Research Article

Detection Of Dynamic Vulnerabilites In Hadoop Systems For Controlling The Fuzzy Adaptive Security Profiles (FASP)

SATHISHA M.S, [1] K.C. RAVISHANKAR, [2]

[1]Department of Computer Science and Engineering,Canara Engineering College, Mangaluru, Karnataka, India
Affiliated to Visvesvaraya Technological University (VTU) Belagavi, Karnataka, India
E-mail: *sathishams1983@gmail.com*

[2]Department of Computer Science and Engineering,Govt. Engineering College, Hassan, Karnataka, India
Affiliated to Visvesvaraya Technological University (VTU) Belagavi, Karnataka, India
E-mail: *kcrshankar@gmail.com*

Article History: Received: 5 April 2021; Accepted: 14 May 2021; Published online: 22 June 2021

Abstract
Hadoop is big data processing framework with capability to process large volumes of data using map reduce parallel processing paradigm. Big data analytics on these large volumes of information provides various intelligent information for business process optimization and governance. With wide acceptance of Hadoop for big data analytics, there is also an increased security vulnerability. In our earlier work [1], Fuzzy Adaptive Security Profile (FASP) was proposed to provide increased security to Hadoop processing platform. The work has shortcoming in terms of protection against wide variety of security vulnerabilities. The approach considered only denial of service attack. Common vulnerability Exposure (CVE) has detailed 23 different vulnerabilities in Hadoop and this work designs a vulnerability scanner based on Hidden Markov model to detect the CVE attacks specific to Hadoop. The designed vulnerability scanner is integrated with FASP using an adaptive security scoring technique to trigger adaptive mitigation mechanisms.

Keywords: Hadoop, Fuzzy Adaptive Security Profile, vulnerability scanner, Hidden Markov Model.

INTRODUCTION

Hadoop is an open source big data analytical platform. It has combination of components – HDFS (Hadoop distributed File System) for storage, data processing using MAP-REDUCE framework and resource management using YARN. With Hadoop processing on Cloud, security and privacy of data becomes challenging and these vulnerabilities must be detected in time and defensive mechanism must be built to protect the data from those attacks. Due to usage of cloud, problems like confidentiality of data, access control for users and privacy protection becomes an issue. Both internal and external attacks arising out of Network security breaches can result in data leakages.
Hadoop was not designed with consideration for security in initial releases. The focus of initial design was on efficient processing of large volume of data. Following were not focused for security in Hadoop.

1. Authentication of users and service
2. Auditing
3. Authorization against impersonation attack
4. Data security in HDFS

Hadoop has become a most popular data analytics tools due to its cost effectiveness and performance. Globally many enterprises are adopting Hadoop for its business intelligence. Following are the important factors influencing the rapid adoption of Hadoop.

1. Ability to work with both structured and unstructured data
2. Efficient and affordable processing of services
3. The business intelligence reports gathered using Hadoop data analysis helps enterprises in sound decision making.

But with these advantages, the security issues are Hadoop hinder its rapid adoption.
Securing the Hadoop platform against security vulnerabilities will increase the adoption rate of Hadoop in industries. Towards this end many solutions have been proposed for providing security and privacy to data in Hadoop platform. Following are in the scope of these solutions

1. Data security and access control
2. Ensuring data privacy
3. Preventing from data corruption

In aim to provide enhanced security to Hadoop platform, performance must not be compromised. With this goal, we have proposed an adaptive security solution using Fuzzy Adaptive Security profile in [1]. In this work personalized and adaptive security was provided for the data in HDFS based on the analysis on current security risk for the data. The solution relied on two core functions of performance and security risk monitoring. Based on the performance requirements and current security risks, the security levels were adaptively controlled using Fuzzy logic. Vulnerability scanner is an important function in this solution which detect the security vulnerabilities and maps to a risk score. But in work [1], vulnerability scanner assessed only two problems of crash attack and denial of service attack. In this work, we extend the vulnerability scanner to detect many vulnerabilities and score the Hadoop system against the vulnerability. The vulnerability scanner can detect the vulnerabilities using machine learning models and uses learning approach to detect the risks due to the vulnerabilities.
Following are the contributions in this work.

1. Analysis of sequence of events resulting in Common Vulnerabilities and Exposures (CVE) in Hadoop platform.
2. Modeling the attack in terms of Hidden markov model to provide a suspicious score
3. Adaptive Mitigation mechanism in Hadoop based on the suspicious score

There has been no earlier work on analysis of common vulnerabilities and Exposure (CVE) in Hadoop platform. To the best of our knowledge, this is the first work to model the sequence of events culminating in vulnerability in Hadoop and measuring the proximity of sequence of events to the vulnerability using a suspicion score.

