A Novel Framework for Cloud based Virtual Machine Security by Change Management using Machine

S.Radharani, V.B.Narasimha
Department of CSE, UCE, Osmania University
Hyderabad, India

***Abstract*—The increased growth in the cloud-based application development and hosting, the demand for higher application and data security is also increasing. The cloud-based applications are hosted on virtual machines and the data generated or used by these applications are also hosted inside the virtual machines. Hence, the security of the applications and the data can be achieved only by securing the virtual machines. There are number of challenges to achieve the security of the virtual machines. Firstly, the size of the virtual machines is large, and the generic cryptographic methods are primarily designed to handle smaller size of the data. Thus, the applicability of these methods for virtual machine are subjected to analysis. Secondly, the additional time required for applying the cryptographic algorithms on the virtual machines impact the response time of the applications, which again impacts the service level agreements. Finally, the virtual machines during the migration are highly vulnerable as the virtual machines are migrated inside the data center networks as simple text data. A good number of research attempts have tried to solve these challenges. Nonetheless, most of the parallel research works have either compromised on the strength of the security protocols or have compromised on the time taken to apply the cryptographic methods. However, the need of the research is to identify the attacks based on the characteristics of connection requests and reduce the time for the encryption and decryption of the virtual machines. This work proposes a novel framework for detection of the attacks based on a machine learning driven algorithm by analyzing the connection properties and prevent the attacks by selective encryption of the virtual machines using another machine learning driven algorithm. This work demonstrates nearly 98% accuracy in detection of the newer and existing attack types.**

***Keywords*—DevOps; deep clustering; VM security; cloud security; VM versioning; progression cryptography**

I. Introduction

The security for the cloud infrastructure has always been a persistent issue as most of the consumers and practitioners do not have clear understanding about the security factors and implementation details. To some extent, the service providers have made a closed loop about the knowledge of cloud security inside the organizations and sometimes only to the selective groups. This makes the deployment of the cloud-based security protocols even harder for the researchers. Nonetheless, the recent research outcomes by various research attempts are opening the closed loops of the knowledge and exploring the possibilities of the deployment of novel and higher performing security protocols. One such work presented by P. Mishra et al. [1]. Nevertheless, the challenges of cloud securities are not only restricted to the data stored on the cloud. Rather, the security challenges can be observed in all the layers of cloud implementations as on the infrastructure layer, platform layer and the services layer. Another work by P. Mishra et al. [2] have confirms this claim. Thus, deploying the security protocol for all the layers of cloud implementation is highly complex due to various aspects such as model complexity or compatibility or interoperability between the layers.

Henceforth, the implementation of the cloud security protocols can be best implemented using the virtual machine architectures. As the virtual machines holds the applications core and the data, generated or consumed by these applications, hence protecting the virtual machines must be the primary concern, which is implemented in this work. This work identifies the challenges of cloud security, internally which is virtual machine security and proposes a machine learning driven framework to protect the VMs.

II. Parallel Research Outcomes

The security of the cloud-based applications is critical as mentioned earlier. The applications and the data on the cloud are visible to authenticated and unauthenticated parties at the same time. Though, the access and identity management aspects of the online access can restrict the privileges on the applications and the data. Nonetheless, the visibility of data cannot be restricted. Hence, the possibilities of the attacks also increase on such data. The work by M. R. Watson et al. [3] have clearly listed the vulnerabilities on the cloud systems and also produced a clear guideline for managing the security. Considering the similar directions, to produce a framework for detection of the attacks based on characteristics, yet another work by V. Varadharajan et al. [4] can be highlighted. These parallel research outcomes are primarily focused on an old framework called Rekall [5] and the produced recent outcomes are the attempts to reduce the complexity and at the same time increasing the responsiveness of the same outlined characteristics. These outcomes have mainly concentrated on the prevention of the attacks.

In the other hand, the domains for attack detections are also very popular among researchers. The work by T. K. Lengyel et al. [6] have clearly listed the possibilities of the attack analysis frameworks to detect the attacks. Nonetheless, these detection processes can be highly complex for the distributed architectures such as cloud or fog or edge-based computing. The application, the data and the userbases are always distributed and most of the times, the execution is parallel. Hence, the protective framework must also comply with the

distributed nature of the architecture. The work by S. Gupta et al. [7] have confirmed to this believe.

The attacks are not only restricted to platform and the service layers. Multiple attacks are also reported on the physical hardware devices. The immediate but costly solution is to provide the hardware security modules or the HSM devices. Nonetheless, as mentioned these solutions are costly and for a cloud-based architecture, the applicability of the HSMs is very limited due to the limited physical access to the infrastructure. The work by D. Kirat et al. [8] have spoken in favor of this statement and confirms the claim. Although, the analysis of the intrusion or attack detections must take place at all the layers of cloud computing and infrastructure layer is not an exception. The work by C. Spensky et al. [9] have elaborated on the possibilities and feasibilities of monitoring for the attack detections on the physical infrastructure layer. This work has been criticized for not considering the possibilities for remote monitoring, which can be achieved using the access to the virtual machines. In the recent times, a good number of virtual machine managers have incorporated the monitoring layers in the VMM structure.

