless, an anomaly behaviour is not always leave a malicious footprint. If a user credentials has been compromised by attacker and use them to interact with the company server, it will not consider as a threat, because the access already pass the network perimeter authentication. This is where ZTA comes into play, the server will never trust a single access, it will always verify. Consequently, an anomaly is not only determined by a malicious activity but also determined by the user activities specified by a series of events made by that user earlier. DeepLog [10] and LogAnomaly [11] are some highly cited publications on system anomaly detection based on this type of criteria, focus more on user behaviour rather than a single log entry.

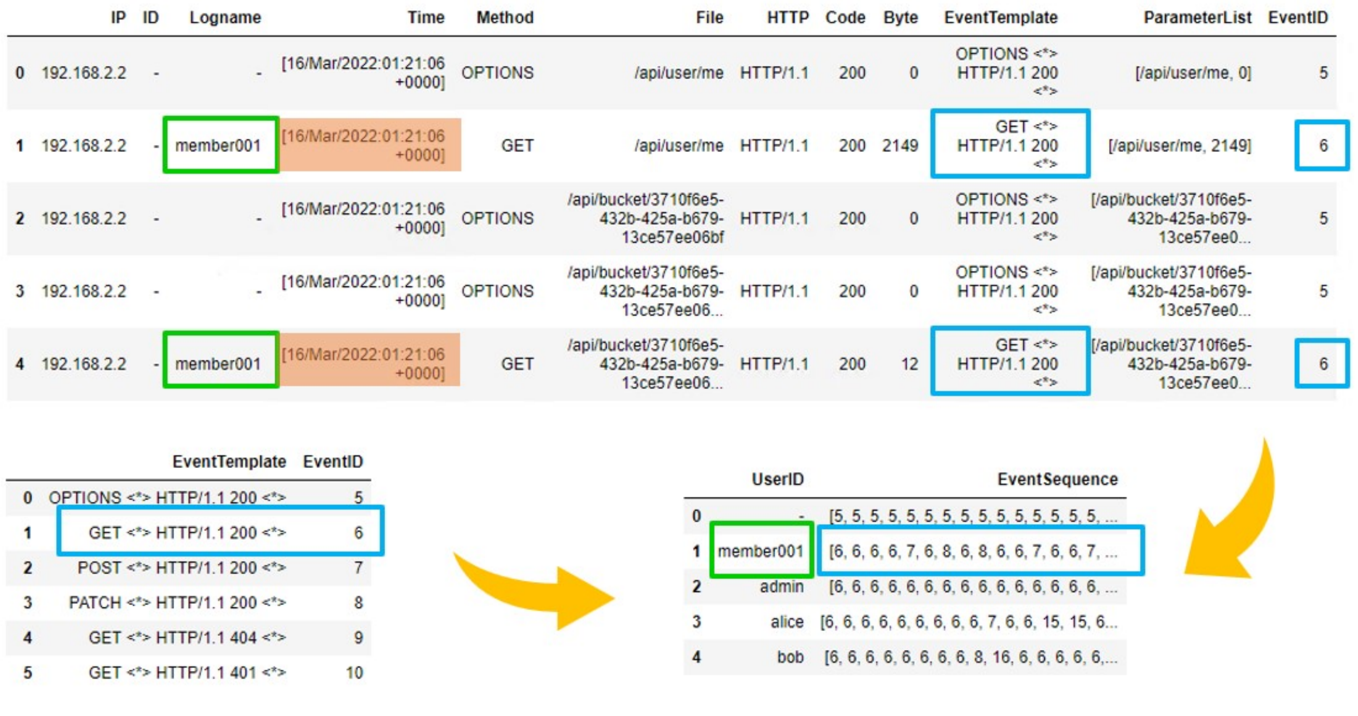
III. The Proposed Framework

A. The Focused Problem

User Behaviour Risk Evaluation is a method on how to never trust a user even though he/she already pass the network perimeter and to verify the session status by producing a score the specify the risk level. The end product of this proposed method is not to deny access regardless the score but more about giving the system a second opinion so it can decide which actions should be made. For instance, the server can send a second authentication (two-way authentication) or a One Time Password (OTP) if the score is low and do nothing if the score is high.

