High-Confidence Computing 2 (2022) 100034

Contents lists available at ScienceDirect

High-Confidence Computing

homepage: www.elsevier.com/locate/hcc

User behaviour analysis using data analytics and machine learning to predict malicious user versus legitimate user

Rohit Ranjan[a,*], Shashi Shekhar Kumar[b]

[a] Innovaccer Analytics, Noida, UttarPradesh, India
[b] IIIT Allahabad, Prayagraj, Uttar Pradesh, India

ARTICLE INFO

Keywords:
Application security
Big data analytics
Machine learning
Random forest
Behavioral analysis
Prediction

ABSTRACT

Research-based on user behavior analysis for authentication is the motivation for this research. We move ahead using a behavioral approach to identify malicious users and legitimate users. In this paper, we have explained how we have applied big data analytics to application-layer logs and predicted malicious users by employing a Machine Learning algorithm based on certain metrics explained later in the paper. Machine Learning would present a list of IP addresses or user identification tokens (UIT),deduced from live data which would be performing a malicious activity or are suspected of malicious activity based on their browsing behavior. We have created an e-commerce web application and induced vulnerabilities intentionally for this purpose. We have hosted our setup on LAMP [1] stack based on AWS cloud [2]. This method has a huge potential as any organization can imply this to monitor probable attackers thus narrowing down on their efforts to safeguard their infrastructure. The idea is based on the fact that the browsing pattern, as well as the access pattern of a genuine user,varies widely with that of a hacker. These patterns would be used to sort out the incoming traffic from and list out IP addresses and UIT that are the most probable cases of hack attempts.

1. Introduction

Applying big data analytics and machine learning on data obtained from application-layer logs would yield a list of probable candidates for malicious attempts. Plenty of work has been done in the field of cyber security and data analytics, but in this paper, we have proposed a new approach to predict a list of probable hackers. This approach is based on the application of Big Data Analytics with Machine Learning.

Abramson and Aha [3] proposed the idea of user identification based on their web browsing behavior. It Not only identifies but also differentiates between users based on their web browsing behavior. *Shi et al.* [4] gave the idea of implicit authentication in which they proposed authentication of users based on their behavior patterns. *Al-Khazzar and Savage* [5] proposed how user authentication can be performed by using information collected from user behavior in reaction to a 3D Graphical maze. Each user had a unique reaction to the graphical maze which was the idea behind identification.

Frank et al. [6] investigated if a classifier can authenticate users continuously based on the way users interact with the touchscreen of a smartphone. They have proposed a set of 30 behavioral touch features that can be extracted from touchscreen logs and demonstrated that different users populate different sub-spaces of this feature space. In simple words, it means every user has a different and unique touch pattern. Pusara and Brodley [7] has presented an approach for re-authentication of user based on the data collected from the computer mouse. Their underlying hypothesis was that user behavior can be successfully modeled based on user invoked mouse movements as every user has a unique mouse movement pattern. *Bergadano et al.* [8] presented a measure for keystroke dynamics which limits the instability of biometric feature for User authentication through keystroke dynamics. This idea considered the notion that every user has a unique keystroke dynamics or keystroke pattern.

Moritz et al. [9] presented behavioral profiling methods and systems for authenticating the user. *Zhauniarovich et al.* [10] published a Survey paper about malicious domain detection using DNS Data Analysis. It showed how analysis of data can yield behavioral information which can be used further to gain critical information. *Yang et al.* [11] proposed a secure model based on multi-cloud. This algorithm was used to upload encrypted data onto clouds and also, supporting SQL queries with encrypted data. In their model, they performed most computations in the clouds to increase query efficiency. Liu [12] talked about current security requirements and corresponding engineering concepts, techniques, and approaches. Different research challenges were identified, and the security models and strategies for open data repositories were also examined. *Al-Shomrani et al.* [13] focused on privacy and security. He explained how Managing security policy is a challenge, which their frame-

* Corresponding author.

https://doi.org/10.1016/j.hcc.2021.100034
Received 17 April 2021; Received in revised form 8 July 2021; Accepted 15 July 2021

Nomenclature

δ	Number of Processes per services in Server
ω	Virtual Memory of Analytics Cluster
α	Physical Memory for Analytics Cluster
β	Shared Memory for Analytics Cluster
γ	Percentage of CPU
θ	Percentage of Memory
Ma	Total Number of malicious Traffic
Le	Total Number of legitimate Traffic
To	Total Traffic on Server
IAM	Identity and Access Management
P_M	Prediction Model Testing after Testing

work would handle for big data. He also advocated about privacy policy being flexible, integrated, context-aware, and customizable.

Lee et al. [14] has proposed a methodology to develop and deploy a big data platform that can be used to collect, process, and store huge Terabytes of event and flow logs. *More et al.* [15] has proposed a new threat detection system that uses big data to analyze the cyber attacks over cloud networks in lesser time. *Shiva et al.* [16] has proposed a security approach of game theory. In this approach author has considered the interaction between the attacks and the defense mechanisms as a game played between the attacker and the defender. *Bhavani et al.* [17] has proposed a mix of the decision tree and random forest algorithms to order any strange conduct in the traffic of the system. So, once we integrate Data Analytics findings with Machine learning (ML) we would have a system that is learning from life as well as historical data, and eventually, at some point in time, it would be a system that would be extremely intelligent and robust.

1.1. Motivation

In the literature listed above, all the proposed solutions and research work is focused on Network and Transport Layers,and those proposed over Application Layers were concerned with authentication only. In-Network and Transport Layer, information is in the form of Packets or Segments and Datagram which is not in an effective stage to be parsed as they are in crude form. Application layer information is in human-readable form and hence easier to parse for user behavior. Writing parsers for application-layer logs is a way more effective than processing Network and Transport Layer traffic. We are keeping our scope confined to the application layer in this research. Also, plenty of research has been performed on behavioral authentication. The behavioral analysis can be utilized for the greater good and not only limited to authentication. In this paper, we propose a new method to predict malicious users from a legitimate users. This is achieved by applying big data analytics and machine learning to application layer logs. We have used Random forest for this research along with decision trees, binary classification, Clustering, and time series to compare the output to select the best-suited approach.

1.2. Research contribution of this work

Based on the information explained above, following contributions are presented in this paper.

- Vulnerable application has been hosted on the internet which generates logs for our experiment and research.
- Big Data Analytics has been used to retrieve and contain big data from the live data stream.
- After pulling out the application server logs,we apply Big Data Analytics with custom metrics to trace out the meaningful data which can be put to use.

Table 1
Symbols Used.

