Threat-Path Estimate-Based Watchword-Chunk Algorithm for Advanced Persistent Threat in the Cloud

Babu Pandipati, Bharathiyar University, India*
Praveen Sam R., G. Pulla Reddy Engineering College, India

ABSTRACT

In cloud computing, an advanced persistent threat (APT) is a cyber-attack that gains access to a network and remains undetected for some time. APTs have proven difficult to detect and protect. In the existing system, they fail to analyze the path of an outbreak when the monitor assigns a weight to the nodes. If a path for an outbreak is detected, the VM is migrated to hosts that do not account for the overloaded problem and underutilized hosts. In addition to the size of resources occupied by the VM, the traffic was increased. This paper proposes the threat-path reckon technique that detects the multiple paths through re-identification and the addition of automatic weight for its neighbor nodes. Based on that weighted path, the secured object emigration technique invokes a mapping function to migrate the VMs. Finally, the data in the VM are stored in a best-fit distribution. Thus, it provides security but achieves the search overheads.

KEYWORDS

Advanced Persistent Threat (APT), Cloud Computing (CC), Secured Object Emigration (SOE) Technique, Threat-Path Reckon Technique (TPR), Watchword-Chunk Algorithm

INTRODUCTION

The recent boom in the development of Cloud Computing (CC) technology has caused a similar boom in hacker attack methods which is being constantly updated. Industries as well as government, now face more serious threats to information security (Vines, Krutz, (2010)). The threat to information from advanced persistent threat (APT) is much greater than that from independent hackers and poses an enormous challenge to network information security systems. Among the important characteristics of APT is that it is advanced and intrusion is at a very high level. It also has a strong shielding ability and the attack path is often indiscernible and this makes it more difficult for traditional methods to detect and put up a defense (Sabahi, (2011)). It is also persistent, the attack is continuous and of long duration, this also makes it difficult for single, point-based detection techniques to handle. Although APT’s carrier exists in big data, it brings a series of difficulties to APT detection and protection, but it can also use big data to test and respond to APT (Che, Duan, et al., (2011)). If there is comprehensive information data at all levels and stages, and any interactive behavior is detected, different data can be used to find different stages for APT analysis.
APT is a major attack model that goes on for a long time, involves a large amount of data traffic, and is multi-faceted (Shaikh, Haider, (2011)). This mode of attack presents major hurdles to which traditional single-point feature matching detection can hardly put up serious and effective resistance. In the traditional data storage model, the enterprise needs to build and manage a server, storage modules, communication devices, and human resources. On the contrary, multi-cloud storage provides customers with more flexibility, scalability, and convenience in low-speed data storage, based on memory requirements, rather than on how to create and maintain storage. It is, therefore, necessary to develop an efficient method for secure data sharing in multi-cloud storage (Bisong, Rahman, (2011)).

One of the best solutions to reduce data risks in multi-cloud storage is the parallel use of many clouds. Several researchers have developed several models using multi-cloud storage services. They differ in file cutting and distribution and cryptographic techniques used. On the other hand, the TPA might be dishonest and turn into an adversary (Heiser, Nicolett, (2008)). As a result, the malicious position of the TPA may cause significant harm by sharing the sensitive and private data of the CC with unauthorized parties and altering the privacy of the CC in order to obtain financial gains and other benefits (Paquette, Jaeger, et al., (2010)). Therefore, to preserve the privacy of the CC, we need to be able to detect whether the TPA plays a dishonest role while conducting the audit of confidential information and private data of the CC. Several existing guidelines are available in the literature to keep the TPA’s potential malicious activities reviewed and controlled (Agarwal, Agarwal, (2011)). Similarly, an analytical model-based approach for quality evaluation of infrastructure-as-a-service cloud is presented in (Xia, Yunni, et al., (2013)). In those existing protocols, however, computational and communication overheads are not addressed in full, leaving room for possible malicious activities from the TPA. Therefore, a lightweight protocol is required to protect the privacy of the CC and to detect the malicious activities of the TPA, if any (Chou, (2013)).

Virtualization is a key cloud computing technology that can turn a host into multiple virtual hosts that have different resources from computer systems that can support applications. Virtualization has many benefits (Kandukuri, Rakshit, (2009)). Reducing the number of physical hosts by turning the physical host into a virtual host reduces energy consumption and achieves energy efficiency. Virtualization is moreover a cost-effective technology. Load balancing, however, is a challenge. Suppose users have some requirements to run their applications including CPU, memory, and hard disk (Tan, Ai, (2011)). Then, in the data center, configure the corresponding VM to host according to the user needs. There will be overloaded hosts and underused hosts after the user configures the virtual machine on the host. Host resources are limited, such as the CPU, memory, and hard disk. If too many VMs running on one host the host can get overloaded and cause exceptions. This problem can be solved through virtual machine migration (Gul, ur Rehman, et al., (2011)). In this way, (Xia, YunNi, et al., (2015)) proposed a stochastics-queueing-network-based approach for enabling migration in the cloud environment.

Traditional defense systems were not very effective in defending against APTs because APTs exhibit features that are hard to detect. APT attacks mimic a system’s normal behavior in terms of both the system calls generated and its network traffic (Mishra, Mathur, et al., (2013)). In addition, APTs defeat defense systems using polymorphic software code and bypass firewalls on the perimeter using standard protocols and approved ports (Hussain, Abdulsalam, (2011)). While the use of polymorphic or encrypted code in malware has avoided signature-based detection techniques, it is still difficult to detect APTs due to their unpredictable non-repetitive behavior. And unlike traditional, automated malware, APTs only compromise a few hosts in a target network that could include hundreds or thousands of hosts (Sangroya, Kumar, et al., (2010)). Such an enormous number of hosts generate huge traffic which inadvertently makes it difficult to detect because the noise generated by APT activities is very small. All of these APT features make them stealthy and thus require a complex detection strategy to catch the aforementioned (Mather, Kumaraswamy, et al., (2009)). Despite this, the identification and study of APTs have drawn tremendous interest from both industry and academia (Kouyama, Taguchi, et al., (2019)). Nevertheless, the dynamic growth of the APT attack network (APT-AN), which evolves as the attack progresses, is an attribute rarely considered in studies of APT attacks.
Therefore from the above discussion, it is necessary to develop an enhanced mechanism to discourse APT outbreaks in the cloud computing field. Thus to tackle the abovementioned issues in path outbreak analysis, overloaded, authentication, and secure storage, this paper proposed a novel technique, which can mitigate the APT in the cloud using a supreme method. Thus, the remainder of the paper is prepared as follows: section 2 describes the papers deals with the security and attacks in a cloud environment; our contribution over the paper i.e., proposed work has deliberate in section 3; section 4 follows the result and the output of our proposed work; finally, the overall work is concluded in section 5.