Reciting back to the monitoring of the virtual machines for attack detection and prevention methods have improved a lot using the virtual machine monitoring possibilities. The survey done by F. Cai et al. [10] confirms few claims directly and indirectly as firstly, the deployment of the security features can be best adopted on the virtual machines. Secondly, the existing cryptographic methods can easily be outperformed in the recent higher demand for best response times and finally, the newer types of the attacks are increasing day by day and a method for detecting the attacks based on the behavior must be adopted. Thus, the demand for the automated framework with these features is the demand of the current research as also demonstrated in the work by A. Almtrf et al. [11].

The primary features of the expected framework must comply with few additional characteristics. The first characteristics is the close association with the software and the hardware modules to track the flow of the application processing characteristics as rightly stated in the work by A. Khurshid et al. [12]. The second characteristics of the proposed framework is to track the changing nature of the data as mentioned in the work by N. E. Moussaid et al. [13]. The final characteristics must comply with the deployed virtual machine-based applications hosted on the cloud platforms as suggested by X. Lu et al. [14]. Thus, this work considers all the recommendations from the parallel research outcomes and further produces the proposed framework for detection and prevention of the attacks on the cloud application, in tern the virtual machine security.

Further, this work realizes the characteristic based detection of the attacks. This not only identifies the known attack types, but also identifies the newer attack types. The work by B. Sudhakar et al. [15] has clearly listed the attack types and the mapping to the connection properties. The conclusive mapping from this work is furnished here [Table I].

TABLE I. ATTACK TYPES AND CONNECTION PROPERTIES MAPPING [15]

Attack Type	**Connection Properties**
Browser Based Attacks	1. Count of the connection requests 2. Access Type Requests
Brute Force Based Attacks	1. Count of the connection requests 2. Ratio between the request and responses
DoS Based Attacks	1. Access Type Requests 2. Service Request Types 3. The rate of change in the service request types
SSL Based Attacks	1. Service Request Types 2. The rate of change in the service request types
Scan Based Attacks	1. Ratio between the request and responses
DNS Based Attacks	1. Service Request Types

It is worth the mention, that these all characteristics or connection properties are available in the KDD dataset [16].

Thus, in the next section of this work, the problem identified in this section in the parallel research outcomes is formulated using mathematical models.

III. PROBLEM FORMULATION & PROPOSED SOLUTIONS

After the fundamental understanding of the research problems in the previous section of this work, this section focuses and elaborates the core problems and proposes solutions to these problems using the mathematical modeling techniques.

The first problem elaborates on the responsiveness of the cloud-based applications due to the adaptation of the attack detection methods. The parallel research outcomes, as seen in the previous sections, shows higher time complexity. The increased time complexity is due to the nature of analysis deployed by these algorithms, which primarily focus on large number of characteristics or the connection properties. Hence, this must be resolved.

Lemma – 1: The reduction of the connection characteristics using the correlation method can reduce the time complexity of the detection method.

Proof: The connections characteristics or the properties extracted from the connection requests can be a very large dataset because of multiple monitoring system. Many of the times, these large datasets provide limited and redundant information, which is again at the cost of higher time complexity. Thus, a machine learning driven process to reduce the number of characteristics can certainly reduce the time complexity.

Assuming that, the set of connection properties, C[], is a collection of multiple characteristics and each characteristics can be identified as C_i. Thus, for n number of total characteristics, the relationship can be formulated as:

$$C[] = \langle C_1, C_2, C_3 ...., C_n \rangle \quad (1)$$

Also, assuming that, C_X is the class variable, which defines the nature of the connection in terms of attacks or normal from the historical information sets.

Hence, the characteristics analysis for detection of the attacks using the standard algorithms can be formulated as,

$$TH_i = \Phi(\exists C_i) : \prod_{RowID=C_i} C[] \tag{2}$$

Here, Φ is the function for extracting the threshold and further, the threshold for attribute C_i is stored in TH_i. Clearly, the threshold must be calculated relatively as with the consideration of the other parameters.

Further, the combined information from the thresholds from all the characteristics can decide the nature of the connection in terms of the class variable as,

$$C_X = \sum_{i=1}^{n} TH_i \tag{3}$$

It is natural to realize that due to Eq. 2 and Eq. 3, the time complexity, T_1, can be formulated as,

$$T_1 = n(n-1) \tag{4}$$

Or,

$$T_1 = n * n = O(n^2) \tag{5}$$

For a large dataset with 100s of parameters or characteristics, this time complexity for detection of the attacks can be very high. Thus, this problem must be solved using parameter reduction process.

Thus, based on the Eq. 1, the correlation formulation can be formulated as,

$$\rho(C_x, C_i) = \frac{(\eta(C[x]) - C_x).(\eta(C[i]) - C_i)}{\sigma C_x . \sigma C_i} \tag{6}$$

Here, ρ defines the correlation value or correlation coefficient, η defines the mean value and σ defines the standard deviation.