Symbols	Description
η	fills for training sets
ζ	alternative method of fills for training data
∂	filling of missing value
μ	input types in fuzzing
π	traffic from user agents
κ	Category 1 for Regression
ϵ	Category 2 for Regression
σ	Geo location identification for incoming traffic

- Predictions from machine learning algorithm decision tree is applied to live traffic.Then with a machine learning algorithm, the random forest has been applied to it.
- Training data was used at first from logs of 48 hours which is later tested using the subset of the same logs.
- After the maturity of this model, when this model is put on to live traffic, we are able to attain 70% accuracy for the prediction of a malicious user (Table 1).

1.3. Organization

The road map of the paper is followed as:- System model and problem formulation are explained in Section 2. The proposed scheme is illustrated in Section 3. The performance evaluation is discussed in Section 4. Finally, Section 5 concludes the article with future scope.

2. System model and problem formulation

In this paper, we have used Big Data Analytics and Machine Learning Algorithm to predict legitimate user and malicious user on Application Layer logs generated by user browsing patterns by feeding our system real-time data sourced from our application launched on the internet. Before the problem formulation, following definitions are used in the proposal.

2.1. Resources

To get a prediction, we have used the metrics which are most relevant to user behavior. These metrics are extracted from server logs in real-time which contains the traffic generated by users browsing on the web application. Once these logs are parsed and fed to big data analytics & machine learning prediction algorithm, we get a list of malicious & legitimate users based on the IP and UIT. We are deducing the outcome by identifying IP and UIT as they are easier to manage for any Network Security team via a dashboard.

2.1.1. Importance

Importance is given to variables which in our cases are the metrics based on which decision tree and later random forest would form. So, we define Accuracy and Gini impurity for our model.There are two measures of importance for each variable in a random forest.The first measure is based on a decrease in accuracy when a variable is excluded.Further,it can be broken down by outcome classes. The second measure is based on the decrease of Gini impurity when a node is split as per chosen variable. a) Accuracy: Accuracy for metrics in our model would be their precise nature of content after they are being parsed. We define accuracy (Acc) as provided below:

$$Acc = \frac{Parsed\,Data}{Total\,Data} \times 100 \times \sum_{data=0}^{data=livestream} Logs \tag{1}$$

Gini importance Every time a split of a node is made on variable m,the Gini impurity criteria for the two descendent nodes m_{child} is less than the parent node.Adding up the Gini decreases for each variable over

all trees in the forest which gives fast variable importance. It can be deduced mathematically as given below:

$$Gini = 1 - \sum_{i=1}^{m} (P_i)^2 \tag{2}$$

where, P_i is the probability of an object being classified to a particular class.

2.1.2. Missing value replacement for the training set

Random forests have two ways of replacing the missing values. The first way has two modes. The first mode is fast and it says that if the variable m is not categorical,then a median of all the values is calculated in class j and missing values are replaced with the same calculated value. Now,if the variable m is categorical, then the missing values are replaced with the most frequently occurring value in class j. These replacement values are called fills represented as (η).

$$\sum_{i=1}^{n} \frac{val_1 + val_2 + ... + val_n}{n} = \eta \tag{3}$$

The second way of replacing missing values is more expensive computationally. It has given a better performance than the first, even with a large amount of missing data. This method replaces missing values only in the training set.It begins by doing a rough, random, and inaccurate filling of the missing values. Then it performs a forest run and computes proximity. This is represented as (ζ) as shown below:

$$x_{m,n} = \sum_{i=1} n \leftarrow mostfrequentnonmissingvalue = \zeta \tag{4}$$

2.1.3. Missing value replacement for the test set

When there is a test set, there are two different methods of replacement depending on labels for the test set. If labels exist,then the fills derived from the training set are used as replacements. Hence, Eq. (3) would suit this case.

If labels do not exist, then each case in the test set is replicated n_{class}, where n_{class}= number of classes.

$$Case(C) \subseteq TestSet(T) \tag{5}$$

$$Case(C) \leftarrow Replicated \times n_{class} = \partial \tag{6}$$

Where, ∂_1 fills missing values in Class1.

$$\partial_1 \leftarrow Class1 \tag{7}$$

Where, ∂_2 fills missing values in Class1.

$$\partial_2 \leftarrow Class2 \tag{8}$$

and so on the process continues till n.

2.1.4. Unsupervised learning

In unsupervised learning, data consists of a set of x-vectors of the same dimension with no class labels or response variables.There is also no figure of merit to optimize, hence leaving the field open to ambiguous conclusions. The usual goal is to cluster the data. This is done if the data falls into different types and number of piles, each of which can be assigned some meaning.

$$Original\,Data \leftarrow data1 \tag{9}$$

$$createSyntheticDatafromOriginalData \leftarrow data2 \tag{10}$$

$$data2 \leftarrow Sampling \leftarrow Random \tag{11}$$

2.1.5. Balancing prediction error

In some data sets, the prediction error between classes is highly unbalanced. Some classes have a low prediction error while others have a high. This occurs when one class is much larger than another. In this case, random forests try to minimize the overall error rate. This will keep the error rate low on the large class while letting the smaller classes have a larger error rate (ER). Below Eq. (12) represents error rate for true case, and Eq. (13) gives false cases.

$$ER_{True} = \frac{T(T+F)}{100} \tag{12}$$

$$ER_{False} = 1 - ER_{True} \tag{13}$$

$$ER_{Total} = \frac{ER_{True} + ER_{False}}{100} \tag{14}$$

2.1.6. Prediction accuracy

The ratio of the number of correct predictions to the total number of input samples determines its accuracy. Accuracy is also dependent on the quality of datasets and the corresponding metrics.

$$A = \frac{Numberofcorrectpredictions}{Totalnumberofpredictions} = \frac{TP+TN}{T} \tag{15}$$

$$A_{Prediction} = (\int_{i=0}^{i=datasets} Logs + \int_{i=0}^{i=min} Error + A) \times 100 \tag{16}$$

3. Proposed scheme

In this research, we are using big data analytics along with machine learning to predict the malicious users. These users were identified by their IPs and User Identification Tokens(UIT), as per their suspicious browsing pattern. This idea is based on analyzing live traffic and predicting malicious users in real-time before they commit any offence.This proactive approach would prevent any attack or breach even before it is about to happen. This model after reaching a maturity threshold is very effective and can be used to protect the assets of an internet-based organization. Any attacker before initiating an attack on an application browses it with the instincts of a hacker and hence browsing pattern is entirely different from that of a legitimate user. The attacker's motive is to exploit the application or corresponding infra or both, while a simple user wants to use the application to fulfill his/her needs. So our proposed solution can detect an attacker as he/she performs recce of application to attack, before initiating the real attack.