RELATED WORK

In [2] authors proposed “Pangr”. It is a vulnerability detection tool. The vulnerability analysis is done based on monitoring the behavior of binaries, stack and heap overflow. This behavior-based vulnerability detection using binaries slows down the performance of Hadoop systems. Authors in [3] proposed a denial of service attack detection framework for Hadoop. The denial of service through flooding TCP-SYN, HTTP and ICMP message were detected using this framework. The counter-based detection scheme is based on fixed threshold and this scheme has large false positives. Due to this it may be blocking genuine traffic too in peak load conditions. Deep learning-based vulnerability detection framework called SyseVR is proposed in [4]. Programs are represented in terms of vectors accommodating the syntax and semantic information concerning vulnerabilities. The approach works only if the source is available, but most cases source code will not be shared and thus its applicability is limited. Authors in [5] used machine learning for vulnerability detection. Lightweight static and dynamic features are extracted and analysis is done to predict if its vulnerable. The system can detect memory corruptions only. Machine learning based DOS attack detection is proposed in [6]. From training dataset, attack signatures are extracted and matched using machine learning algorithms to detect attack. Following features were extracted from the flows – entropy, coefficient of variation, quantile coefficient, rate of change and a random forest classifier is trained to classify the DOS attack based on the features. Authors in [7] analyzed the impact of insider attacks and security issues in Hadoop cluster. The study was conducted in three dimensions of attacks from compromising nodes, malicious users and network intruders. The authors analyzed port scanning attack, dictionary attack, computer exploit attack, man in middle attack and authentication bypass attack. Merkle tree-based verification of map reduce jobs is proposed in [8] for vulnerability risk on map reduced jobs. Merkel hash is computed on the map reduce outputs and verification is done on it to detect corruption. In [9] cross cloud map reduce framework is proposed to check the integrity vulnerabilities of map reduce tasks. Hybrid cloud is used in this solution. Integrity verification is done on private cloud leaving rest of operations to public cloud. Random task replication, random task verification and credit accumulation are the strategies adopted for integrity violations. Authors in [10] proposed a detection framework called InTect to detect the invader jobs vulnerability and prevent Hadoop system from performance degrade. Features are extracted from the jobs and support vector machine classifier is built to classify the invader jobs. Authors in [11] proposed a mechanism to predict failures in Hadoop systems. Clustering is done to group similar error sequences. The clusters are then used to train a Hidden Markov Model (HMM) to predict failure. Authors in [12] proposed a genetic algorithm-based solution for denial of service attack detection. Genetic algorithm is used to profile the incoming packets to detect features of packets. Based on the features, detected entropy analysis is done to detect DDOS attack.

PROBLEM DEFINITION

Most solutions for vulnerability detection are designed to detect only denial of service failures. Common Vulnerabilities and Exposures (CVE) has detected 23 security vulnerabilities in Hadoop [13]. The majority of this vulnerabilities can be launched via map reduce operations too. Securing Hadoop platform against these vulnerabilities is important to prevent privacy leakages and data corruption attacks on HDFS. The security of Fuzzy Adaptive Security Profile (FASP) proposed in [1] can be enhanced if the vulnerability scanning process can detect the vulnerabilities defined by CVE. This work designs a machine learning based vulnerability scanner to detect the some of the security vulnerabilities defined by CVE.

Vulnerability scanner for FASP

The architecture of the FASP solution is given in figure 1. Vulnerability scanner is an important module in the FASP which detects DOS attacks and protects the Hadoop system from DOS attacks. It also identifies the nodes crash based on past history of failures. In this work we extend the vulnerability scanner for some of the vulnerabilities defined by CVE. The vulnerabilities addressed in this work is detailed in table 1.