The standard deviation calculation can be formulated as,

$$\sigma C_i = \sqrt{\frac{\sum \{C_i - \eta(C[i])\}^2}{n}} \tag{7}$$

Further, the total correlation sets can be stored in Corr[] and the highest values can be taken to identify m number of characteristics for final analysis as,

$$Corr[] =< \exists \rho(C_x, C_i) > \tag{8}$$

And,

$$m :\rightarrow Corr[] \tag{9}$$

Thus, in the light of Eq. 4, the new time complexity, T_2, can be formulated as,

$$T_2 = m(m-1) \tag{10}$$

Or,

$$T_2 = m * m = O(m^2) \tag{11}$$

As, $m << n$, thus it is conclusive to state that

$$T_2 << T_1 \tag{12}$$

Thus, reduction of the time complexity using the attribute reduction method is highly feasible.

The second problem elaborates on the detection of the attack types. The attack types can be identified using a cluster analysis on the connection characteristics or the properties. As seen in the previous section of the work, the parallel research outcomes mostly fail to detect the newer attacks, though the types of the attacks are not very new and have a strong similarity with the existing and known types of attacks.

This problem can be solved using deep cluster technique. The clustering method for identification of the attacks is significant as the identification of attacks direct towards anomalies in the connection, which is easily identifiable as outliers using the clustering method.

Lemma – 2: The deep clustering method can identify the newer types of attacks using the outlier identification method.

Proof: The outliers as a result of clustering process identifies the anomalies using various characteristics and similarities of the characteristics domain values. Based on the nature of the data used in the clustering process, the outliers can define various meanings. As in this research the data used are the connection characteristics, hence the outliers will denote the abnormal connections or the attacks.

Continuing and revising the Eq. 1, for all the characteristics, there must be domains for each characteristic as,

$$C[][] = \langle C_1[], C_2[], C_3[]...., C_n[] \rangle \tag{13}$$

Further, the clustering process must be performed initially for each and every characteristic or attribute domains as

$$CL_i[] \leftarrow \Phi(\exists C_i[]) \tag{14}$$

Here, the set of clusters for the i^{th} attribute will be stored in $CL_i[]$ and Φ denotes the clustering process.

Henceforth, the number of members in each cluster must be validated and the cluster with the lowest number of members are the potential clusters, inside which the outliers will reside.

Thus, the iterative clustering must be performed until the outliers, in this case the attacks, is not identified as,

$$\omega \leftarrow \Phi(\exists \left| CL_i[] \right|_{low}) \quad (15)$$

The terminating condition for Eq. 15 iteration is $\omega \to 1$.

Henceforth, it can be stated conclusively, the minute deviations can be identified using this proposed method and further any new attack can also be detected, which has very little similarity to the existing attack types.

The final problem, which this research aims to solve is the reduction of the cryptographic algorithm implementation time. As seen in the previous section of this work, the cryptographic algorithms are not designed to handle the large data, which is case of virtual machine files are very large in volume. Also, due to the higher adaptability of the DevOps processes across all organizations for application development, the changes made to the application and indirectly to the virtual machines are very high. This makes the process of applying cryptographic algorithms further difficult.

Henceforth, the solution is to track the changes made to the virtual machines in terms of application code and data and apply incremental encryption process to reduce the time.

The proposed solutions are converted to algorithms, which are furnished in the next section of this work.

IV. Proposed Algorithms and Framework

After the formulation of the concepts of solutions in the previous section, in this section of the work, the proposed algorithms and the proposed frameworks are furnished.

Firstly, the Connection Characteristics Reduction using Correlation Analysis algorithm is furnished.

Algorithm - I: Connection Characteristics Reduction using Correlation Analysis (**CCR-CA**) Algorithm

Input:
Connection Characteristics set as CS[]

Output:
Reduced Characteristics set as RCS[]

Process:
Step - 1. Load the CS[] set
Step - 2. Mark the class characteristics as CX from CS[x]
Step - 3. For each attrbiute in CS[] as CS[i]
- a. Calculate the standard deviation, as SD[] using Eq. 7
- b. Calculate the correlation of CS[i] with CX as Corr[i] using Eq. 6

Step - 4. For each element in Corr[] as Corr[j]
- a. If Corr[j] Not Equal Corr[j+1] & Corr[j] is Max
  - i. Store RCS[j] = CS[i]
- b. Else,
  - i. Continue
- c. Corr[j] = Null
- d. Stop if Count(RCS[]) >= Count(CS[])/2

Step - 5. Return RCS[]

The above algorithm is framed to solve the first problem discussed and based on the proposed Lemma – 1.

Secondly, the Deep Clustering Based Attack Detection algorithm is furnished.