The big picture of our framework is depicted in Fig. 1. Ideally, we need to use our own log data form the web-based application system that we design to build the anomaly detection model for our risk evaluation, but since it is still under developed and the log data is not sufficient enough, we utilize public Apache Access log dataset instead. We realize that the score it produce will not reflect the access risk of our users but we will get model blueprint that ready to be re-train whenever our system is finish. So in this study, we focus on the pipeline building and evaluation of the initial model behind the Behaviour Trust Scoring feature.

As mentioned earlier, Apache Access log format comply with one the official format specified by domain experts. So,



	IP	ID	Logname	Time	Method	File	HTTP	Code	Byte	EventTemplate	ParameterList	EventID
0	192.168.2.2	-	-	[16/Mar/2022:01:21:06 +0000]	OPTIONS	/api/user/me	HTTP/1.1	200	0	OPTIONS <*> HTTP/1.1 200 <*>	[/api/user/me, 0]	5
1	192.168.2.2	-	member001	[16/Mar/2022:01:21:06 +0000]	GET	/api/user/me	HTTP/1.1	200	2149	GET <*> HTTP/1.1 200 <*>	[/api/user/me, 2149]	6
2	192.168.2.2	-	-	[16/Mar/2022:01:21:06 +0000]	OPTIONS	/api/bucket/3710f6e5-432b-425a-b679-13ce57ee06bf	HTTP/1.1	200	0	OPTIONS <*> HTTP/1.1 200 <*>	[/api/bucket/3710f6e5-432b-425a-b679-13ce57ee0...	5
3	192.168.2.2	-	-	[16/Mar/2022:01:21:06 +0000]	OPTIONS	/api/bucket/3710f6e5-432b-425a-b679-13ce57ee06...	HTTP/1.1	200	0	OPTIONS <*> HTTP/1.1 200 <*>	[/api/bucket/3710f6e5-432b-425a-b679-13ce57ee0...	5
4	192.168.2.2	-	member001	[16/Mar/2022:01:21:06 +0000]	GET	/api/bucket/3710f6e5-432b-425a-b679-13ce57ee06...	HTTP/1.1	200	12	GET <*> HTTP/1.1 200 <*>	[/api/bucket/3710f6e5-432b-425a-b679-13ce57ee0...	6

	EventTemplate	EventID
0	OPTIONS <*> HTTP/1.1 200 <*>	5
1	GET <*> HTTP/1.1 200 <*>	6
2	POST <*> HTTP/1.1 200 <*>	7
3	PATCH <*> HTTP/1.1 200 <*>	8
4	GET <*> HTTP/1.1 404 <*>	9
5	GET <*> HTTP/1.1 401 <*>	10

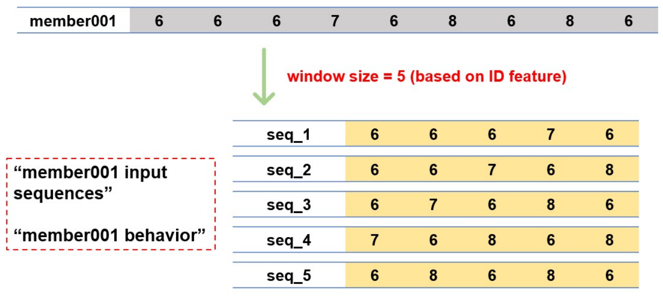
	UserID	EventSequence
0	-	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
1	member001	[6, 6, 6, 6, 7, 6, 8, 6, 8, 6, 6, 7, 6, 6, 7, ...
2	admin	[6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, ...
3	alice	[6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 6, 6, 15, 15, 6...
4	bob	[6, 6, 6, 6, 6, 6, 6, 6, 8, 16, 6, 6, 6, 6, 6,...

Fig. 2. Parsing Stage.

given a new log message, we define the structured version with the help of regular expression. As we can see from Fig. 2, the system parameters are depicted by the green rectangle, the timestamps marked by orange background, and the Event Contents are depicted by the plain background. System parameters will be collected as they are, because they already in structured form while the event content will be process further. Since we want to portray user behaviour as a sequence of user events, we engineer EventTemplates and EventIds features from the structured log data and group every EventID of the same user as his/her behaviour. Now, every log entry is equipped with its own EventID and EventTemplate. The next important step is to partition our log data into sequences based on the need to be resolved. We want to score user behavior risk within the organization, so we need to partition based on user ID. We collect EventIDs from particular user ID and produce EventSequences for that user. Eventually, every user will have its own EventSequence and it will grow as long as the user establish a connection to the server. To some extent, we might also need to involved timestamp to get more appropriate EventSequences.