LITERATURE REVIEW

Li et al (Li, Yang, (2019)) addressed the dynamic cloud storage recovery (DCSR) problem using differential game theory. Initially, the benefit of the APT attacker and the total loss of the cloud defender was measured by introducing an expected State evolution model capturing the expected state evolution process of the CSS under a combination of attack strategy and recovery strategy. Secondly, we derive a necessary condition for the DCSR problem’s Nash equilibrium and thus introduce the idea of a competitive strategy profile.

Liu et al (Liu, Chung-Hsin, et al., (2019)) proposed a detection mechanism, for the early detection of APT threat using Big Data and Splunk analysis. Because once an attack is successful, then the system’s timely detection is of paramount importance to mitigate its impact and will prohibit APTs from further spreading. Then using data mining techniques to find malicious IP positions.

Neupane et al (Neupane, Neely, et al., (2019).) proposed a novel defense system named Dolus to mitigate the impact of targeted attacks against high-value services hosted on cloud platforms based on SDI. This Dolus framework will implement a ‘pretense’ flexibly and collaboratively to deter the attacker based on the knowledge of the threat obtained from an analysis of the attack function. Using foundations from pretense theory in child play, Dolus takes advantage of elastic capacity provisioning via quarantine virtual machines’ and SDxI policy coordination across multiple network domains to deceive the attacker by creating a false sense of success.

Kumar et al (Kumar, Kumar, (2020)) created a VM allocation for each physical machine; a virtual machine for cloud-based VM migration and a B&B-based approach for virtual server-based multidimensional estimated VM variable assignment. These results are formulated and analyze the different allocation techniques, such as the first-fit approach, best-fit technique, and modified technique, and they have proposed an approach for better cloud computing environment VM migration. After, an energy-efficient VM migration procedure is presented to lessen energy utilization in the cloud environment.

Tandel et al (Tandel, Parmar, et al., (2019)) analyzed the conditions for the deterioration of performance and the effect of the VM migration on performance and energy loss. Servers host the VMs at the Data Centres. The increase in jobs at VMs leads to overhead at the servers which brings the need for the redistribution of VMs to other servers. The relocation of VM has to be done in such a way that the performance is not affected and at the same time the energy consumption should be kept at a minimum level.

Therefore, from the above discussion (18) No mechanism for the relocation of VM (19) absence of attack path detection (20) no effect of the neighborhood (21) does not consider overloaded and underutilized hosts (22) inefficient secure storage. Subsequently, to overwhelm those major issues in the emerging field of cloud computing and offer greater more strong security this paper efficiently proposed the optimal APT’s detection technique.

OPTIMAL PLACEMENT OF CLOUD UNITS TO MITIGATE THE APT’S

New models of network attack and hacking are constantly changing to keep pace with rapid growth in network technology. Advanced Persistent Threat (APT) is a complex, targeted method of attack, typically coordinated by a group of hackers.

A long period of strategic planning and information-seeking typically precedes an attack on a given goal. Focus is on a single entity, using highly specific techniques to launch the attack and
collect confidential information. Advanced Persistent Threat (APT) occurs in the cloud environment which causes huge losses to governments and other organizations. Rather than detecting the cloud resources threatened by these attacks, it is the best way to secure the cloud resources to mitigate the loss. Current techniques track and assign a weight to the nodes and find solutions to move those nodes while failing to determine the route of the outbreak. In addition, finding the fastest path to an outbreak along with multiple paths to a target node is also important. Once a route for an outbreak is found, the VM is relocated to hosts that receive less weight that does not compensate for the overloaded and underused hosts in addition to the size of the VM’s occupied resources, thereby increasing the traffic here. Also, APT targets the data in these VMs and is intended to protect them productively. Encryption and third-party auditing techniques include security data to be kept in transit but also the data to be assured at questionnaires’. Figure 1 illustrates the security architecture of cloud computing based on the front end and back end levels.

Therefore this work proposes a Threat-Path Reckon technique that analyzes and detects the multiple paths and assigns weights for these paths. This technique also introduces an optimal algorithm to find the shortest path to the target VMs based on the key nodes and key paths. However, if a node is detected then there is concern that the effect is reflected on neighbor nodes. Thus the algorithm also accounts for the re-identification of nodes and automatic weight addition for its’ neighboring nodes.

Furthermore, if a VM falls prey as per the analysis the migration of VM to hosts with less weight, where overloaded units and underutilized hosts play a major consideration. The proposed Secured Object Emigration technique calculates the priority for VM based on the occupied resources and the underutilized hosts evoke a mapping function to migrate the VMs. Here, when two unutilized hosts evoke the same mapping function then the traffic for migration is calculated based on the nearest hosts.
and path. Eventually, the data in the VM are stored in a secured way where the data are distributed based on first-fit or best-fit which provides security also creates search overheads in storage. To avoid the problems, the Watchword-Chunk algorithm is introduced that generates storage portions and categorizes them with defined percentages. Furthermore, the user involves directly in the placement of data portions by using their storage password into a storage pattern.

Consequently, with the detailed explanation of the proposed technique with the optimal placement of cloud units such as VM, host, and sensitive data in the cloud the Advance persistent Attack can be protected.