3.1. Metrics

To get the prediction of server logs data, we have used metrics that are most relevant to user behavior. These metrics are extracted from server logs which contain the traffic generated by users browsing on a web application. Once these logs are parsed, processed via a data analytics framework, and fed to a machine learning prediction algorithm, we get the list of suspicious users from incoming source IPs or UIT.

3.1.1. Time

User behavior for time decides if a user is trying to browse or a hacker is fuzzing. This means the number of hits or calls generating from a particular IP address or a user-specific token can be measured. A Hacker tends to analyze the application at first and their hits would be low but once the analysis is done hits would exponentially rise but it's vice versa for a user.

$$Time \propto \frac{1}{Hacker}, \forall Hacker \in User \tag{17}$$

$$Time \propto User, \tag{18}$$

$$Time = \sum_{time=0}^{time=live} Time_{Browse} + Time_{Fuzz} \tag{19}$$

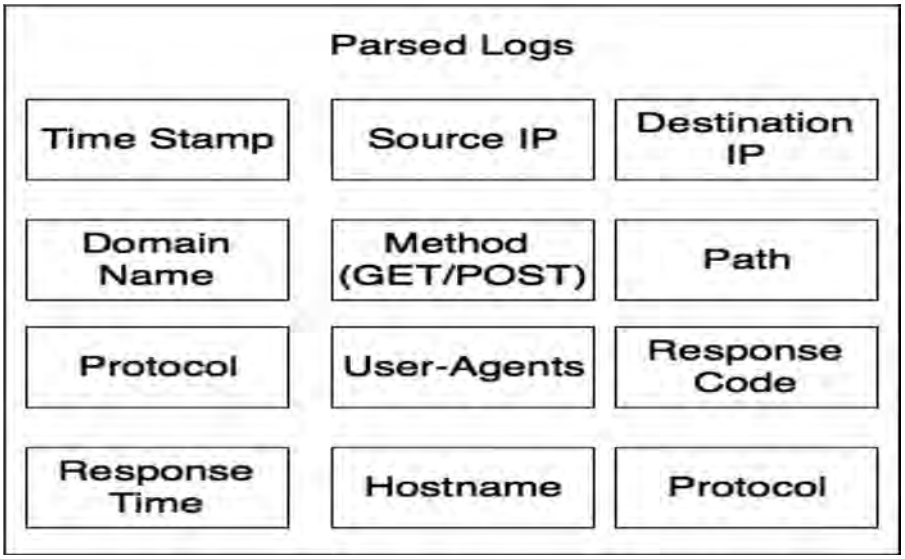

Fig. 1. Contents retrieved on parsing Application Layer logs.

3.1.2. Fuzzing

The methodology of passing different types of input to a parameter to find out how an application responds to various input types in fuzzing. This won't necessarily comprise of the payloads but different sets of expected and unexpected inputs for the application to assess how a server responds. This helps an attacker to understand the attack surface and prepare an attack vector. Where, μ represents input types in fuzzing.

$$\sum_{i=0}^{\infty}(\mu_{valid} + \mu_{invalid}) = Fuzzing \tag{20}$$

3.1.3. Timestamp

This metrics is used to find out the total number of incoming traffic per IP or user ID in a given time interval. We used traffic per second as a time interval to calculate T.S from each IP or user ID. For our experiment, the value of T.S was comparatively low but for public web portals, it has to be even less.

Deducing from Eqs. (17) and (18), we conclude,

$$T.S = \frac{\sum_{time=0}^{\infty}\Delta H_i}{time\ per\ user} \tag{21}$$

where, ΔH_i is several hits on the portal, i shows the time per second, and *time per user* is the set of active users at an instance (per second) on our experimental portal.

3.1.4. Payloads

Inbound traffic could also contain malicious payloads from attacker. This makes it easier to detect and there are hundreds of solutions like SIEM solutions, available in market to detect these payloads. Also,there are solutions to prevents from these attacks like Web Application Firewall. Logs containing WAF payloads are best source of metrics and they can be used for training data.

$$WAF = \sum_{i=0}^{\infty} Logs_{Payloads} + Tr_{Data} \tag{22}$$

where, Tr_{Data} is Training Data.

$$Tr_{Data} = \sum KnownPayloads + \sum_{time=0}^{time=live}\rho \tag{23}$$

3.1.5. Malicious reputations

Incoming traffic from IPs is also marked based on their reputation. A very know portal providing information about the reputation of IPs is AbuseIPDB [18]. The probability of malicious users coming from flagged IPs is very high.

$$P(M) = \frac{FlaggedIP}{TotalincomingIP} \tag{24}$$

3.1.6. User agents

A lot of inbound traffic,particularly from script kiddies are automated tools.These tools have specific user agents and signatures. Hence, it's very easy to identify and catch. We used this as a metric for prediction. We represented traffic from user agents with π

$$\sum_{User-Agents}^{Live-Traffic} + \sum_{User-Agents}^{Training-Data} = \sum_{time=0}^{time=\infty}\pi \tag{25}$$

3.1.7. Brute force

The process of making multiple synchronized attempts primarily to break authentication can be a perfect metric for prediction. Hack attempts made from particular IP or user identification token can be analysed to find out the crux of incoming traffic and the rate of flow of incoming traffic.

$$\int_0^{\infty}(BF) = \sum_{traffic}(payload + fuzzing) \tag{26}$$

3.1.8. Geo location

Incoming traffic from an unexpected geo locations with an unexpected rate is another ideal metric for predicting attacker traffic. This can be understood from the fact that inbound traffic from a non-English speaking nation onto a web portal serving English content is suspicious. This idea is again an ideal metric for prediction along with other metrics. We are calculating this metric as shown below:

$$\sigma = \int_{a=live}\frac{IP_a}{Hits} + \int_{time=0}^{live}\frac{Hits}{Time} \tag{27}$$

3.2. Random forest

Among numerous Machine Learning algorithms available, Random Forests (RF) suits our requirements the best than any other. The prime motive behind selecting Random Forest is the idea of deciding for a particular user. This decision is based on the fact that we have multiple metrics which would facilitate the decision-making process with a random forest algorithm. This decision is administered by the Classification

and Regression approach. In the regression approach, the output variable in the regression is numerical (or continuous) and is categorical (or discrete) in classification. In our research, we focused on Classification as we need Yes or No in prediction specifically 1 (Malicious) or 0 (Legitimate). Since we needed output for our case to be in 1 or 0, RF was the best available algorithm. This means, data can be categorical as well as numerical, so let's define function approximation using mapping function (f), input variables (x), and output variables (y).