For example, we can see from the bottom right in Fig. 2 that member001 produces log entry that matches the Event-Template identified as 6. The log partition function will record this and put it in member001's EventSequence.

We developed our sequence input by utilizing identity feature in our log entry. Meanwhile, our model were trained based on user historical behavior (within the same user role) portrayed by the EventSequences. Afterwards, we will use them to train our anomaly detection deep learning model. But first, we need to define what "user behavior" is.



member001	6	6	6	7	6	8	6	8	6

window size = 5 (based on ID feature)
"member001 input sequences"
"member001 behavior"

seq_1	6	6	6	7	6
seq_2	6	6	7	6	8
seq_3	6	7	6	8	6
seq_4	7	6	8	6	8
seq_5	6	8	6	8	6

Fig. 3. Partitioning Stage.

Every user activity on the web server exhibit a long EventSequence as a record of their interaction with the server. These activities considered as behavior that our model must captured. In order to do this, we build a list of sequence consisting of partitioned EventSequences with specified window size. As depicted in Fig. 3, suppose we have member001 and its EventSequence, also given window size equal to 5. We create a list of sequences considered as member001's behaviour. Our anomaly detection model will learn this behaviour and capture the normal behaviour pattern of member001. This will enrich the information that we can retrieved to build the User Behavior Risk Evaluation model.

B. User Behavior Risk Evaluation

In our experiment, we use Apache access log, that is log data generated by the Apache web server based on user access, since we need to evaluate user behavior. This log is not in the Loghub repository, so we need to find another repository for this type of dataset.

In this work, the requirement is not a label whether a user access is considered anomaly or not, but a scoring function to describe how risky a user access could be. After the parsing phase, we extracted several distinctive events from our dataset. For system behavior anomaly detection, every log message will form chronological sequence that can be partitioned into sequences with a certain length. Therefore, we will have sequences of user events considered as user behavior which we will then use for model training process. The sequence length is a parameter that we need to try with several different value to get the best result.

The model structure, depicted in Fig. 4, composed of stacked LSTMs in several hidden layers with each layer consist of several LSTM cells. At the end, we have a fully connected layer which has the number of nodes as many as the total EventIDs collected at Log Parsing stage. To capture the user behavior pattern, this list of sequence is fed to the model and each sequence will be used to predict the next EventID. Let's say, seq_1 has 5 first EventIDs as input, as this sequence goes through the network, it will try to predict the next EventID, which is actually the last EventID of the next sequence, seq_2. And then an internal Neural Network mechanism called back propagation will update the weights in the hidden layers. It goes on and on until the very last sequence and this is how the model learns how to capture the user behaviour pattern.

As we stated earlier, we employ public Apache Access dataset from the that is not build specifically to our User Behaviour Risk Evaluation. To use this dataset in our work, we will make several adjustments and assumptions. First, We will consider that 1 user represent 1 role, so we will train our model using this dataset to get user behaviour within that role. And second, due to the lack of labels, we assume that all log entries in the dataset are normal. We try a different performance measurement, other than measurement that used by HDFS or OpenStack dataset from LogHub repository which have labels.

After we trained the anomaly detection model, the next task is to defined how to obtain the Behaviour Trust (BT) score in the inference process. Given several new EventIDs from newly user access to the server, we create sequences just like in the previous stage, only this time we fed them to the model that has already trained. The output consists of probability distribution of n EventIDs. We take $top-k$ out of n to be our candidates. If the actual label (the last EventID of the next sequence) falls outside the $top-k$, we consider the sequence (that represent user behaviour) as Anomaly. And that is how we will measure our model performance.

$$BT = 1 - \frac{|S_A|}{|S|} \tag{1}$$

Given S is a set of sequences involved in the inference process and S_A is a set of anomaly sequences in S. We define Behaviour Trust score as in (1). The BT score will fall between 0 and 1.