The vulnerabilities caused in two major ways

1. Execution of commands
2. Leakage of data through network interfaces

Following are the vulnerabilities caused through execution of commands

- User permission can be modified to deny access to data during the map reduce execution. (V1)
- Arbitrary commands can be executed causing slowdown of map reduce jobs (V2)
- A configuration file with directions to refer sensitive data is constructed by the malicious users (V3)
- Impersonation can be done through map reduce jobs (V4)

Following are the vulnerabilities caused through leakage of data through network interfaces

- Leakage of passwords and sensitive information via job execution (V5)
- Through map reduce, file sharing can be done. (V6)
- Map reduce jobs can be used for tool for password leakage. (V7)
- Through map reduce operation, remote client can get write access to blocks. (V8)
- Leakage of block token can be done by map reduce job. (V9)

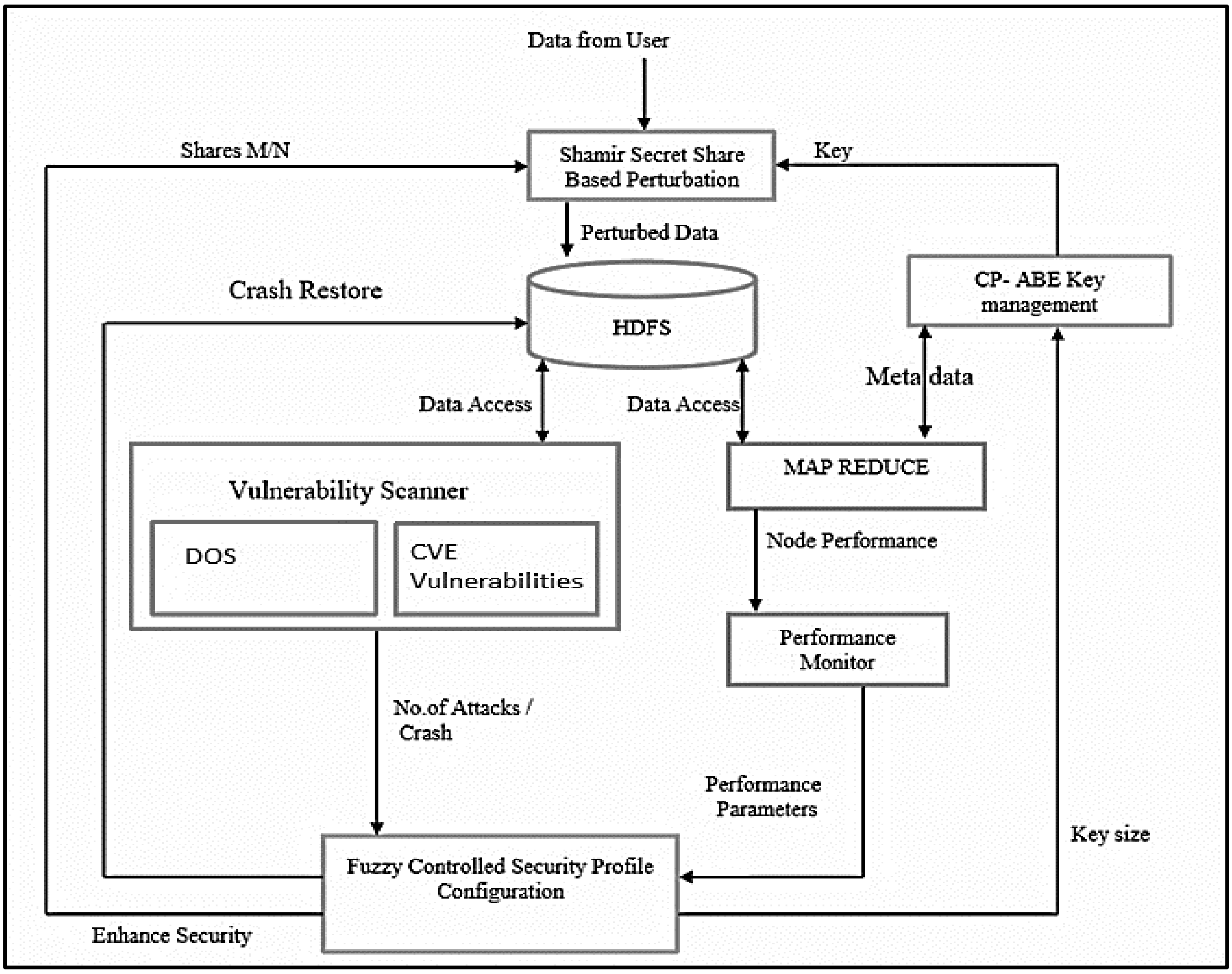


Figure 1: FASP Architecture

Table 1: Vulnerabilities addressed

Sl.No	**ID**	**Details**	**Internal attack mode**
1	CVE-2018-11767	Incorrect grant of permission to the users	User permission can be modified to deny access to data during the map reduce execution.
2	CVE-2018-11766	Invalids command execution with root access	Arbitrary commands can be executed causing slowdown of map reduce jobs
3	CVE-2017-15718	Password leakage for applications	Leakage of passwords and sensitive information via job execution
4	CVE-2017-15713	Exposing private files	The malicious user can construct a configuration file containing XML directives that reference sensitive files on the MapReduce job history server host.
5	CVE-2017-3166	Sharing of sensitive files	Through map reduce, file sharing can be done.