$$f(x) = y, \forall x, y \in Variables \tag{28}$$

3.2.1. Classification

Classification algorithms map function (f) from the input variables (a) to discrete and categorical output variables (b).

$$b = \begin{cases} 1, & \text{if Yes.} \\ 0, & \text{otherwise. (No)} \end{cases} \tag{29}$$

3.2.2. Regression

Regression algorithms map function (f) from the input variables (a) to numerical and continuous numerical output variables (b).

$$b = \begin{cases} \kappa & \kappa \in Category1. \\ \epsilon & \epsilon \in Category2. \end{cases} \tag{30}$$

In equation below Data Set is derived from the raw Data set. This Data set is used to obtain the training set and testing set.

$$DataSet \leftarrow CompleteDataLog \tag{31}$$

3.3. Training set

A training set is a subset of the overall data set which is used to train a model. This data set is used to educate models based on metrics about how to make predictions. This would be the first attempt to teach a model about metrics.

$$TrainedSet \subset DataSet \tag{32}$$

3.4. Testing set

A testing set is used to test the trained model. This is done to verify if applied to check the working of training data. The prediction model is put on trial before making it live. Also, TrainigSet would always be much larger than TestSet, $TrainingSet >> TestSet$

$$TestSet \subset DataSet \tag{33}$$

$$\int_{Training}^{Test} P_M = \sum TrainingSet + \sum TrainedSet \tag{34}$$

3.5. Prediction

Once the training set of data is ready, it is fed to the testing phase. This leads to prediction based on training data. We used R-Studio [19] for the prediction.

$$\sum left(y_i) - (y_{L*})^2 + \sum right(y_i) - (y_{R*})^2 \tag{35}$$

where yL* = mean y-value for left node yR* = mean y-value for right node

3.6. Log parser

Log parser is used to parse application-layer logs to be fed into the Data Analytics platform. Raw logs are fed to log parser which churns out all the necessary parameters of log into CSV format. Any ordinary parser would do this work with some customization. The same is shown in the Fig. 1 below as well. Also, in Algorithm 1 ALP explain the process of parsing logs for Big Data Analytics. We took different metrics from live logs and they were fed into our proposed mechanism.

$$csv \leftarrow LogParser \leftarrow RawData \tag{36}$$

Algorithm 1: Application Log Parser (ALP).

Input: Data from Application Server Logs
Output: Parsed data in csv based on metrics supplied
$livedatastream1 \leftarrow accesslogs$
$livedatastream2 \leftarrow errorlogs$
$\theta \leftarrow (livedatastream1 + livedatastream2)$; `// Fetching live traffic from web server`
$\sum_{i=0}^{\infty} \leftarrow \theta_{m=0}^{livedatastream}$; `// Collecting live data stream in cache`
$StorageServer \leftarrow \sum_{i=0}^{theta}$; `// Saving cached data in storage server`
$\lambda \leftarrow StorageServer_{j=0}^{j=livedata}$; `// Data reached analytics server`
$Parser \leftarrow \lambda_m^{data}$
while *(Parser Data) != 0)* **do**
  $\gamma \leftarrow \lambda$; `// Data is passed to analytics algorithm`
  $output file \leftarrow \gamma$
  while *(output file)!=0* **do**
    verify contents; `// verify data being written into file`
    if $(contents) \leftarrow correct$ **then**
      continue
      exec function(output file); `// Call execution function to get desired metrics`
    else
      break loop; `// Exit from loop in case of error`
      exit
    end
  end
  if *(wait for live stream of data)* **then**
    wait for 5 seconds; `// Wait in case of delay recheck`
    exec function(output file); `// Call execution function to get desired metrics`
  else
    error in configuration; `// configuration error, so exit program`
    break loop
    exit
  end
  $\iota \leftarrow \sum_{i=0}^{\theta}$
  return ι
end

4. Performance evaluation

The algorithm below parse the logs from Web Server and convert them into a CSV file which can be easily processed to run data analytics solutions Algorithm 1. After CSV is obtained, it is fed to Execution Function Algorithm 2 which yields parsed dataset as per the metrics. This dataset is used to create a Decision Tree that yields nodes with values wither 0 or 1. We then calculate ROC AUC scores using scikit learn. metrics.roc_auc_score function [20]. Finally, Random Forest Algorithm 3 is used based on metrics to give the final result of prediction (Tables 2 and 3).

4.1. Numerical settings

We used Magento Software [21] and established the entire infrastructure on AWS cloud. We have used the Pandas package in Python & Sci-Kit [22] for Machine Learning. This setup has been hosted on an EC2 instance of AWS cloud with a Security Group open to public internet on all ports.

Algorithm 2: Execution Function Algorithm.

Input: Data fed into Execution Function
Output: Data Parsed as per metrics
exec(CSV) ← ALP(ι)Select X, Where, $x \in m$, x ≪ m, $m \in totalnumberof features$, x ∈ selected no. of features$d \leftarrow (\sum_{m=1}^{13} metrics) D_a \leftarrow$ d, d ∀ Nodes; `// calculate the node d using the best split point`
$counter = 0$; `// Total number of daughters`
$a = 0$; `// Variable topass daughter nodes in Random Forest`
while $(i \leq D_a)$ **do**
  d = BestSplit ← i; `// i represents Nodes return d, Where d ∈ DaughterNodes`
  $i = i + 1$; `// Split nodes into daughter nodes using best split`
  $counter = counter + 1$; `// number of nodes has been reached`
end
if $(a \leq counter)$ **then**
  RandomForest ← d
  $a = a + 1$
else
  break loop
end
exit

Algorithm 3: Random Forest Application Algorithm (RFAA).

Input: Data Set from Decision Tree
Output: Output in Numerical values of 0 and 1
$\sum_{i=0}^{Node=n} \Pi(DecisionTree) \leftarrow RandomForestTree \leftarrow RandomForestClassifier$; `// Multiple Decision Trees constitutes a Random Forest`
$Tree \in DecisionTreeClassifiermodel \leftarrow RandomForestClassifier(n_{estimators} = 100, bootstrap = True, max_feature =' sqrt')Training \leftarrow Tree_{TrainData}$; `// Training Data for Tree`
$\Gamma \leftarrow (model.fit)\Gamma_{train} \leftarrow model.fit(train, train_{table*})$; `// Model is ready to make predictions on the testing data`
$Prediction \leftarrow \Gamma Prediction_{Class} = \Gamma.predict(test.data) Probability_{Class} = \Gamma.predict(test.data)[:, 1] metrics \leftarrow roc\_auc\_score(y, y\_pred, average = None)$; `// Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores`
$roc_{value} \leftarrow roc\_auc\_score(test.data, Probability_{Class}) Feature \leftarrow metric(x), pandas$; `// pandas module in python`
$\Theta \leftarrow pandas.DataFrame(Q, I, S, V)$; `// cleansed data set passed to data frame function in pandas`
$\Theta.head() \leftarrow 1/0$Where,$Q = List(trainingdatacolumns) I = mode(featureimportance) S = sort_values V = S(importance, ascending = FALSE)$**if** $\triangle \leftarrow \Theta.head(1))$ **then**
  print Malicious
  Report to Dashboard
else
  print Legitimate
end

4.1.1. Practical setup

- Set up Website for 35 days on the internet
- Advertise its presence in the Hacker community
- Advertise juicy information content or bug bounty to lure attackers
- Make a realistic web portal and not some honeypot or vulnerable webserver

Table 2
Data Processing Hardware Requirements.