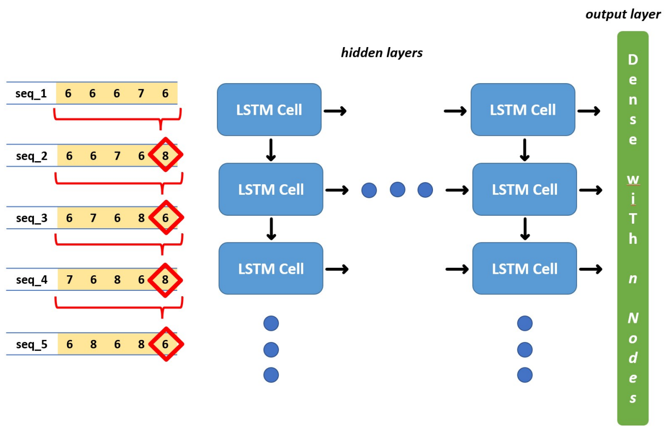


Fig. 4. The Initial Model Architecture.

IV. Experiment Results

A. Dataset

Our work utilizes 2 public datasets to represent small and large system. The small log data was drawn from a tourism website called http://www.almhuette-raith.at on January 19^{th}, 2022. Unfortunately, it has limitation related to the unbalanced number of user access to its web server and no information about the access role in the system. This dataset was collected for about 2 years from Dec 19, 2020, to Jan 19, 2022, with more than 2.2 million log entries.

For the larger system, we employ CTF1.log dataset, a public Apache Access log entries provided by horangi.com, a cybersecurity company based in Singapore. They offer an integrated SaaS security platform with world-class cybersecurity services. CTF1 consists of 3,270,764 log entries, collected between August 28^{th} and October 12^{th}, 2015. To the best of our knowledge, CTF1 provides the largest log data with sufficient number of IDs. There are 4947 identities found in this dataset, which represent a broad variety of users accessing the Apache web server.

Our development starts by preprocess the dataset, which are parsing and partitioning, before we feed them into our model. Recall that this is a public datasets, it is not specially provided for User Behavior Risk Evaluation in a secure network environment where all users should have passed the network perimeters authentication. So, these datasets will not perfectly fit our requirement, we need to make some adjustments and assumptions as we described earlier, especially for the scoring part.

These are some limitation that we encountered. First, our User Behavior Risk Evaluation model requires sufficient log entries containing an adequate number of users. Moreover, we build the model for every roles in the organization. The information about which user belongs to which role is very critical. Although, role information is not directly provided in the log entries, it can be obtained via system's back-end information. So, to know the underlying knowledge about the system that generate the log is very beneficial to our model. Second, there are no label for model evaluation in production. In our Behavior Trust scoring that we explained earlier, we produce the score by predicting the next event based on series of other events that precede it, and not rely on the actual label specified by domain expert. Even so, there are some system log datasets that provide labels specified by their domain expert. These labels distinguish the normal log entry and the anomaly. Although not all well-known log anomaly detection algorithms take a supervised approach, these labels are still very useful to evaluate a model in production. Therefore, there are several unsupervised log anomaly detection methods measure their performance using the labels [10] [11] [12].

B. Evaluation

Since this work still in its early stage, our main focus is how to build the overall pipeline starts from log analysis phase to the Behaviour Trust scoring phase. We need to ensure that this procedure will work successfully in production. As of now, our complete work is still in progress and our risk evaluation model will be adjusted along with the growth of our own log data recorded by our server in production. That is why we use the term "Initial Model" in this paper.

In this early phase, We evaluate the performance of the initial model using Tourism and CTF1 datasets based on $Top-k$ with window-size equal to 10. Number of k candidates that we can tolerate in the scoring process defines how precise we want the model to be. If k equal to 1, it makes our model very strict and will not tolerate slightly different behaviour. It is ideal for system security but in production it can trigger more false alarms, and it is not system admins and users favor. On the other hand, The bigger the number of candidates, will make our model more relax and yields higher performance result. But it comes with consequences, the model will become less sensitive.

Window-size accommodate the length of EventID in 1 behaviour. It defines how much information should a user carry for risk evaluation. A further examination can be perform to find the best window-size for a particular log data. Moreover, this examination will be resource-aware since bigger number of EventIDs can direct to bigger computational cost. As we observed, window-size equal to 10 is the balanced point for CTF1 dataset and our computational power.

TABLE I
PERFORMANCE EVALUATION

Top-k	**Dataset**	
	Tourism	***CTF1***
1	0.9987	0.8273
3	0.9988	0.9457
5	0.9989	0.9774

*Window-size = 10

As describe in Table I, our model performs very well on the Tourism dataset with the accuracy 0.9 in all the variation of $Top-k$. This dataset consider too easy for our model due to the lack of users involved. In fact, there only 1 user who dominate the access to the server while the others only record very limited log entries. It seems our model learns this particular user behaviour dominantly so that the performance is always perfect regardless the number of k.