**Threat-Path Reckon Technique (TPR)**

In this section, the proposed TPR technique efficiently performs the determination of the shortest and possible attack paths and allocates the weightage of that path through a novel shortest and edge weighting algorithm and the Weighted Neighbor Matching algorithm. The detailed steps are followed. The network structure below indicates possible attack paths that may be traversed by an omniscient attacker to reach the goal if there are perceived vulnerabilities.

The attacker remains at any given instance on a \( N_i \) node and only moves to the next \( N_j \) node if the bug is present and exploitable. The likelihood of this state change is given by:

\[
Q_r(S_i) = Q_r^{u+}(S_i) \cdot Q_r^{exp}(S_i)
\]  

(1)

where \( Q_r^{u+}(S_i) \) is the probability presented by the vulnerability \( j \) in the cloud and \( Q_r^{exp}(S_i) \) the conditional probability that such weakness will be exploitable if, present? This means that the system view of the attacker only concurs with that of the real system if it is considered omniscient. Nevertheless, the perspective of the attacker does not agree with that of the real system in a real-world scenario. Rather his point of view is determined by the set of vulnerabilities he perceives to be present in the system. This collection of perceived vulnerabilities describes a new \( G_r \) environment graph that is a subset of the real system’s attack graph:
Since the knowledge of the vulnerability of the device is a major determinant in an APT, this knowledge is increased by the attacker through the discovery of the software running on the targeted cloud portion and the associated vulnerability. Knowledge of the program alone is not adequate, as it is unclear whether or not the software is patched. The probability of finding the vulnerability is heavily dependent upon the time elapsed since the vulnerability became publicly known. Assuming that the likelihood of this weakness resulting from Equation (1) decreases linearly over time, we have:

Where $t_q$ requires time to patch all vulnerable software, $t$ time is running out of patch release, and all software is patched to $t > t_q$. Base scores are used to determine the conditional likelihood $Q_{\text{pp}}^{\text{u}+}$ of exploiting a known vulnerability. Finally attack network structure is designed then to build the network from the attacker’s perception.

**TPR Attack Network (TPR-AN)**

Based on the configuration of the attack network in Figure 1, these are constructing the Network Attack (TPR-AN) from the viewpoint of the attacker.

It is not unusual for APT attacks to use several vectors of attack by following various exploits. The attack usually involves a series of transitions over time across the cloud network depending on the nature of the attack from one node to the other. In addition, other vulnerabilities, such as zero-day, occur over time meaning that the TPR-AN exhibits a dynamic property in terms of time indicating the addition of new attack nodes to the TPR-AN. Likewise, when found, the cloud provider patches up certain bugs that lead to the removal of the TPR-AN node associated with it. It means that attack nodes are vulnerable over time to be added and deleted to the TPR-AN. So to capture this dynamicity, the TPR-A’s corresponding attack graph can be expressed in terms of time as:

$$G_{\text{att}}(t) = (N_t, E_t)$$

where $N_t = \{n_y | y = 1, 2, 3, ..., h_i\}$ is a list of nodes involved in the attack up to the time $t$ and $E_t = \{r_y | y = 1, 2, 3, ..., k_i\}$ is the set of related edges for $N_t$. The addition and deletion of attack nodes at the time $t + 1$ can be expressed to satisfy the conditionality:

$$G_{\text{att}}(n, E) : \begin{cases} N_{t+1} = (N_t \cup \{n_{t+1}\}) - \{n_{t+1}\} \\ E_{t+1} = (E_t \cup \{r_{t+1}\}) - \{r_{t+1}\} \end{cases}$$

where $\{n_{t+1}\}$ denotes node addition with the growth of the dynamic attack network, and where $\{n_{t+1}\}$ denotes failure node removed from the attack network when the previous attack failed. Similarly, $\{r_{t+1}\}$ denotes the presence of an attacking edge associated with the added attack node while $\{r_{t+1}\}$ denotes the absence of the edge associated with the failure node.
Each node $N_i$ of the TPR-AN network casts a conditional probability distribution $Q_{r \{N_i \mid \text{parents}(N_i)\}}$ representing the quantified sample space constraint imposed on the child nodes by the parent nodes.

But although the attacker, according to the threat model, is a highly skilled APT threat actor who will resist backtracking to retain stealthiest and a long-lasting undetected presence, it is only reasonable to assume that the attacker could still switch paths depending on the attack scenario. It provides many directions inside the TPR-AN, then to apply the provisional prospects to the detection network.

**Conditional Probabilities with Detection**

The probabilities discussed so far are not detection factors. As the attacker traverses the cloud network targeting one node after another, it creates noise in the form of network traffic and device calls that serve when the seed for the IDS. Let $G_i^+$ denote the random variable indicating whether the IDS detects access to the $N_i$ node, and $G_i^-$ denotes the lack of detection in the cloud network. They measure identification likelihoods as:

$$Q_{r \{G_i^+ \mid N\}} = \frac{Q_{r \{N \mid G_i^+\}} - Q_{r \{N\}}}{Q_{r \{N\}}}$$  \hspace{1cm} (6)

Equation (6) above indicates the IDS’s success in detecting infiltration attacks. If the attacker is detected in the network and his advances stopped at this specific node, he can look for another route given the vulnerability is exploitable. Otherwise, he’ll be taken aback by the APT attack chain’s initial stages. Bearing in mind identification, we quantify the risk of undetected exposure as:

$$Q_{r \{N_i \}} = 1 - \prod_{i=1}^{n} \left(1 - Q_{r \{N \mid N_i\}}.Q_{r \{N_i\}}.\left[1 - Q_{r \{G_i^+\}}\right]\right)$$  \hspace{1cm} (7)

Consequently, considering Equation (7), the probability of accessing the $N_i$ node in the attack network via the parent nodes $N_j$ is given by:

$$Q_{r \{N_i \mid N_j\}} = 1 - \prod_{j=1, j \neq i}^{n} \left(1 - Q_{r \{N \mid N_j\}}.Q_{r \{N_j\}}\right)$$  \hspace{1cm} (8)

And in the light of IDS detection, maintain that the attacker does not backtrack to revisit an already accessed $N_j$ node from the current $N_i$ node, not only because it does not carry any new information or add any value, but also because the APT actor will want to preserve a stealthy presence by not creating excessive traffic by backtracking the IDS to warn. Next to assign the weights to detected paths.