6	CVE-2016-5001	Grant illegal access to files	Leakage of block token can be done by map reduce job
7	CVE-2016-3086	Leakage of password	Map reduce jobs can be used for tool for password leakage
8	CVE-2012-3376	Providing write access to user who have only read access	Through map reduce operation, remote client can get write access to blocks.
9	CVE-2012-1574	Impersonation of authenticated users	Impersonation can be done through map reduce jobs.

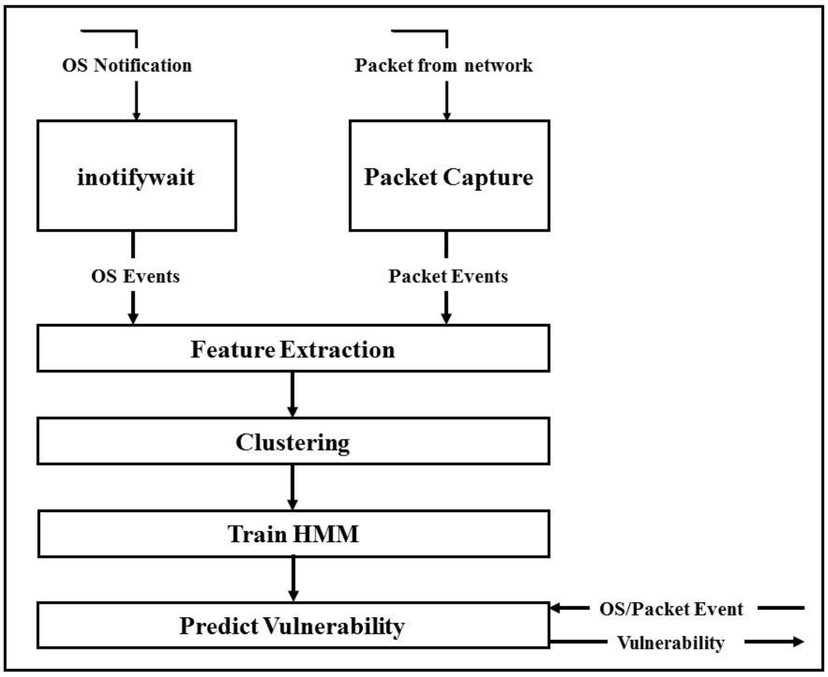


Figure 2: Vulnerability detection process

Vulnerabilities due to execution of commands can be checked by monitoring of commands executed on OS. Linux file related commands can be monitored using inotifywait utility.

Vulnerabilities due to leakage of data through network interfaces can be checked by monitoring the packets going in and out of interfaces.

The vulnerability detection process is given below. OS Notification and Packet from network are captured and provided to feature extraction module. Feature extraction module extracts essential features from the OS notifications and packets from network and converts them to events. The events are sequenced based on its presence in the session and grouped. The event sequences are clustered. Following procedure is followed to cluster the dataset.

Let D of n elements be the dataset to be partitioned to K clusters. The data D is split to K parts as $D = \cup_{k=1}^{K} S_k, S_{k1} \cap S_{k2} = \emptyset , k_1 \neq k_2$. The partition is done using modified k-means algorithm with density sensitive distance metric. The density sensitive distance metric between two points is calculated as

$$D_{ij} = min_{p\varepsilon P_{i,j}} \sum_{k=1}^{|p|-1} L(p_k, p_{k+1}) \quad \text{------------(1)}$$

The cluster centers c_k is obtained in modified k means as optimal solution of

$$minz = \sum_{d^j \varepsilon S_k} \left\| x - d^{(j)} \right\|, x = (x_1, x_2, \dots x_m) \; \varepsilon \; R^m| \quad \text{------------(2)}$$

‖. ‖ represents the 2-norm. The optimal solution can be determined using nonlinear equations as

$$\frac{dz}{dx_i} = \sum_{d^{(j)} \varepsilon S_k} \frac{(x_i - d_i^{(j)})}{\|x - d^{(j)}\|} = 0, i = 1,2, \dots m \quad \text{------------(3)}$$