Configuration	Specifications
Memory	2GB
Build Server	2GB
Web Node	1GB
DB	30GB
MySQL	30GB
Cores	64

Table 3
Hardware Configuration of Web Server.

Processor	OS	HDD	Memory	SSD
8	Ubuntu 16.04	250GB	16GB	NO
8	CentOS 7.1	200GB	24GB	NO
4	CentOS 7.1	100GB	8GB	Yes
16	CentOS 7.1	100GB	16GB	Yes

Table 4
Data Set Size.

Window	Size (in GB)	Number of Samples
Day 1-5	0	0
Day 6-10	4.8	1689001
Day 11-15	5.2	1865189
Day 16-20	3.4	1421943
Day 21-25	4.1	1541412
Day 26-30	7.2	1992817
Day 31-35	5.7	1911124
Total	29.6	10421486

- Web server framework is available at disposal to use and they need to be configured
- We used AWS for setting up the LAMP stack and hosting our web portal
- We would be logging all the hits in DB (PostgreSQL/MySQL/MongoDB)
- Projection would be derived from this data

4.1.2. Experimental setup

Setting up experiment required deployment of a live web portal on the internet which should be vulnerable, so it can attract malicious traffic. To prepare a setup for live traffic, we used Linux Ubuntu 16.04 64-bit, Apache Web Server, MySQL DB, and Magento framework [21], which is an e-commerce platform written in PHP. We intentionally made Magento framework vulnerable with OWASP Top 10 [23] and SANS Top 25 [24] vulnerabilities. This setup was hosted on the Internet with AWS Elastic IP and was made publicly accessible on the internet. The idea was to implement a Honeypot-like application that would yield necessary data for the completion of research work. Idea behind this experimental setup was to invite hackers and users to browse on our application and record their usage pattern based on the clicks. Component level diagram of deployed architecture is shown below in Fig. 2.

4.2. Results

The results are obtained based on the algorithms designed in the proposed scheme.Live traffic is captured and passed to ALP Algorithm 1. This algorithm provides us parsed data as per specific needs as input to Execution Function Algorithm 2. This algorithm provides CSV as per metrics. This CSV is inserted as input for Random Forest Algorithm(RFA) 3 and the random forest is generated. As shown in Table 4 which depicts the size of the data set and the number of samples which essentially contains the logs which were analyzed to fetch the outcome of this research. So in our computation approx. 10 million lines of application-layer logs

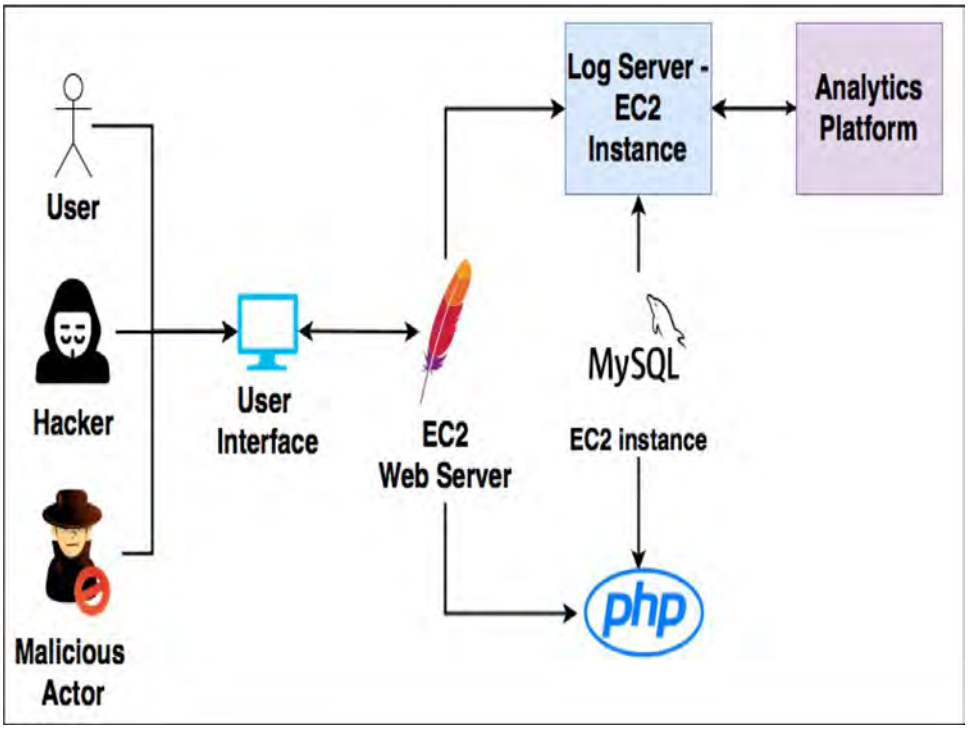

Fig. 2. Setting up Experimental E-Commerce Portal.