Algorithm - II: Deep Clustering Based Attack Detection (**DC-AD**) Algorithm

Input:
Reduced Characteristics set as RCS[][]
Output:
Detected Attacks as DS[]

Process:
Step - 1. Load the RCS[][] set
Step - 2. For each element in RCS[][] as RCS[i][]
- a. Apply K-Means Clustering on RCS[i][] and store the result in CL[i][] using Eq. 14

Step - 3. For each element in CL[][] as CL[j][]
- a. If count(CL[j][]) -> min(count(CL[][])
  - i. Apply K-Means Clustering on CL[j][] and store the result in CL1[i][] using Eq. 15
  - ii. Repeat the process untill Count(CL1[i][]) -> 1
  - iii. Identify the attack characteristics as DS[k] = RCS[i]
- b. Else,
  - i. Continue

Step - 4. Return DS[]

The above algorithm is framed to solve the second problem discussed and based on the proposed Lemma – 2.

Thirdly, the Random Crypto Key Generation algorithm is furnished.

Algorithm - III: Random Crypto Key Generation (**RCKG**) Algorithm

Input:
Large Random numbers as P & Q

Output:
I. Public Key as PK
II. Private Key as PKK

Process:
Step - 1. Calculate the modulus, M as M = P * Q
Step - 2. Select the derived encryption factor, DE as DE>1 and DE<(P-1).(Q-1)
Step - 3. Generate public key, PK as PK = (M,DE)
Step - 4. Generate private key, PKK as PKK = {1 MOD (P-1).(Q-1)}/DE
Step - 5. Return PK and PKK

Fourthly, the Progressive Virtual Machine Encryption using Change Detection algorithm is furnished.

Algorithm - IV: Progressive Virtual Machine Encryption using Change Detection (**PVME-CD**) Algorithm

Input:

I. Version Management of VM as VMS[]
II. Public Key as PK (M, DE)

Output:

Encrypted Virtual Machine as VME

Process:

Step - 1. Load the virtual machine versions as VMS[]
Step - 2. For each element in VMS[] as VMS[i]
  a. Configuration Management:
    i. Identify the import and include statements
    ii. Store the configuration management as CM[i]
  b. Data Management:
    i. Identify the variable in the code
    ii. Store the data management as DM[i]
  c. Life Cycle Management:
    i. Identify the loops and conditional statements
    ii. Store the life cycle management as LCM[i]
Step - 3. For each element in VMS[] as VMS[i]
  a. If CM[i] Not Equals to CM[i+1]
  b. Then, Store the changes CMC[j] = CM[i]-CM[i+1]
  c. If DM[i] Not Equals to DM[i+1]
  d. Then, Store the changes DMC[j] = DM[i]-DM[i+1]
  e. If LCM[i] Not Equals to LCM[i+1]
  f. Then, Store the changes LCMC[j] = LCM[i]-LCM[i+1]
  g. Merge the changed components as CC[i] = CMC[j] U DMC[j] U LCMC[j]
  h. Build the encrypted VMS[i] as VME = pow(CC[i],DE) mod M
Step - 4. Return VME

Fifthly & finally, the Progressive Virtual Machine Decryption using Change Detection algorithm is furnished.

Algorithm - V: Progressive Virtual Machine Decryption using Change Detection (**PVMD-CD**) Algorithm

Input:

I. Encrypted Virtual Machine as VME
II. Private Key as PKK (DE, M)

Output:

Decrypted Virtual Machine as VM

Process:

Step - 1. Load the encrypted virtual machine as VME
Step - 2. Build the decrypted virtual machine, VM as VM = pow(VME,DE) mod M
Step - 3. Return VM

The above algorithms are framed to solve the third problem discussed in the previous section of this work.

Further, the final framework is furnished here [Fig. 1]:

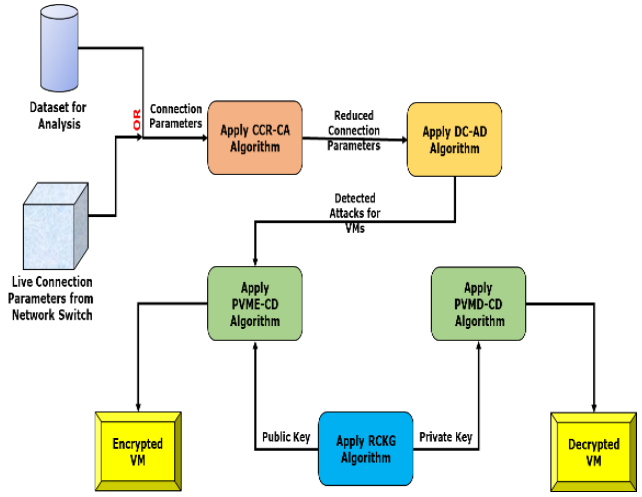


Fig. 1. A Framework for Cloud based Virtual Machine Security by Change Management using Machine Learning.

Further, in the next section of this work, the obtained results from these proposed algorithms are discussed.

V. Result and Discussion

After the detailed understanding on the proposed algorithms, here the obtained results are furnished.

Firstly, the used dataset [16] is analyzed here [Table II].

Further, the data is visualized graphically here [Fig. 2].

Here this is important to observe that, the many attributes have higher unique distributions and further demonstrates unique characteristics to detect large number of attacks.

Secondly, the impact or the correlation analysis results are furnished here [Table III].