**Shortest and Edge Weighting Algorithm**

From the perspective of the attacker, now present the proposed algorithm for finding the shortest path of attack in a given graph. The input to the algorithm is a matrix of weight while the output is the shortest path defined as a three-tuple composed of the effective distance, the cardinality of the steps of atomic attack in that direction, and the time cost of the attack. Initially, define the $WM = (r_{i,j})$ weight matrix as a mapping of the probability of effective exploitation of a vulnerability $r_{i,j}$ present in the network to the corresponding factor in the CM connectivity matrix:
The input matrices to the algorithm are therefore $WM_0$, $WM_1$, $WM_2$, and $WM_3$, given have 4 sources of attack. The corresponding sources of attack for those matrices are respectively $wm_0$, $wm_1$, $wm_2$, and $wm_3$. The performance consists of a maximum of Attack path items ($B_{eff}$) with the minimum number of Atomic Attack steps $A_t$. Thus the algorithm steps are described for generating the shortest path given the input matrices.

The algorithm extracts from each input matrix different attack paths and calculates with the least number of attack steps the weightiest path. We consider a single matrix as an input to the algorithm for illustration purposes. The resulting weight matrix derived from the mapping probabilities of exploiting the vulnerabilities present on the attack graph’s connectivity matrix is expressed as:

$$WM = \begin{bmatrix}
  r_{01} & r_{02} & \cdots & r_{0n} \\
r_{11} & r_{12} & \cdots & r_{1n} \\
r_{21} & r_{22} & \cdots & r_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
r_{n1} & r_{n2} & \cdots & r_{nn}
\end{bmatrix}$$  \hspace{1cm} (9)

In the weight matrix $WM = (r_{ij})$ for the range $\{0, 1\}$, we define the weight function by mapping discrete node probabilities to edges in the following way:

$$\xi : R(H_i) \rightarrow \mathbb{R}^+ \cup \{0\}, \text{ where } H_i \subseteq H_{att}$$  \hspace{1cm} (11)

Following Equation (12), the weight of the edge between two $N_i$ and $N_j$ nodes is expressed as $\xi(r_{ij})$. The cumulative edge weight (path) of an edge stream is given as product:

$$\xi(P) = \prod_{r_{ij} \in P} \xi(r)$$  \hspace{1cm} (12)

In addition, we define the cardinality of the path, which consists of multiple edges between $N_i$ and $N_n$, as an interval $\left[\xi(r_{[i,j]})\right]$ to indicate the number of edges between those nodes. It refers to the number of atomic attacks acts the attacker takes to enter the target node. From $WM = (r_{ij})$ we deduce from $N_0$ to $N_{v2}$ five guided feasible paths described as follows:
Therefore, the attacker has different paths from source $N_0$ to reach the target node $N_v2$, and their options increase after successfully implementing the first level exploit via either $N_1$ or $N_2$. To calculate the overall likelihood of getting to the destination, analyze all possible paths from source to destination open to the attacker. This is analogous to the probability of hitting the goal node after traversing and leveraging individual nodes that are not otherwise captured in the hierarchical assignment of nodes.

Path 3 is unfeasible in terms of reaching $N_v2$ as the weakness exploited through the edge $e_{(2,3)}$ does not allow any further efforts to reach the target but rather results in a DOS assault on $N_3$. It means we have four directions toward the target. Following Equation (11) and (12), we determine the shortest path of attack as the maximum effective distance ($B_{eff}$) from source $N_0$ to target $N_v2$ with the least number of steps of attack action:

\[
\text{Shortestpath}(SP) = \left\{ B_{eff} = \max \prod_{i,j=0}^{n} (r_{ij}) \right\} \text{min} \left| r_{ij} \right| \rightarrow r_{mn}
\]
\[
\begin{align*}
[B_{eff}^{|j|} r_{i[j]}] &= \max \{ \xi(p) \mid P : r_i \rightarrow *_{p} \} \\
\left( A t \sum_{p} \right) &= \min \left| \xi \left( r_{i[n]} \right) \right|
\end{align*}
\] 

(14)

where \( \left( A t \sum_{p} \right) \) is the number of steps of an atomic attack in a given direction \( P_i \) then to add the conditions of (5) to the list of feasible paths (6),

\[
\begin{align*}
(1) & \quad p_1 : \left[ B_{eff}^{|j|} r_{i[j]} \right] = 0.667, \quad \left( A t \sum_{p} \right) = 2 \\
(2) & \quad p_2 : \left[ B_{eff}^{|j|} r_{i[j]} \right] = 0.489, \quad \left( A t \sum_{p} \right) = 3 \\
(3) & \quad p_3 : \left[ B_{eff}^{|j|} r_{i[j]} \right] = 0.123, \quad \left( A t \sum_{p} \right) = 3 \\
(4) & \quad p_4 : \left[ B_{eff}^{|j|} r_{i[j]} \right] = 0.435, \quad \left( A t \sum_{p} \right) = 4
\end{align*}
\] 

(15)

From the above equation (15) feasible paths it is clear that the \( P_1 \) path resembles the shortest path of attack, as it bears the heaviest weight of the path with the least number of atomic attack moves. This implies that the attacker is more likely to reach the target node while doing the least number of attack actions via this path. The tie between path \( P_2 \) and \( P_4 \) in the number of attack steps may be broken by considering their respective path weights indicating path \( P_2 \) is the better option.

The \( P_5 \) path has the highest number of attack actions required suggesting that the attacker will have to exploit more vulnerability to reach the target if he is to choose that direction. It is also worth noting that path \( P_1 \) and \( P_4 \) lead to infringement of only one (Confidentiality, Integrity, and Availability) CIA tenet, i.e., confidentiality and availability respectively, which means that the resulting protection of the infringed network is different after attacks on these two separate paths. Because of this, route \( P_2 \) should then be compared to \( P_5 \) because both paths lead to the same breached security status of the target system by sharing the same end-attack edge \( r_{1,u2} \), that is, a complete violation of all three CIA principles. Road \( P_2 \) is, therefore, a better path than \( P_5 \) by having a 25% weighted path and 9.94% attack phase behavior.