On the contrary, with bigger log data and more users contained in CTF1 dataset, our model yields a reasonable result and yet promising. In the sensitive setting, with $Top-k$ equal to 1, the initial model obtains a very descent performance at 0.8 accuracy. Whilst in the more relax setting, our model perform reasonably well. In line with the increment of k, the performance rise up significantly. The model achieves 0.95 and 0.98 for k equal to 3 and 5 respectively. This is very promising since CTF1 dataset resemble our future own dataset with a large amount of log entries and thousand of users involved. We hope in the future, when our system deploys into production, this initial model can be implemented with a little bit of adjustment. This will accelerate our deployment and save a lot of resources.

V. Conclusion and Future Work

Our early stage User Behaviour Risk Evaluation shows a promising future in term of initial model performance. Accuracy on large scale log data proves that our framework is ready for production level system with the growth of log data in the server. Even with the most sensitive setting, our model performance is still acceptable, besides the number goes higher as we lessen its sensitivity.

We developed our input sequences by utilizing the identity feature, which is the user ID, in our log entry. Our model were trained based on user historical behavior (within the same user role) portrayed by these sequences. We suspect that a periodic time-based partition might also be beneficial in this stage. With the help of periodicity, we can capture the surroundings situation right when user make a connection to the server. This will enrich the information that we can retrieved to build the User Behavior Risk Evaluation model. So, we will explore more on this in our future work.

Subsequently, our scope in this study focuses on user behavior evaluation by analyzing log data from the company web server. There are other production servers working as a whole in an organization to run the company business in day-to-day operation. These servers also have users interacted with them that need to be evaluate as well. In the future, we will expand our strategy in this work to broader servers, to address the cybersecurity issues and network threats caused by user behaviours.

References

[1] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST Special Publication 800-207, DOI: 10.6028/NIST.SP.800-207, August 2020.

[2] Executive Office of the President, Improving the Nation's Cybersecurity, Document number 2021-10460, pp. 26633–26647, May 2021.

[3] C. Nobles, "Botching Human Factors in Cybersecurity in Business Organizations," HOLISTICA, vol. 9, pp. 71–88, DOI: 10.2478/hjbpa-2018-0024, December 2018.

[4] A. A. Moustafa, A. Bello, and A. Maurushat, “The Role of User Behaviour in Improving Cyber Security Management,” Frontiers in Psychology 12:561011, DOI: 10.3389/fpsyg.2021.561011, June 2021.

[5] J. Liu, J. Zhu, S. He, P. He, Z. Zheng, and M. R. Lyu, ”Logzip: Extracting Hidden Structures via Iterative Clustering for Log Compression,” IEEE/ACM International Conference on Automated Software Engineering, 2019.

[6] P. He, J. Zhu, S. He, J. Li, and Mi. R. Lyu, ”Towards Automated Log Parsing for Large-Scale Log Data Analysis,” IEEE Transactions on Dependable and Secure Computing, 2018.

[7] S. He, J. Zhu, P. He, and M. R. Lyu, ”Loghub: A Large Collection of System Log Datasets towards Automated Log Analytics,” arXiv:2008.06448, 2020.

[8] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, ”Drain: An Online Log Parsing Approach with Fixed Depth Tree,” IEEE International Conference on Web Services, 2017.

[9] M. Du, F. Li, ”Spell: Online Streaming Parsing of Large Unstructured System Logs, ”IEEE Transactions on Knowledge and Data Engineering, 2018.

[10] M. Du, F. Li, G. Zheng, and V. Srikumar, “DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning,” ACM SIGSAC Conference on Computer and Communications Security, DOI: 10.1145/3133956.3134015, October 2017.

[11] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, “LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs,” International Joint Conference on Artificial Intelligence, pp. 4739–4745, DOI: 10.24963/ijcai.2019/658, 2019.

[12] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J. Lou, M. Chintalapati, F. Shen, and D. Zhang, “Robust Log-Based Anomaly Detection on Unstable Log Data,” ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 807—817, DOI: 10.1145/3338906.3338931, August 2019.