The obtained results are again visualized graphically [Fig. 3].

Here it is natural to realize that the many of the attributes have demonstrated higher correlation than the other attributes. As per the proposed algorithm, the threshold of the correlation is calculated as 0.223 and based on the correlation theory, the positive impacted and meaning full attributes correlation must be above 0.50. Thus, again based on the proposed algorithm, the median value of the correlation is considered as 0.135.

Henceforth, based on the new correlation threshold, the following attributes are identified in the reduced set [Table IV].

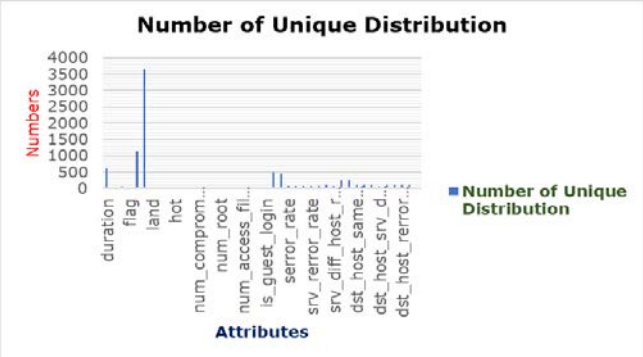
Further, the reduced set is also analyzed graphically here [Fig. 4].

Here, it is worth noting that, due to this process the information loss is minimum as the diversified nature of the dataset with high distribution is kept intact.

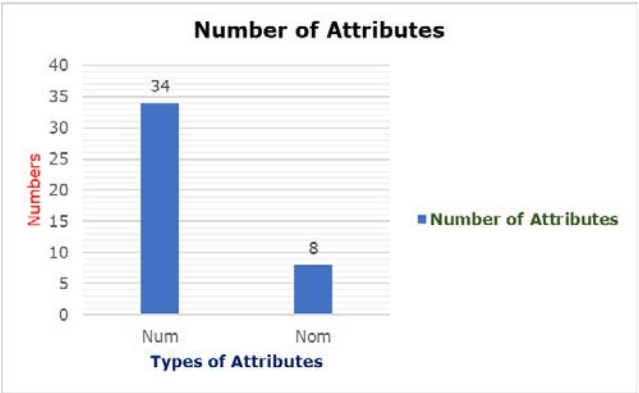
Further, the results from the deep clustering process to detect the attacks are furnished here [Table V].

TABLE II. DATASET ANALYSIS

SNO	Attribute Name	Attribute Type	Missing Value (%)	Number of Unique Distribution
1	“duration”	Num	0%	624
2	“protocol_type”	Nom	0%	3
3	“service”	Nom	0%	64
4	“flag”	Nom	0%	11
5	“src_bytes”	Num	0%	1149
6	“dst_bytes”	Num	0%	3650
7	“land”	Nom	0%	2
8	“wrong_fragment”	Num	0%	3
9	“urgent”	Num	0%	4
10	“hot”	Num	0%	16
11	“num_failed_logins”	Num	0%	5
12	“logged_in”	Nom	0%	2
13	“num_compromised”	Num	0%	23
14	“root_shell”	Num	0%	2
15	“su_attempted”	Num	0%	3
16	“num_root”	Num	0%	20
17	“num_file_creations”	Num	0%	9
18	“num_shells”	Num	0%	4
19	“num_access_files”	Num	0%	5
20	“num_outbound_cmds”	Num	0%	1
21	“is_host_login”	Nom	0%	2
22	“is_guest_login”	Nom	0%	2
23	“count”	Num	0%	495
24	“srv_count”	Num	0%	457
25	“serror_rate”	Num	0%	88
26	“srv_serror_rate”	Num	0%	82
27	“rerror_rate”	Num	0%	90
28	“srv_rerror_rate”	Num	0%	93
29	“same_srv_rate”	Num	0%	75
30	“diff_srv_rate”	Num	0%	99
31	“srv_diff_host_rate”	Num	0%	84
32	“dst_host_count”	Num	0%	256
33	“dst_host_srv_count”	Num	0%	256
34	“dst_host_same_srv_rate”	Num	0%	101
35	“dst_host_diff_srv_rate”	Num	0%	101
36	“dst_host_same_src_port_ra ”	Num	0%	101
37	“dst_host_srv_diff_host_ra”	Num	0%	58
38	“dst_host_serror_rate”	Num	0%	99
39	“dst_host_srv_serror_rate”	Num	0%	101
40	“dst_host_rerror_rate”	Num	0%	101
41	“dst_host_srv_rerror_rate”	Num	0%	100
42	“class”	Nom	0%	2



(a)



(b)

Fig. 2. (a) and (b) Analysis of the Dataset.