The entire end arcs directly connected to the intended victim are the main edges for the respective direction being considered. The main nodes are those nodes that are directly linked via the key edge to the victim node. In our case, the key edges applicable to Table II are \( r_{(1,u2)}, r_{(3,u2)}, \) and \( r_{(u1,u2)} \), while the corresponding main nodes are \( N_1, N_3, \) and \( N_v1 \).

Since the resulting breached security state via key edge \( r_{(1,u2)} \) is a subset of that via key edge \( r_{(3,u2)} \), the security analyst may need to mitigate the attacks being conducted via the latter end edge of the attack. It means turning key node \( N_i \) into a failure node and this implies that the system is categorically secured against integrity and availability attacks via path \( P_3 \) unlike if the failed node was generated at \( N_1 \), the confidentiality, honesty, and availability of the network could be infringed via path \( P_4 \) even if it has the worst path metrics, i.e. 60 percent more attack behavior and a smaller path weight by 34.8%.

It should be noted that if there is a tie in the combined edge weight of two given paths, the tie is broken by selecting the path with the lowest \( \left( A t \sum_{p} \right) \). Likewise, if there is a tie in the number of \( \left( A t \sum_{p} \right) \) atomic steps for two specific attack paths, e.g. \( P_2 \) and \( P_4 \), the tie is broken by choosing the highest cumulative edge weight path \( \xi (p) \). In this approach, the proposed algorithm essentially determines the shortest path where a tie is arbitrarily resolved. Then perform the weighted neighbor matching for the estimated shortest path.

**Weighted Neighbor Matching**

In the node matching process, not only the exact node characteristics are provided as the basis for calculation, but also the effect of each re-identified node on the unidentified nodes is considered dynamically. A node that
has been successfully re-identified would intuitively have a certain feedback effect on its neighboring nodes. To suggest a weighted neighbor matching approach SWNM based on the matrix of dynamic similarity. When a node is successfully re-identified, its neighbors will increase their weight in the dynamic similarity matrix, and the node-related rows and columns according to the concept of consistency.

**Dynamic Similarity Matrix:**
Using the global and local characteristics of the node $i \in V^a$ and $j \in V^u$, the similitude of each pair of nodes is determined, and a dynamic similarity matrix $\text{dynamic}_\text{sim}$ is established. The similarity matrix update is coordinated with the next weighted neighbor matching.

**Weighted Neighbor Matching:**
The method maintains a node map $(i) = j$ with maximum similarity, keeping the number of node $i \in V^a$ which has the greatest similarity to the unrecognized node $j \in V^u$. The node is capable of allowing duplicates here. The mapped node $j$ is selected as the identified node according to map $(i)$ in each iterate of the node matching process, that is to say, the de-anonym node pair is $(i, j)$. And then, according to the theory of consistency, the dynamic similarity matrix $\text{dynamic}_\text{sim}$ is modified. The corresponding rows and columns of node $i$ should be removed when the above steps are complete. At the same time, for the node neighbors $i$, $1/\sigma$ node similarity value $i$ would be added to them in $\text{dynamic}_\text{sim}$ (for example, in Fig. 1, after successfully mapping the anonymized node ‘A’ to priori ‘a’ the similarity values of node pairs consisting of neighbors ‘A’ and ‘a’ are increased). The map $(i)$ then deletes the node $i$ and recalculates the values of the map $(i)$ to ensure that each $i \in V^a$ maps to the node with the highest similarity value in $j \in V^u$. Iterating these node pairs, the matching method until all nodes are successfully re-identified.

In algorithm 2, the de-anonymized node pairs contain all the anonymized graph and auxiliary graph node mapping results.

**Algorithm 2: De-anonymization of weighted neighbor matching**

Step 1: Acquiring the node features $F^a$ and $F^u$. Then to assign the node features for dynamic similarities.

Step 2: For all nodes $i$, $j$ belongs to $V^a$ and $V^u$. Next to mapping the node $i$, equalized to top similarities of node features ($F^u$) with another node $j$.

Step 3: Whether mapping length is greater than zero, which means maximum value of dynamic similarities is mapped to $u$. and mapping the values are equalized to $v$.

Step 4: Then apply the condition $\text{DeanonymizedNodePairs.add} ([u,v])$. Therefore, for all neighboring pairs $(a,b)$ of $(u,v)$ demarcated the principle as $\text{DynaSim}[a][b] += \text{DynaSim}[u][v]/\sigma$

Step 5: Finally to update the dynamic similarities of the node and mapped values then to proceeds the de-anonymized node pairs.
Therefore, the proposed Threat-Path Reckon technique analyzes and detects multiple paths and assigns path weights. This methodology also uses an optimized algorithm based on the main nodes and key paths to find the shortest path to the target VMs. If a node is detected, however, there is concern that the effect on neighboring nodes is mirrored. The algorithm also accounts for node re-identification and automatic weight addition for the adjacent nodes of the algorithm. Hence, if a VM falls victim as per the study, the migration of VM to hosts with less weight plays a major consideration where overloaded units and underused hosts consequently VM’s migration is performed through proposed Secured Object Emigration technique as follows.

Secured Object Emigration Technique

In this section, the heuristic-based VM migration algorithm is composed of two sections. The first step is to pick VMs to move from overloaded hosts, and we are proposing VMs-selection to decide such VMs. The second part is getting the mapping between VMs and underused hosts then we employed VM allocation to acquire it.

VMs Selection Design

In this segment, we proposed that VMs-Selection pick VMs from overloaded hosts to migrate, taking into account the connectivity and the size of the resources occupied by VMs.