Let $||x - d^{(j)}||$ be represented as q_j , the above equation can be written as

$$\sum_{d^{(j)} \varepsilon S_k} \frac{(x_i - d_i^{(j)})}{q_j} = 0, \text{ i=1,2...m} \quad \text{----------(4)}$$

The above equation can be rewritten as

$$x_i = \sum_{d^{(j)} \varepsilon S_k} \frac{d_i^{(j)}}{q_j} / \sum_{j=1}^{n} \frac{1}{q_j}, i = 1,2..m \quad \text{-----------(5)}$$

It can be further represented using iterative formula as

$$x_i^{k+1} = \sum_{d^{(j)} \varepsilon S_k} \frac{d_i^{(j)}}{q_j^k} / \sum_{j=1}^{n} \frac{1}{q_j^k} \quad \text{------------(6)}$$

Where $q_j^k = ||x^{(k)} - d^{(j)}||$

For iteration, the initial point can be taken as average of the points

$$x_k^0 = \sum_{d^{(j)} \varepsilon S_k} \frac{d_i^{(j)}}{|S_k|}, \text{ k= 1...K}$$

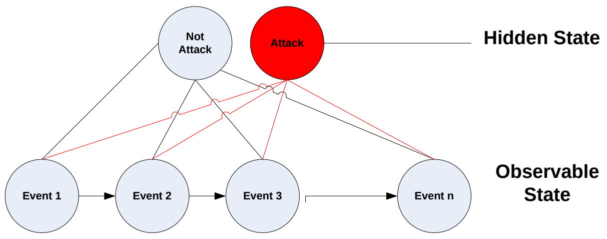
These clusters are tagged manually to 10 labels (nine labels for V1 to V9 and one for normal).

For all labels from V1 to V9, corresponding HMM model is built which gives the final state of “Attack” or “No Attack”. HMM is used in this work to predict “Attack” or “No Attack” based on the event sequence. The event sequence is given as input to the HMM model. The transition matrix for the HMM model is created using the labelled event sequences. HMM is characterized by three units Hidden states $X = \{x_1, x_2, x_3\}$, Observations state $Y = \{y_1, y_2, y_3\}$ and transition probabilities$A = a_{ij} = \{P[q_{t+1} = x_j \,|q_t = x_j\,]\}$ and emission probabilities $B = b_{ij}$. HMM can be represented as

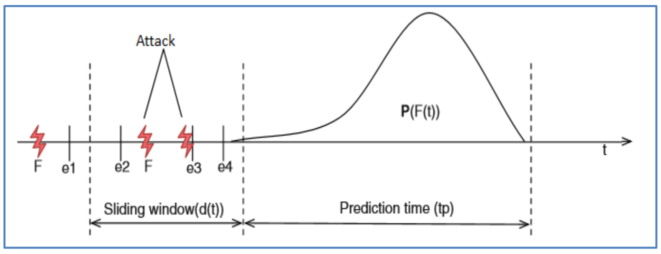
$$\lambda = \quad (\pi, A, B)$$

A is the state transition matrix. Each entry gives the probability of transition from one state to another. B is the emission matrix providing the probability of observingY_t called$b_j(Y_t)$. The initial transition matrix is given by π.

The observation symbols are the events in the system given as $O_1 = \{e_1, e_2, e_3, \dots, e_n\}$. Events are provided as inputs to the HMM model and model transitions to Attack state when the event sequence happen to a violation as per the specific class label.



The classified event sequences are used as input to train the model. The event sequence is the sequence of events happening within a sliding window of length Δt as given below



In the figure above, F are the places where Attack or vulnerability happens. Till absorption state is reached, state transition occurs. The learning speed and accuracy depends of the values chosen for time step. The initial state transition probability π is fixed as 0.5. In training stage, the most likely state transition sequence and the model parameters $\lambda = (\pi, A, B)$ are computed. The optimum values for the parameters π, A, B are obtained during the training. With the goal to maximize likelihood of sequence, parameters are maximized. For initial steps, number of states, number of observations, transition probability and emission probability are pre-specified. The initial parameters are calculated from the past observation, such that the model can predict accurately from the initial phase. Parameter value gets optimized as training process. Expectation Maximization algorithm is used for training the HMM model.

Model parameters are optimized based on maximum likelihood in this algorithm. Starting from random seed, increase the number of iterations for HMM model to settle. Training must be done to effectively represent error sequences and to check model transmits to failure state on failure. To do effective training, in this work we propose to use a new training strategy faster than Baum-Welch algorithm and gradient-descent techniques.