TABLE III. CORRELATION ANALYSIS

SNO	Correlation with “Class” Variable
1	0.150
2	0.112
3	0.368
4	0.525
5	0.016
6	0.097
7	0.008
8	0.039
9	0.009
10	0.057
11	0.135
12	0.618
13	0.021
14	0.018
15	0.022
16	0.021
17	0.016
18	0.052
19	0.070
20	0.000
21	0.010
22	0.116
23	0.353
24	0.092
25	0.282
26	0.280
27	0.517
28	0.513
29	0.550
30	0.261
31	0.192
32	0.399
33	0.645
34	0.636
35	0.276
36	0.030
37	0.022
38	0.312
39	0.308
40	0.528
41	0.506

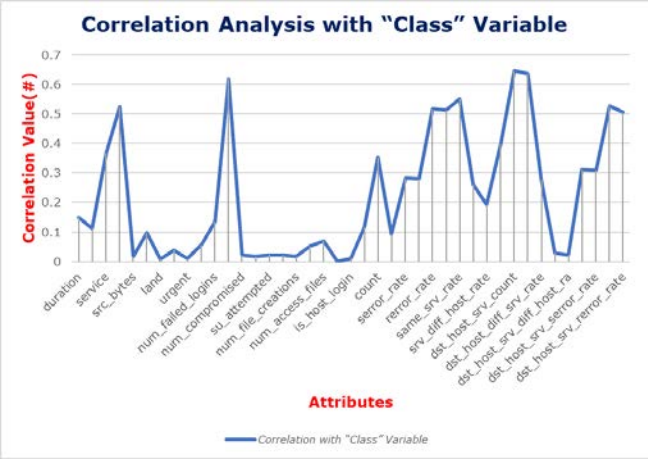


Fig. 3. Correlation Analysis.

TABLE IV. REDUCED ATTRIBUTE SET WITH CORRELATION

SNO	Attribute Name	Correlation with "Class" Variable
1	"dst_host_srv_count"	0.645
2	"dst_host_same_srv_rate"	0.636
3	"logged_in"	0.618
4	"same_srv_rate"	0.550
5	"dst_host_rerror_rate"	0.528
6	"flag"	0.525
7	"rerror_rate"	0.517
8	"srv_rerror_rate"	0.513
9	"dst_host_srv_rerror_rate"	0.506
10	"dst_host_count"	0.399
11	"service"	0.368
12	"count"	0.353
13	"dst_host_serror_rate"	0.312
14	"dst_host_srv_serror_rate"	0.308
15	"serror_rate"	0.282
16	"srv_serror_rate"	0.28
17	"dst_host_diff_srv_rate"	0.276
18	"diff_srv_rate"	0.261
19	"srv_diff_host_rate"	0.192
20	"duration"	0.150
21	"num_failed_logins"	0.135

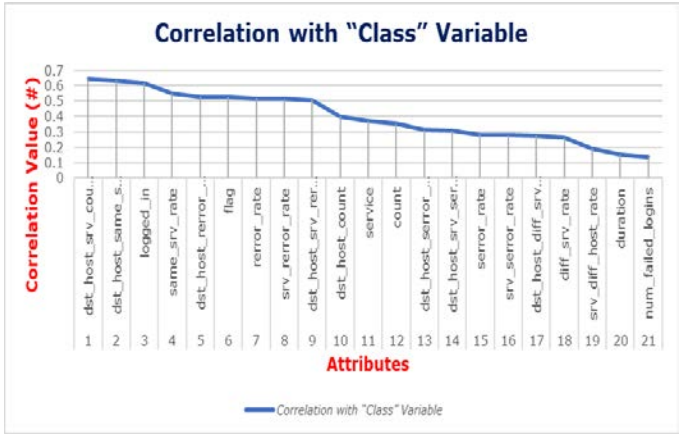


Fig. 4. Reduced Attribute Set Correlation Analysis.

TABLE V. ATTACK DETECTION ACCURACY ANALYSIS

Analysis Metric	Number of Values	Percentage (%)
"Correctly Classified Instances"	84248	98.2335
"Incorrectly Classified Instances"	1515	1.7665
"Kappa statistic"	0.9638	-
"Mean absolute error"	0.032	-
"Root mean squared error"	0.12	-
"Relative absolute error"	-	6.5494
"Root relative squared error"	-	24.2856
"Total Number of Instances"	85763	-

The results are observed visually here [Fig. 5].

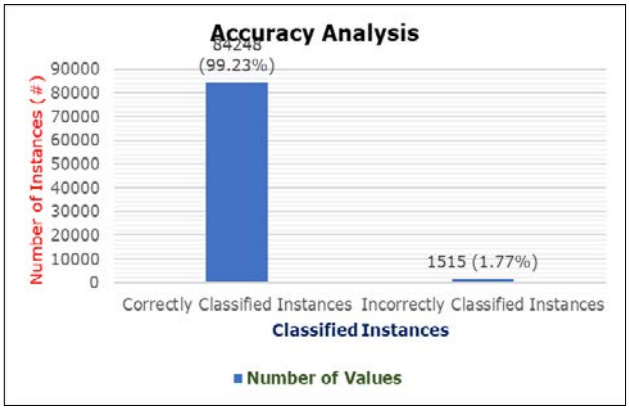


Fig. 5. Detection Accuracy Analysis.