VMs-selection’s concept is to pick VMs residing on \( h^+ \) \( H^+ \) with less traffic and smaller occupied resource to move, thus reducing the cost of contact. The algorithm will, therefore, be implemented as follows. Next, the VMs \( h^+ \) \( H^+ \) are sorted in ascending by \( O(u^+ C(u^+)) \) instead, beginning with \( u^+ \) the highest value of \( O(u^+ C(u^+)) \) pick the VMs in sequence and position them in list \( W \) until the host is overloaded. Finally, these selected VMs are the VMs to be migrated. The \( O(n \log n) \) sorting needs and the for-loop to pick VMs to migrate requires \( O(n) \),

\[
\text{where, } n = \max_{h^+ \in H^+} \left\lfloor \frac{s^+}{s^+} \right\rfloor.
\]

Thus the algorithm elaborately described the VM selection based on one host \( h^+ \).

VMs-selection is introduced on each overloaded host and all selected VMs are finally obtained. Subsequently, the VM was optimized to allocate the mechanism of the VM to find the destination hosts.

VMs Allocation Mechanism

In this section, we consider that the process of allocating to find the destination hosts for VMs to be migrated is modeled as an exchange.

Exchange Model

There are three players on the exchange market, including investors, sellers, and auctioneers. The buyers are referring to the VMs to move, and they need to purchase the services they need. The sellers are the underused hosts selling their idle capital. The third-party auctioneer solves the problem of mapping between buyers and sellers, and their final payment.

Buyer, i.e., VM to be migrated \( u^+ \) \( VM \) submits an offer defined by \( B_i \) to the auctioneer and \( B_i \) can be denoted as a 2-tuple: \( O(u^+), B_i \) \( vi \), where \( O(u^+) \) is the size of \( u^+ \)’s resource demand and \( v_i \) indicates \( u^+ \)’s valuation, i.e., the highest price \( u^+ \) is willing to pay for using these resources \( O(u^+) \). \( B = \{B_1, B_2, \ldots, B_N\} \) denotes all buyers’ bids, \( N \) is the total number of VMs picked from all overloaded hosts.
Algorithm 3: VM selection

Step: 1- the input of the algorithm is the set of VMs on $h^+_j$, $V(h_j)$. Then sort out the VMs by $\mathcal{O}(u_i) \cap (\mathcal{O}(u_j))$, in descending order.

Step: 2- $\emptyset$ value of VMs are allocated/assigned to $W$. Then consider the $u$ value is one to $\mathcal{O}(u_i)$, here assume the $u$-th total communication is $\mathcal{O}(u_i) \cap (\mathcal{O}(u_j))$.

Step: 3- if any of the one host $h^-_j$ is overloading, defined the following selection $w = w \cup (u^-_j)$.

Step: 4- finally, proposed algorithm will return the selection process every time of overloading the host.

Sellers, i.e., submits a $S_j$ bid to the auctioneer by the underused host $h^-_j \in H^-$. $S_j$ is referred to as a 2-tuple: $(o_j, p_j(m_j))$, where $p_j(m_j)$ corresponds to the unit price of the resource $o_j$ given by $h^-_j$, which is partly constant. $m_j = \sum_{i=1}^{N} y_{ij} O(u_{im})$ is the volume sold $h^-_j$. Figure 3 shows $h^-_j$'s unit price $p_j(m_j)$. The $p_j(m_j)$ is 2 if the $m_j$ is greater than 10. The price of the product decreases as the quantity sold increases, which represents the discount and may encourage consumers to move to this host at lower unit prices. All sellers' bids are referred to as $S = \{S_1, S_2, \ldots, S_M\}$, the total number of underused hosts is $M$.

The auctioneer begins matching buyers and sellers after receiving the bids $B$ and $S$ submitted by buyers and sellers and decides the buyers and sellers winners. The equivalent result the auctioneer produces is represented with matrix $X$. Upon creating the mappings between buyers and sellers, the auctioneer must decide the $b_j$ to $h^-_j$ payments and $u_{im}$, $V_i$, which is the difference between the purchaser’s value $u_i$ and payment $c_i$, reflecting the utility of $u_{im}$:

$$V_i = \begin{cases} u_i - c_i & u_{im} \in W^B \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

The utility $h^-_j$ is defined as the difference between the buyers’ total income $b_j$ and the discount for his resources asking price $b^-_j$:

$$V_j = \begin{cases} b_j - b^-_j & h^-_j \in W^S \\ 0 & \text{otherwise} \end{cases} \quad (17)$$
Where \( b_j = m_j p_j (m_j) \) the auctioneer’s goal in this auction model is to optimize social welfare which is the number of buyers and seller’s benefit as follows:

\[
SW = \sum_{i=1}^{N} V_i + \sum_{j=1}^{M} V_j
\]  

(18)

After the exchange, the model next section described the VM’s allocation design based on a novel algorithm.

**ALLOCATION ALGORITHM DESIGN**

In this section, to calculate the priority for VM based on the occupied resources and the underutilized hosts evoke a mapping function to migrate the VMs. Initially, the proposed technique introduces the allocation algorithm to obtain the mapping. The purpose of this algorithm is to move as much as possible the VMs that were chosen from the same overloaded host to the same underused host, which can minimize connectivity costs and lower buyers’ costs in terms of the unit price of the seller.

Let, \( S_i = u_i / \sqrt{p(um_i)} \) um_i’s bid density. Firstly, VMs are sorted in decreasing order according to \( S_i \). Let be the VMs list sorted by L. Then, turn to suit seller, select VM from the sorted list L. Suppose the currently selected VM is um_i.

The algorithm runs through all underused hosts to find hosts that can satisfy two Um_i. One condition is that the host can meet um_i’s resource demand and another is that the host’s asking price
is not less than um_i’s valuation u_i. To J_i = \{ j_h | y_j \geq Y(um_i), u_i \geq Y(um_i)q_h(m_j), 1 \leq h \leq M \} be the set of the host that satisfies the two conditions of um_i. If J_i = \emptyset, um_i match failed. If J_i \neq \emptyset, the host is matched with um_i and x_{ij} = 1

\[ j^*_h = \text{arg max}_{j \in J_i} EO_{ij} \]  \hspace{1cm} (19)

Here, h^* indicates the host h that um_i matches. Where EO_{ij} = u_i - O(um_i)p_j(m_j) is a revenue increment of um_i matching h_j.