The idea in this approach is formulating the probability of the observation sequenceO_t , O_{t+1} pairs and use the Expectation Maximization algorithm to learn the model λ

Table 2: Packet Features

FEATURE NAME	DESCRIPTION
Duration	Length of the connection
Protocol_type	TCP, UDP, etc.
Service	Application layer services like http, telnet, etc.
Src_bytes	Size of payload from source to destination
Dst_bytes	Size of payload from destination to source
Flag	Status of the connection
Wrong_fragment	The total number of fragments received corrupt
Urgent	The total number of fragments which are urgent
Num_failed_logins	Number of times login failed
Logged_in	For successful login it is 1, 0 otherwise
Num_compromised	Count of compromised
Root_shell	1 in case of root shell
Su_attempted	1 in case of Su attempted

Num_root	Total count of root access
Num_file_creations	Total number of files created
Num_shells	Total number of shell prompts opened
Num_access_files	Number of operations on access-controlled files
Num_outbound_cmds	Outbound command in FTP session
Is_hot_login	Boolean indicator for hot entry
Is_guest_login	Boolean indicator for guest login
Count	Number of connections to the same host as the current connection in the past two seconds
serror_rate	Rate of SYC error connects
rerror_rate	Rate of RERR connections
same_srv_rate	Rate of same service connections
diff_srv_rate	Rate of different service connections
srv_count	Number of connections to the same service as the current connection in the past two seconds
srv_serror_rate	Services that have SYC error
srv_rerror_rate	Services that have REJ error
srv_diff_host_rate	Rate of connections to different hosts

Table 3: OS event features

FEATURE NAME	DESCRIPTION
Command name	Name of the command executed
Access parameters	Access parameters passed to command
Access login	1 in case of super user, 0 otherwise
Passwords in command result	1 in case of password in the command result, 0 otherwise
Sensitive access	1 in case of access of sensitive folder, 0 otherwise
File access changed	1 in case of file access permission changed, 0 otherwise
Config file created	1 in case if config file is created and 0 otherwise.
Command access failures	Number of times command failed due to access permission
Number times access permission downgraded	Number of times access permission downgraded from higher to lower say from read only to write.
File reads from sensitive locations	1 in case file is read from sensitive location
File read network out cooccurrence	Ratio of cooccurrence of file read and network packet out event.

There are following two steps in EM.

Expectation Step (E)	A function is created to calculate log-likelihood from current estimate
Maximization Step (M)	Calculation of parameters to maximize the expected log-likelihood function found during Expectation step.

Optimal state sequence is found using Viterbi algorithm. Viterbi finds optimal state sequences of Markov chain. The sequence of states is calculated using Viterbi algorithm for the states $S = \{S_1, S_2, \dots S_n\}$ such that

$$S = argmax_s P(S, O, \lambda)$$

Viterbi algorithm returns an optimal state sequence of S. At each step t, the algorithm allows S to retain all optimal paths that finish at the N states. N optimal paths are computed at time t+1. Once training is completed, the model is used for predict attack or not attack. The observed events sequence passed to the HMM model, to predict the attack or not attack for the corresponding label.

The output of HMM model is a suspicious score and if the suspicious score is greater than a threshold, the sequence of events is decided as attack.

The suspicious score(L) for the HMM model is calculated as

$$L = W_1 S_1 + \sum_{k=2}^{N} \frac{1}{P_{i,j}} W_j S_j^k \quad \text{------------}(7)$$

$P_{i,j}$ is the probability of state transition from state i to state j

W_j is the weighting factor of state j.

S_j^k is the suspicious score of state j comparing with k observed state.

RESULT

The performance of the proposed vulnerability detection is compared with machine learning models – Naïve Baiyes, SVM and Neural Network. Since this is first work on predicting the CVE vulnerability in Hadoop clusters, the comparison is done with other machine learning models. The performance metrics are collected from machine learning models applying the methodology given in figure 3.