Clearly from the results it is worth noting that the accuracy of the proposed deep clustering algorithm for attack detection is 99.23% with the newer types of attacks.

Further, the change detection algorithm for the virtual machines produces a log of tracked changes. The analysis is performed over 1000 virtual machines, however, for the visualization on 10 virtual machine logs are presented here.

Sample Change Detection Log File
Change Tracking for VM #1 Tracking for version #1 VM Size reduced by 126 GB
Change Tracking for VM #2 Tracking for version #1 VM Size reduced by 95 GB
Change Tracking for VM #3 Tracking for version #1 VM Size reduced by 94 GB Tracking for version #2 VM Size reduced by 42 GB
Change Tracking for VM #4 Tracking for version #1 VM Size increased by 138 GB
Change Tracking for VM #5 Tracking for version #1 VM Size reduced by 183 GB Tracking for version #2 VM Size increased by 28 GB
Change Tracking for VM #6 Tracking for version #1 VM Size increased by 236 GB Tracking for version #2 VM Size reduced by 28 GB
Change Tracking for VM #7 Tracking for version #1 VM Size increased by 208 GB
Change Tracking for VM #8 Tracking for version #1 VM Size increased by 275 GB
Change Tracking for VM #9 Tracking for version #1 VM Size reduced by 32 GB Tracking for version #2 VM Size reduced by 42 GB Tracking for version #3 VM Size increased by 79 GB
Change Tracking for VM #10 No Changes Detected

From the above sample log file, the following aspects are conclusive regarding the virtual machine change detection algorithm:

1) The changes for any virtual machine can be detected over multiple versions of the same VM.

2) The changes are reflected in terms of size; however, the actual change management is tracked based on characteristics of the virtual machines.

3) The detection algorithm also ensures no changes if the version of the same virtual machine is not updated.

Henceforth, it is conclusive that, the change management algorithm is perfectly justifying the claims made is this work.

Further, the key generation algorithm outputs are analysed here [Table VI]. During the testing phase, the algorithm is tested for more than 1000 instances. However, for representation purposes only 10 examples from the total outcomes are furnished.

TABLE VI. KEY GENERATION TIME ANALYSIS

Test Sequence #	**Key Generation time (ns)**
Seq #1	7
Seq #2	9
Seq #3	14
Seq #4	7
Seq #5	10
Seq #6	20
Seq #7	15
Seq #8	25
Seq #9	10
Seq #10	14

The results are visualized graphically here [Fig. 6].

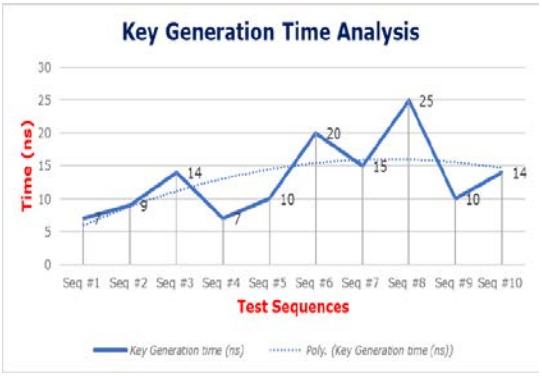


Fig. 6. Key Generation Time Analysis.

It is evident from the above results, that the time taken for the key generation demonstrates fairly liner characteristics, which is always expected for any best key generation algorithms.

TABLE VII. CRYPTOGRAPHIC ALGORITHMS TIME ANALYSIS

Test Sequence #	**Encryption Time (ns)**	**Decryption Time (ns)**
Seq #1	19	22
Seq #2	6	16
Seq #3	4	19
Seq #4	10	7
Seq #5	17	1
Seq #6	11	20
Seq #7	18	14
Seq #8	14	20
Seq #9	8	15
Seq #10	12	16

Further, the encryption and decryption time analysis is furnished here [Table VII]. During the testing phase, the algorithm is tested for more than 1000 instances. However, for representation purposes only 10 examples from the total outcomes are furnished.

The results are also visualized graphically here [Fig. 7].

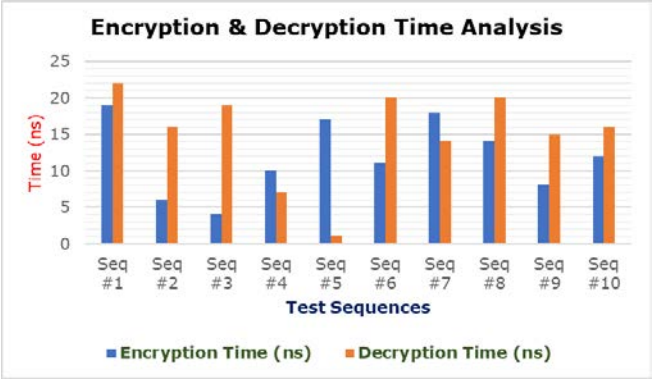


Fig. 7. Encryption and Decryption Time Analysis.

The results obtained in terms of time taken to perform the encryption and decryption operations on the detected changes on the virtual machines clearly showcase a trend of reduced time. This reduction is achieved due to the change management algorithm deployed for the virtual machine versions.