First, the algorithm identifies other VMs residing on the original um_i’s host from L and places them in the list L_i \subset L, where L_i = \{ um_a | r(um_a) = 1, \forall um_a \in V \{ r(um_i) \} \} is the source host of um_i, and r(um_i) = 1 implies that um_i residing on r(um_i) is chosen for migration. Instead, in effect, pick VM to suit the seller L'_i. Suppose the VM you are choosing is um_v. If j^*_h meets um_v requirements, um_v matches j^*_h, and x_{uj} = 1. Otherwise, the algorithm goes through all underused hosts to find hosts that meet um_v conditions. If J_v = R(um_v) missed some sellers to match. If J_v \neq \emptyset, the host

\[ j^*_v = \text{arg max}_{j \in J_v} \phi_{uk} \] \hspace{1cm} (20)

Turns to the destination host of um_v, and x_{uk} = 1, where \phi_{uk} = \alpha R_{uk} - \beta \cos t_{uk}. The weight coefficients are \alpha and \beta.

The algorithm selects the next VM to suit in L_i until it fits all VMs in L_i. Then delete L_i distribution from L and start matching the next VM in L. The algorithm loops until all VMs in the suit. Thus we finally get matching matrix X.

The proposed Secured Object Emigration technique calculates the occupied resources priorities for VM, and the underused hosts evoke a mapping function to migrate the VMs. Here, when two unused hosts invoke the same mapping feature, then the migration traffic is calculated based on the nearest hosts and path. Finally, the data in the VM is stored in a secure manner where the data is distributed based on first-fit or best-fit, providing security while generating an overhead search. The cryptographic methods also just encrypt the data, and it may be vulnerable to APT after storage in an idle state. The following is introduced to resolve both Watchword-Chunk algorithms.

Watchword-Chunk Algorithm

Virtual machine Load refers to a total load of encapsulated programs within it. We are aware that there are certain applications with volatile and inconsistent data consumption rates. A service provider has a challenge in this scenario to provide the VM with sufficient storage to suit the changing needs at any unusual time without following a pattern. The service provider must ensure they do not overwhelm the physical storage and maintain consistent physical storage capacity.

The CSP’s goal will be to use as little physical storage as possible to reduce the cost of electricity and encourage green computing. Because this phenomenon is closely related to the problem of Bin Packing we solve it as one. Each VM load is an object and the physical storage units are the bins. According to our proposed architecture, physical storage is divided into classes such as S all (S), Medium (M), Large (L), and Big (B) covering all data load ranges. The physical storage segmented into these classes o distributes the total capacity which is the reason why it is called ‘watch’ and, secondly, since it has the proposed architecture, it can adapt to the different heterogeneous data artifacts
from the VMs, so we call it’ word.’ Because this phenomenon is closely related to the problem of bin chunking we solve it as one. VM load is an object, and the physical storage units are bins.

According to our proposed architecture, physical storage is divided into classes such as Small (S), Medium (M), Large (L), and Big (B) covering all data load ranges. The physical storage segmented into these classes o distributes the total capacity which is the reason why it is called’ Distributed’ and, secondly, since it has the proposed architecture, it can adapt to the different heterogeneous data artifacts from the VMs, so we call it’ Adaptive.’ In many ways the distributed and adaptive properties of the structured solution gain. This effectively manages the available resources for each VM both within and across the physical servers. They’re the same.

1. Distributes the loads across the storage pool uniformly and regularly, thus reducing the risk of overloading storage cloud
2. The search operation to fit into the new item is much less time-consuming because it determines the item size and scans for that particular class only in the storage pool instead of randomly searching the entire physical area.
3. Migration can be performed smoothly within the classes S, M, L, and B at the time of migration to turn off any physical store. It saves time for measuring the entire physical storage to find a suitable place for moving the VM in the entire physical pool.
4. It takes less time when deploying data to the storage compared to deploying data after determining a suitable location like First Fit and Best Fit do.
In the above algorithm, we use a function called Generating Chunks Number. In our watchword-chunk algorithm, the unusual solution is the special generation of many chunks of data to be stored...
in cloud storage units. The purpose of this approach is to protect the data from the risk of losing the confidentiality of data in the event of any leakage. The following description of this function is given.

**Password-Based Address Allocation for Storage Security**

If the data is stored in the raw form of cloud storage, then it is inherently dangerous. We have to make it untraceable to keep it safe from insider threats. The consumer, therefore, involves directly in the placement of data chunks in our built technique by using their database password formulated in a storage pattern. The locations of the data chunk storage are undisclosed and thus make the system foolproof and free from insider threats as shown in Figure 4.

**Figure 4. Virtual machine allocation Mechanism**

In Figure 5, each user produces a specific binary string for calculating the chunk size of the data to be distributed in the storage. The explanation for this was done is as follows.

a) The decision about fragmentation is based on the password of each user so it has greater security from the insider danger of co-users.

b) Segmenting and spreading the data into chunks across the data storage provides a balanced structure for storage rather than overloading and causing latency. From a security perspective, we maintain greater data confidentiality and privacy because the bits of data are stored in separate hardware in the cloud storage and not just in different virtual disk spaces in the same hardware.

c) By reduce the extended waiting times for other users to complete their request, the load spread through multiple storage points ultimately increases the IO Throughput.

d) By reducing the number of bins used, which reduces the number of active servers, we achieve better resource utilization and the carbon footprint. The reduction of active servers is a reduction in the consumption of energy that would have been used in the operation of these servers.

Hence, the aforementioned proposed techniques elaborately described an APT’s detection and protection in cloud computing through determining the shortest path and allocates the weightage to them. Subsequently, the novel technique migrates the VM’s and stores them to secure the data via robust password creation. Therefore, thus it accomplishes the detection with greater more accuracy. The following experimental calculation section clearly described the efficiency of the proposed techniques.
RESULTS AND DISCUSSION

This section effectively describes the proficiency of our proposed work by analyzing the shortest path through weightage allocation and performs the VM’s migration which offers strong protection of VM’s stored data. The competence of the proposed system is described by comparing the obtained experimental results with the conventional APT’s detection approaches.