Features given in table 3, are extracted from the OS events and features mentioned in table 2 are extraction from packets. The training data set is created with these features and two classes of vulnerability or no vulnerability. Three different machine learning models – SVM, Neural Network and Naïve Baiyes are trained using the dataset. The test set is classified using the three machine learning models and following performance measures are collected

- Accuracy
- Sensitivity

- Specificity

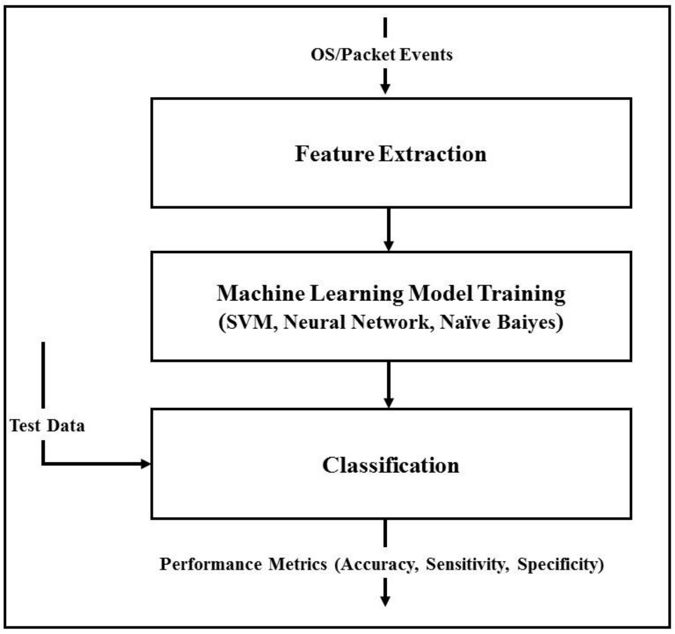

OS/Packet Events
Feature Extraction
Machine Learning Model Training
(SVM, Neural Network, Naïve Baiyes)
Test Data
Classification
Performance Metrics (Accuracy, Sensitivity, Specificity)

Figure 3 Machine Learning models

The neural network is trained with following parameters.

Table 4: Neural Network parameters

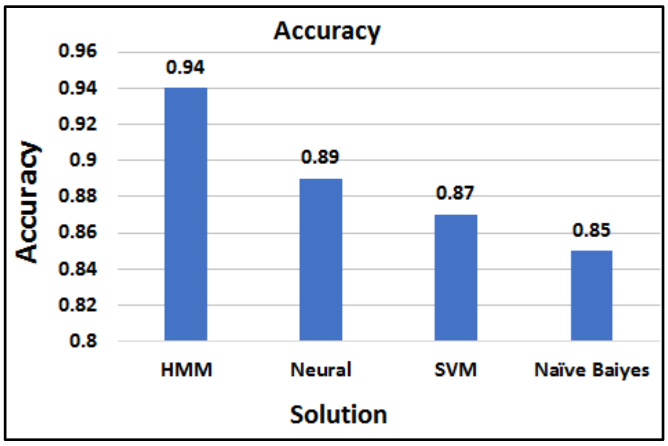
Parameter	**Values**
Layer count	3 Layers
Input Layer neurons	42
Hidden Layer neurons	82
Output Layer neurons	2
Max Iterations	1000
Error Rate	0.01

The SVM is trained with following parameters

Table 5: SVM Parameters

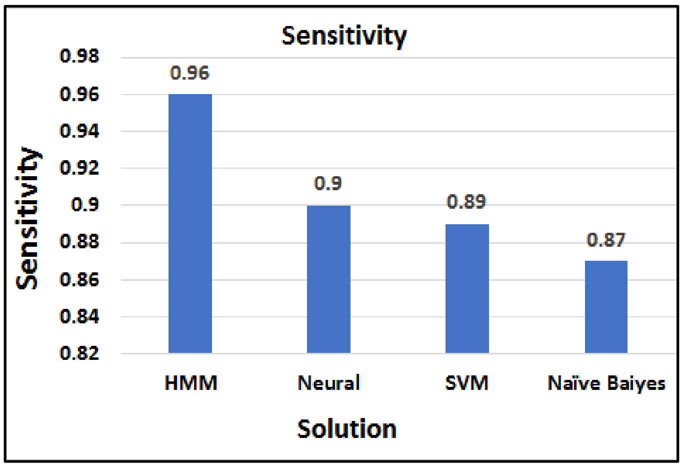
Parameter	**Values**
Kernel	Radial Bias Kernel
Degree	3
Gamma	0.1

The accuracy is measured for proposed solution and compared with machine learning models and the result is given below