Further, in the next section of this work, the obtained results are compared with the other parallel research outcomes.

VI. Comparative Analysis

The obtained result from the proposed framework is highly satisfactory. Nonetheless, without a comparative analysis, no work can be concluded as benchmarked outcome. Thus, in this section of the work, the proposed framework using various parameters is compared with the parallel popular research outcomes [Table VIII].

Henceforth, it is conclusive to state that, the proposed framework has outperformed the parallel popular research works in terms of capabilities and as well as in terms of model complexity.

Finally, in the next section of the work, the research conclusion is presented.

TABLE VIII. Comparative Analysis

Author, Year	Methodology	Capabilities	Model Complexity
X. Lu et al. [14], 2020	Machine Learning	Reactive Security	$O(n^2)$
N. E. Moussaid et al. [13], 2020	Machine Learning	Reactive Security	$O(n^2)$
F. Cai et al. [10], 2019	Machine Learning	Reactive Security	$O(n^2)$
B. Sudhakar et al. [15], 2019	Machine Learning	Reactive Security	O(n*m)
Proposed Framework	Machine Learning	Reactive & Proactive Security	O(n)

VII. Conclusion

This research establishes benchmark in many aspects. In any of the parallel research outcomes, the reduction of time for applying the cryptographic aspects is ignored, which as per this work is most evident to increase the responsiveness of the cloud security. Also, this work elaborates the possibilities of detection of the attacks with the simplest model with least complexity. The proposed mathematical models and algorithms are strong evidence of the claim that, this framework is not only capable of detection of existing or known attacks, rather, this framework can also detect newer or unknown types of attacks based on the connection characteristics analysis. The detection rate on the benchmarked dataset is over 98%, which is again a benchmark for these types of framework.

References

[1] P. Mishra et al., "Intrusion detection techniques in cloud environment: A survey", J. Netw. Comput. Appl., vol. 77, pp. 18-47, 2017.

[2] P. Mishra et al., "VAED: VMI-assisted evasion detection approach for infrastructure as a service cloud", Concurrency Comput.: Practice Experience, vol. 29, 2017.

[3] M. R. Watson et al., "Malware detection in cloud computing infrastructures", IEEE Trans. Depend. Sec. Comput., vol. 13, no. 2, pp. 192-205, Mar./Apr. 2016.

[4] V. Varadharajan and U. Tupakula, "On the design and implementation of an integrated security architecture for cloud with improved resilience", IEEE Trans. Cloud Comput., vol. 5, no. 3, pp. 1-14, Jul.–Sep. 2017.

[5] Rekall: Memory FOrensics and Analysis Framework, May 2014, [online] Available: http://www.rekall-forensic.com/.

[6] T. K. Lengyel, Stealthy Monitoring with Xen Altp2m, 2016, [online] Available: https://blog.xenproject.org/2016/04/13/stealthy-monitoring-with-xen-altp2m/# comments.

[7] S. Gupta and P. Kumar, "System cum program-wide lightweight malicious program execution detection scheme for cloud", Inf. Secur. J.: A Global Perspective, vol. 23, no. 3, pp. 86-99, 2014.

[8] D. Kirat et al., "BareCloud: Bare-metal analysis-based evasive malware detection", Proc. 23rd USENIX Secur. Symp., pp. 287-301, 2014.

[9] C. Spensky, H. Hu and K. Leach, "LO-PHI: Low-observable physical host instrumentation for malware analysis", Proc. Netw. Distrib. Syst. Secur. Symp., pp. 1-15, 2016.

[10] F. Cai, N. Zhu, J. He, P. Mu, W. Li and Y. Yu, "Survey of access control models and technologies for cloud computing", Cluster Comput., vol. 22, no. S3, pp. 6111-6122, May 2019.

[11] A. Almtrf, Y. Alagrash and M. Zohdy, "Framework modeling for user privacy in cloud computing", Proc. IEEE 9th Annu. Comput. Commun. Workshop Conf. (CCWC), pp. 0819-0826, Jan. 2019.

[12] A. Khurshid, A. N. Khan, F. G. Khan, M. Ali, J. Shuja and A. U. R. Khan, "Secure-CamFlow: A device-oriented security model to assist information flow control systems in cloud environments for IoTs", Concurrency Comput. Pract. Exper., vol. 31, no. 8, Apr. 2019.

[13] N. E. Moussaid and M. E. Azhari, "Enhance the security properties and information flow control", Int. J. Electron. Bus., vol. 15, no. 3, pp. 249-274, 2020.

[14] X. Lu, L. Cao and X. Du, "Dynamic control method for tenants' sensitive information flow based on virtual boundary recognition", IEEE Access, vol. 8, pp. 162548-162568, 2020.

[15] B. Sudhakar, V. B. Narsimha, G. Narsimaha, Detection of Intrusion using Hybrid Feature Selection and Flexible Rule Based Machine Learning, International Journal of Engineering and Advanced Technology (IJEAT), 2019.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.