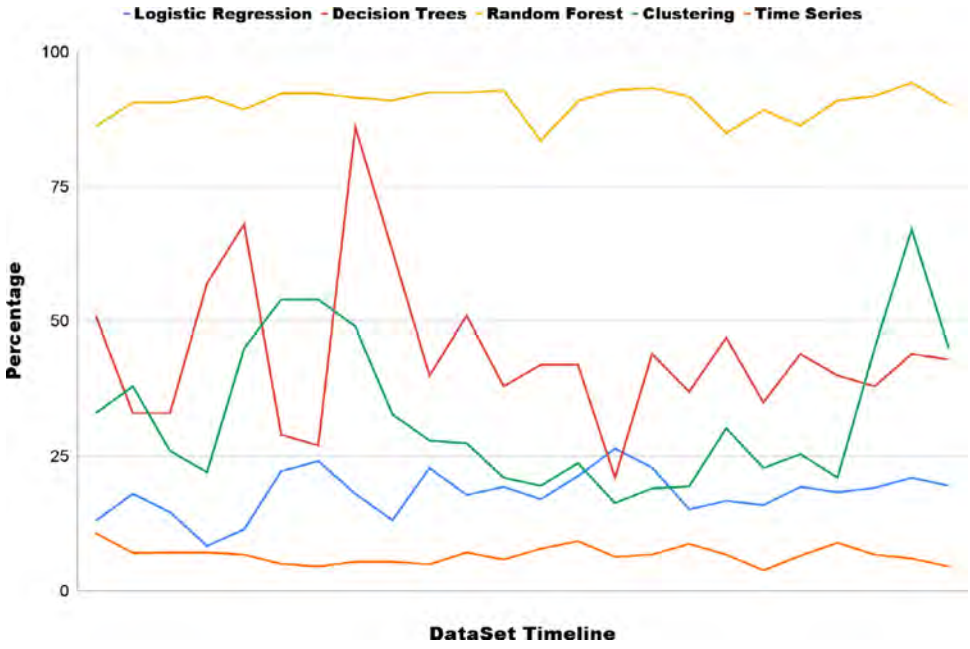

Fig. 3. Output Comparison of Algorithms.

were parsed and processed. The first five days are not counted to make sure that we get a cooling period before the site gets to know that it contains vulnerabilities in the community.

4.2.1. Makespan

A comparison of multiple regression approaches was made and it was seen that the Random Forest Algorithm has the best results. We found out that as dataset content increases, time delay decreases. This means that once the training set has completed processing, and processing of testing set takes place, time delay starts decreasing as shown in Fig. 3. Once maturity is reached a constant delay is there for live prediction but it's negligible and can be ignored.

4.2.2. Execution time of prediction

Execution time is with a delay of few seconds for live traffic in the beginning. As the system matures, this execution time is reduced to its minimum. Among multiple algorithms compared, Random Forest stood tall and ahead of all other model algorithms and the same is depicted in Fig. 4.

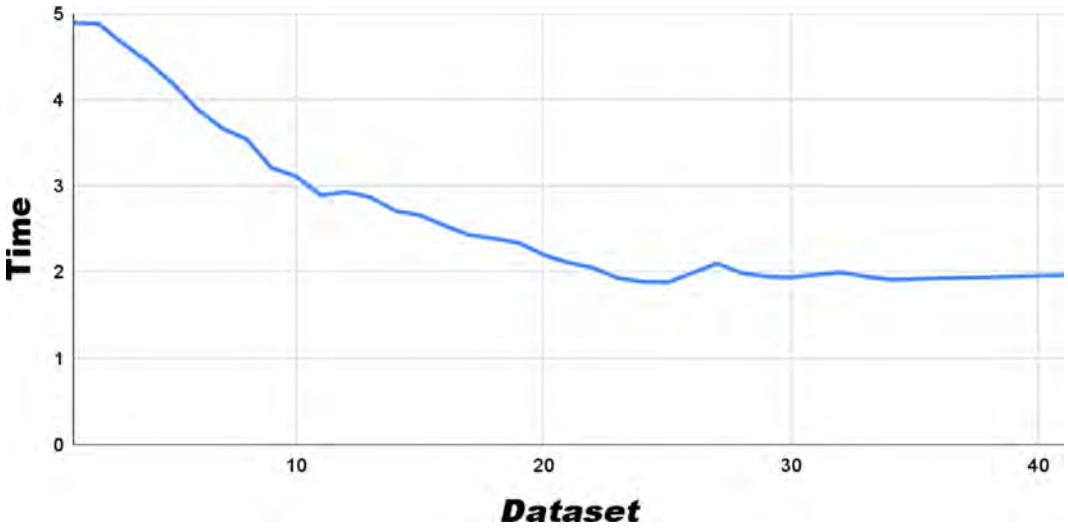

Fig. 4. Workflow Time.

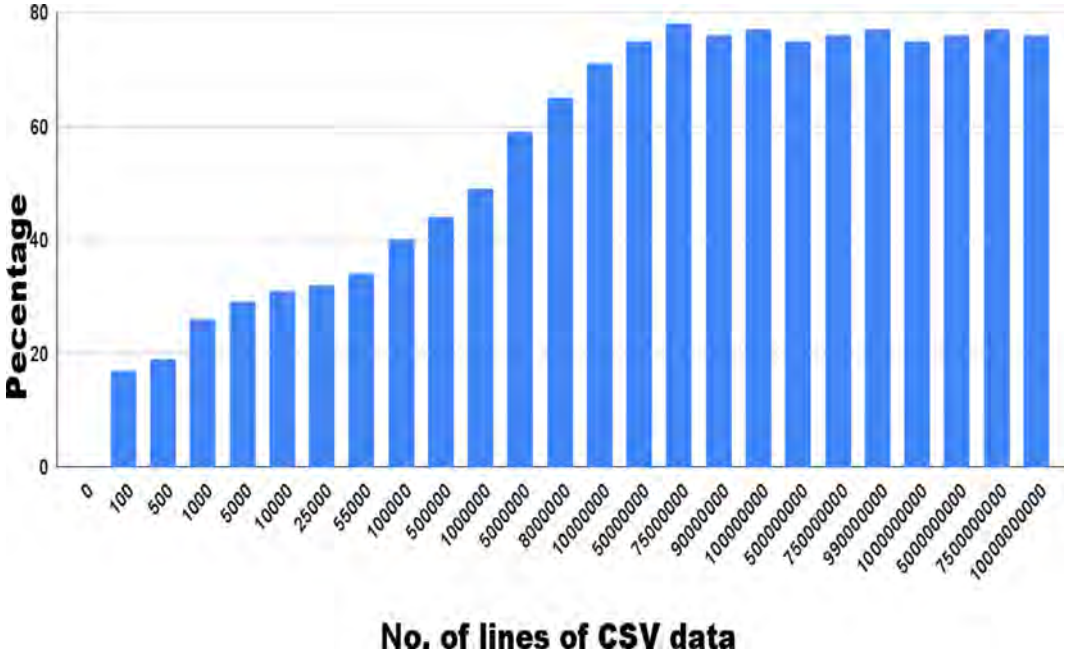

Fig. 5. Impact of Training Data.

4.2.3. Workflow completion

As evident from Fig. 5, an evident 70% accuracy was achieved. This accuracy was obtained when Algorithm 3 was used and the maturity of the model was reached. This model was slowly maturing and as training data kept on increasing so does the accuracy of prediction.

4.2.4. Throughput

The process of Data Analytics and Machine Learning is a resource-intensive process but the outcome of resource utilization has a far-reaching outcome.As the system model matures,so does its throughput increases. As evident from Fig. 6.