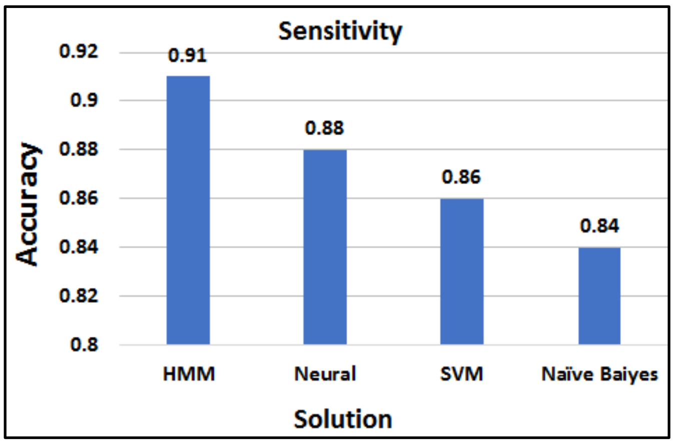
Solution	Accuracy
HMM	0.94
Neural	0.89
SVM	0.87
Naïve Baiyes	0.85

The accuracy in proposed HMM model is comparatively higher than machine learning solutions, because of the way of modelling the relationship between the events while machine learning model use only snap shot information of events.
The sensitivity is measured for proposed solution and compared with machine learning models and the result is given below



Solution	Sensitivity
HMM	0.96
Neural	0.9
SVM	0.89
Naïve Baiyes	0.87

The sensitivity is higher in the proposed HMM as any deviation from normal is identified as vulnerability without doubt.
The specificity is measured for proposed solution and compared with machine learning models and the result is given below



Solution	Sensitivity
HMM	0.91
Neural	0.88
SVM	0.86
Naïve Baiyes	0.84

The specificity measure which is an indicator of capability of system to detect non vulnerability is higher in the proposed HMM solution compared to neural, SVM and Naïve Baiyes.

CONCLUSION

Vulnerability scanner is an important component for FASP solution. The vulnerabilities must be detected with high accuracy to prevent malicious activities and ensure security in Hadoop platform. Different from earlies works of securing only against DOS attack, this work proposes a solution to detect the vulnerabilities defined by CVE on Hadoop platform. The solution is designed in an extensible way, so that it is easy to extend the platform for new kinds of attacks.

REFERENCES

[1] S. M. S and K. C. RAVISHANKAR, "Dynamic Data security for Hadoop Systems using Fuzzy Adaptive Security Profiles (FASP)," 2019, 1st International Conference on Advances in Information Technology (ICAIT), Chikmagalur, India, 2019, pp. 558-565, doi: 10.1109/ICAIT47043.2019.8987332.
[2] D. Liu et al., "Pangr: A Behavior-Based Automatic Vulnerability Detection and Exploitation Framework," 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, 2018, pp. 705-712.
[3] Hameed, Sufian& Ali, Usman. (2018). HADEC: Hadoop-based live DDoS detection framework. EURASIP Journal on Information Security. 2018. 10.1186/s13635-018-0081
[4] Zhen Li and Deqing Zou, "SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities", arXiv, cs. LG,2018
[5] G. Grieco, G. L. Grinblat, L. C. Uzal, S. Rawat, J. Feist, and L. Mounier, "Toward large-scale vulnerability discovery using machine learning," in Proceedings of the 6th ACM on Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 2016, pp. 85–96.
[6] Maglaras, Leandros,Lima Filho, Francisco Sales de,"Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning",Security and Communication Networks,2019.
[7] Daoudhiri, Kaoutar& Najat, Rafalia&Abouchabaka, Jaafar. (2018). Attacks and Countermeasures in a HADOOP Cluster.
[8] Wang, Yongzhi& Shen, Yulong & Wang, Hua & Cao, Jinli& Jiang, Xiaohong. (2016). MtMR: Ensuring MapReduce Computation Integrity with Merkle Tree-based Verifications. IEEE Transactions on Big Data. PP. 1-1. 10.1109/TBDATA.2016.2599928.
[9] Y. Wang, J. Wei and M. Srivatsa, "Result Integrity Check for MapReduce Computation on Hybrid Clouds," 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, 2013, pp. 847-854.
[10] L. Cheng, Q. Shen and C. Dong, "Invader Job: A Kind of Malicious Failure Job on Hadoop YARN," 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, 2018, pp. 1-6.

[11] B Agrawal, T Wiktorski, C Rong, "Analyzing and Predicting Failure in Hadoop Clusters using Distributed Hidden Markov Model", International Conference on Cloud Computing and Big Data in Asia, pp. 232-346, 2015

[12] M. Mizukoshi and M. Munetomo, "Distributed denial of services attack protection system with genetic algorithms on Hadoop cluster computing framework," 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, 2015, pp. 1575-1580.

[13] https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-22215/Apache-Hadoop.html.