4.3. Unexpected outputs

Since Data processing is a task that is prone to outputs which is not an expected outcome. In Random Forest Algorithm these concerns are addressed. Since output was 1 for positive results and 0 for negative, output was shown clearly in Fig. 7. Our model clearly distinguished between legitimate and malicious traffic based on IP and UIT. There were instances where predictions were wrong as shown in the figure below as colors mismatch.

4.4. Network traffic

Internet traffic is not only restricts the HTTP/HTTPS but other protocols even for a web application. In our experimental setup we have observe that SSH, TCP/UDP, DNS, etc. protocols being actively used as shown in Fig. 8. This shows effectiveness and coverage of our experiment as we captured almost all forms of internet protocols and attack targeting those protocols.

5. Conclusion

In this paper, we propose a novel technique for the prediction of malicious users from web application traffic. We have applied Big Data Analytics to analyze huge amounts of data that would be generated by

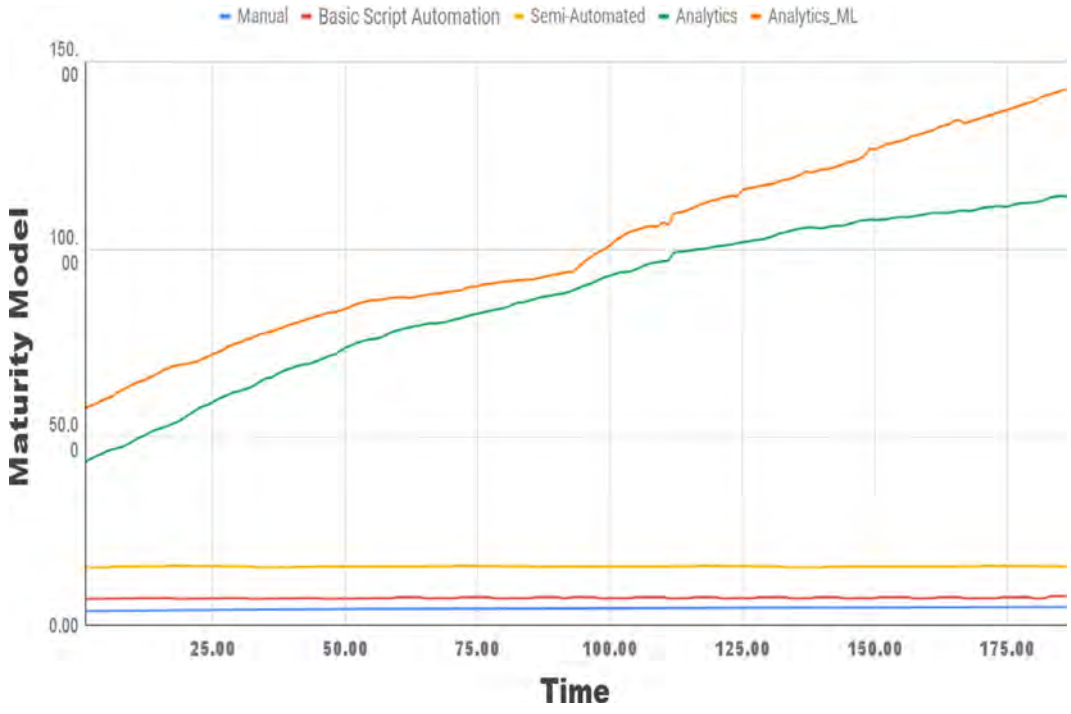

Fig. 6. Comparison of different Prediction Strategies.

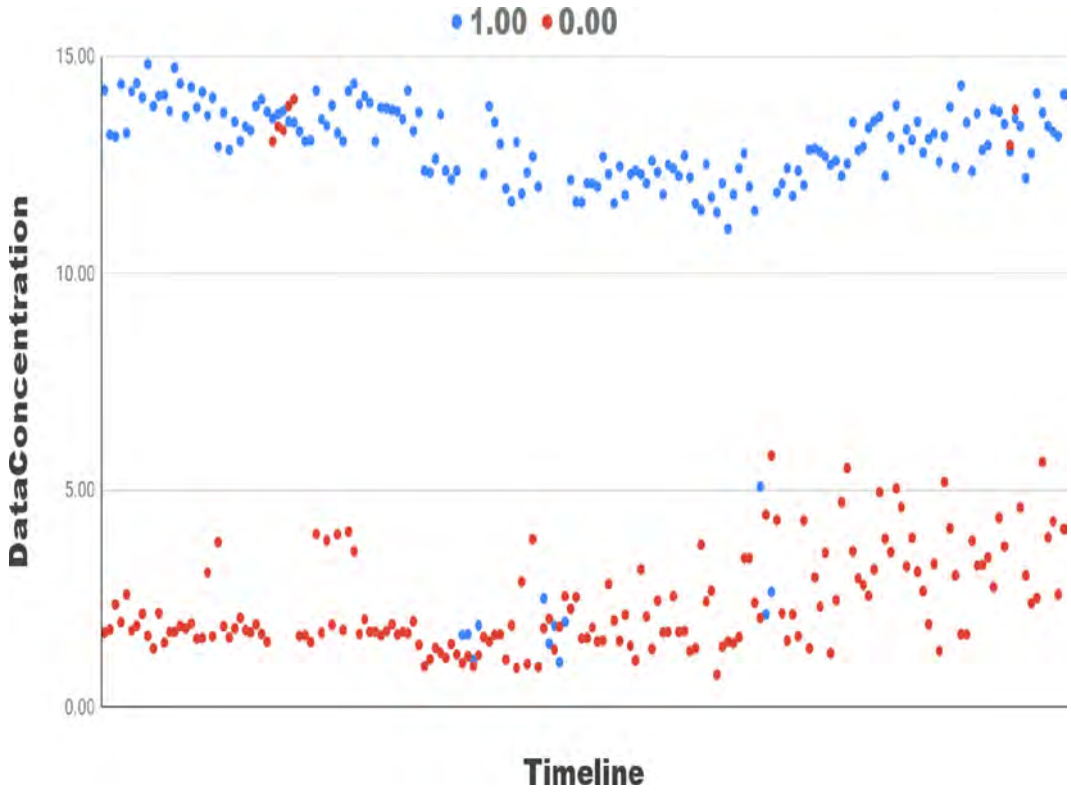

Fig. 7. Data Concentration.

a public e-commerce website. We have hosted a website on the internet and it was thrown open for public access for fifty days with peak traffic for thirty days. The logs have been generated in the real-time which is used to predict malicious users. Once data is analyzed we feed this data to the Random Forest algorithm which results in Yes or No based on the training data set provided to it. Training data set were prepared manually by analyzing the big data. This training data set was crafted in such a way that all the metrics for prediction were covered. Final results predicted malicious users with 65-70% accuracy. The entire process has been done in real-time with live data flowing in continuously. In the future, we would aim to obtain 90% accuracy and reduce resource utilization as well as a time delay to almost zero. Also, we plan to work on

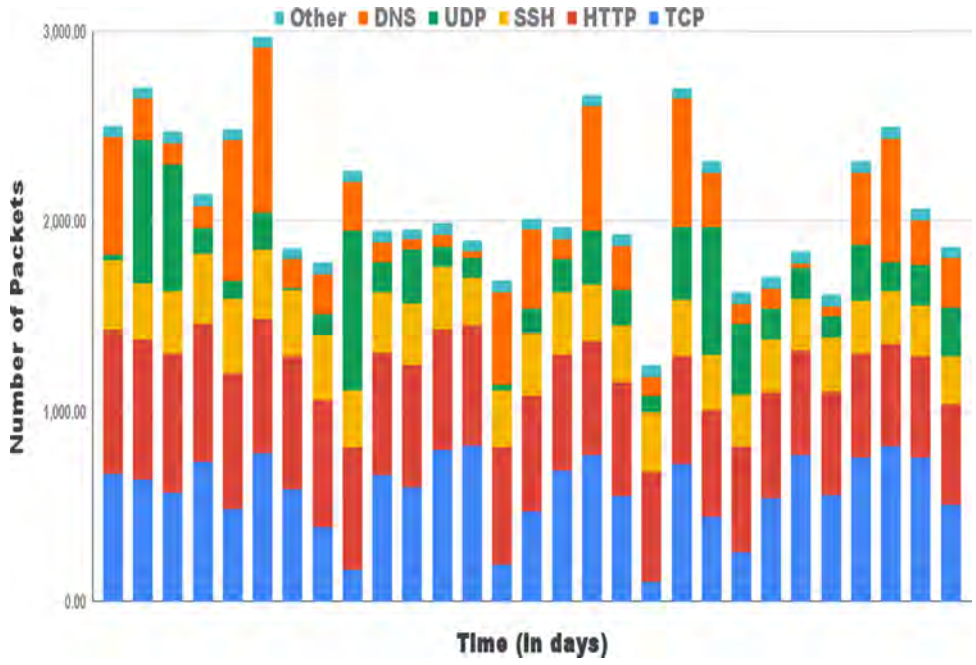

Fig. 8. Types of Incoming Traffic.

implementing Artificial Intelligence (AI) and remove the need for any supervision in ML. Finally, we would explore if we can implement this approach at Network and Transport Layer data.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Potter, What is a lamp stack?, 2018. [Online]. Available: https://www.liquidweb.com/kb/what-is-a-lamp-stack/
[2] Amazon Web Services, Online, AWS [Online]. Available: https://aws.amazon.com, 2006.
[3] M. Abramson, D. Aha, User authentication from web browsing behavior, in: The Twenty-Sixth International FLAIRS Conference, 2013.
[4] E. Shi, Y. Niu, M. Jakobsson, R. Chow, Implicit authentication through learning user behavior, in: International Conference on Information Security, Springer, 2010, pp. 99–113.
[5] A. Al-Khazzar, N. Savage, Graphical authentication based on user behaviour, in: 2010 International Conference on Security and Cryptography (SECRYPT), IEEE, 2010, pp. 1–4.
[6] M. Frank, R. Biedert, E. Ma, I. Martinovic, D. Song, Touchalytics: on the applicability of touchscreen input as a behavioral biometric for continuous authentication, IEEE Trans. Inf. ForensicsSecur. 8 (1) (2012) 136–148.
[7] M. Pusara, C.E. Brodley, User re-authentication via mouse movements, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, 2004, pp. 1–8.
[8] F. Bergadano, D. Gunetti, C. Picardi, User authentication through keystroke dynamics, ACM Trans. Inf. Syst. Secur.(TISSEC) 5 (4) (2002) 367–397.
[9] K. Moritz, S.S. Aultman, J.J.A. Campbell, D. Casillas, J.E. Neuse, S.T. Alonzo, T.B. Buckingham, G.C. Fernandez, M.K. Mortensen, Behavioral Profiling Method and System to Authenticate a User, US Patent 9,185,095, 2015.
[10] Y. Zhauniarovich, I. Khalil, T. Yu, M. Dacier, A survey on malicious domains detection through DNS data analysis, ACM Comput. Surv. (CSUR) 51 (4) (2018) 1–36.
[11] Z. Yang, L. Wang, X. Song, Secure model based on multi-cloud for big data storage and query, in: 2016 International Conference on Advanced Cloud and Big Data (CBD), IEEE, 2016, pp. 207–214.
[12] L. Liu, Security and privacy requirements engineering revisited in the big data era, in: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), IEEE, 2016, p. 55.
[13] A. Al-Shomrani, F. Fathy, K. Jambi, Policy enforcement for big data security, in: 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), IEEE, 2017, pp. 70–74.
[14] J.-H. Lee, Y.S. Kim, J.H. Kim, I.K. Kim, K.-J. Han, Building a big data platform for large-scale security data analysis, in: 2017 International Conference on Information and Communication Technology Convergence (ICTC), IEEE, 2017, pp. 976–980.
[15] R. More, A. Unakal, V. Kulkarni, R. Goudar, Real time threat detection system in cloud using big data analytics, in: 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE, 2017, pp. 1262–1264.
[16] S. Shiva, S. Roy, D. Dasgupta, Game theory for cyber security, in: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, ACM, 2010, p. 34.
[17] T.T. Bhavani, M.K. Rao, A.M. Reddy, Network intrusion detection system using random forest and decision tree machine learning techniques, in: First International Conference on Sustainable Technologies for Computational Intelligence, Springer, 2020, pp. 637–643.
[18] I. Marathon Studios, Abuseipdb, 2019, [Online]. Available: https://www.abuseipdb.com
[19] community.rstudio.com, R studio crash course, 2019[Online]. Available: https://rstudio.com
[20] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.
[21] M. software, How to get the magento software, 2019[Online]. Available: https://devdocs.magento.com/
[22] SciKit, Team scikit, 2019[Online]. Available: https://scikit-learn.org/stable/
[23] OWASP, Owasp top ten, 2017. [Online]. Available: https://owasp.org/www-project-top-ten/
[24] SANS, 2020 cwe top 25 most dangerous software weaknesses, 2020[Online]. Available: http://cwe.mitre.org/top25/archive/2020/2020.