System Specification

The proposed system has been implemented in MATLAB/SIMULINK to demonstrate competent security. The system specifications are,

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Platform</strong></td>
</tr>
<tr>
<td>OS</td>
</tr>
<tr>
<td>Processor</td>
</tr>
<tr>
<td>RAM</td>
</tr>
</tbody>
</table>

Simulation Results

Initially proposed techniques perform the detection of network shortest path with proper allocation of weightage. In our experiment first executes the detection of the shortest path only small numbers of nodes, thus it illustrated in figure 6.
After detecting the shortest path then needs to migrate the VM’s in figure 7 described the VM’s migration with energy consumption of nodes. Thus it clearly described the allocation of VM’s are achieved by our proposed technique in a fast and effective less energy consumption manner.

Figures 8 and 9 described the shortest path detection and allocation of a number of network nodes in the cloud. Hence it has shown our proposed technique proficiently achieves the detection and migration of VM’s if the network has numerous amounts of nodes.

Based on proposed techniques figure 10 and 11 described the VM’s migration is almost 100% successful when the number of task index is increased. And the energy consumption is also considerably reduced during task allocation performances.
Figure 8. Shortest path detection with huge numbers of nodes

Figure 9. VM's migration of a huge number of nodes

Figure 10. Shortest path detection and weightage allocation in cloud computing
Performance Evaluation

In order to evaluate the performance of the proposed system following evaluation metrics are taken into account, they are packet delivery ratio, BER, throughput, queue length, SNR, SNIR.

In figure 12 illustrates how much amount of packet delivery in a particular period. Thus the calculation is based on the expected packets from the destination and created packets from the source file. Therefore fig 12 clearly described the proficient performance of packet delivery between nodes.
In figure 13 described the performance of BER depends on Delay occurrences. In this way, the BER increases due to the increase of delay. This figure has clearly explained the efficiency of the proposed method in a bit error rate. The bit error rate or bit error ratio (BER) is the number of bit errors divided by the total number of bits transmitted over a period being studied. BER is a performance measure less than a unit, often expressed as a percentage.

In figure 14 described the performance of BER depends on SNR occurrences. This figure has clearly explained the efficiency of the proposed method in a bit error rate. One of the reasons for the increase of bit error rate here is signal to noise ratio. Whereas, the BER increases due to the increase of signal to noise ratio. However, the bit error rate needs to be reduced while the increase of signal to noise ratio to obtain efficient performance.
In figure 15 described the performance of BER depends on throughput occurrences. When the throughput of the proposed system increases, the bit error rate was increased slightly. This figure has clearly explained the efficiency of the proposed method in a bit error rate.

In figure 16 described the performance of BER depends on SINR occurrences. Initially, the signal to interference plus noise ratio increases with an increase in BER, then it reduces rapidly, after particular BER, the SINR has been highly increased. This figure has clearly explained the efficiency of the proposed method in a bit error rate.

Figure 17 denotes the queue length performances based on time. This presented the number of tasks waiting in the queue with its waiting time. Thus it is described the average numbers of data are equal to the mean number of data in the cloud.

Figure 18 described the total time of the histogram. The histogram above shows a frequency distribution for time to respond for detection of the attack path by the proposed system, and the height indicates the number of paths in each time range.

Figure 19 represents the utilization of window size, thus it shown the 0.5Mb of window size is utilized in our system with 10ms. Window size is the measured window size which depends on whether or not window scaling is true. If window scaling is not used or the scaling factor is 1 or if it is not clear if window scaling is true or not because the TCP 3-way handshake has not been recorded, then the two values are the same.

Figure 20 described the performance of SNR depends on throughput. As SNR increases gradually, the throughput of the proposed system increases linearly. Thus it denotes the amplitude of signal and noise in the data contains the high throughput and the rate of information transfer is greater more value in our proposed methods.

Figure 21 described the performance of SINR depends on capacity. Where the proposed system capacity is higher, when the signal to interference plus noise ratio was lower, and the capacity reduces linearly with the increase of SINR. Thus it denotes the amplitude of signal and noise in the data contains the high capacity and the rate of information transfer is greater more value in our proposed methods.

Figure 24 depicts the computational complexity of the proposed system. For the first 10 window size, computational complexity rapidly increases up to 80%. After that, the complexity increases gradually with the increase of window size. Both the run time as well as the computational complexity of the proposed system was much lower. Therefore in terms of cost-effectiveness, the proposed system exhibits improved performance.

**Comparison Results**

In this section, the performance of detection accuracy is compared with existing techniques such as DNS (Yan, Li, et al., (2020)), DL (Wan, Wu, et al., (2019)), and AICM (Cheng, Zhang, et al., (2019)). Thus it pronounced our proposed system for detection the APT in cloud performs in a great effective and accurate manner.

Consequently, the result section clearly illustrates that the proposed methods outperformed in APT’s detection and achieves greater more accuracy through determining the weightage allocating the shortest path, and provides high authentication to the VM’s stored data. As from figure 25, the accuracy of the proposed system achieves 99%, which is 16% higher than AICM and 35% higher than DNS. In this way, the overall accuracy of the proposed system highly outperforms the other existing methods.

Figure 26 shows the comparison of the proposed method with existing methods in terms of resource utilization, whereas the proposed system utilizes 92% of total resources, which is 3% greater than DNS and 10% greater than DL methods. Thus the proposed detection scheme has achieved optimized resource utilization value.

Figure 27 shows the comparison of the proposed method with existing methods in terms of resource allocation. The resource allocation of the proposed system achieves 90%, whereas the existing systems such as DNS, DL, AICM achieve 82%, 81%, 74% respectively. Accordingly, the proposed
Figure 15. BER Vs. throughput

Figure 16. BER vs. SINR

Figure 17. Queue length vs. time
Figure 18. Histogram of total times

![Histogram of total times](image)

Figure 19. Window size vs. utilization

![Window size vs. utilization](image)

Figure 20. SNR vs. throughput

![SNR vs throughput](image)
Figure 21. SINR vs. Capacity

Figure 22. Run time

Figure 23 depicts the execution time of the proposed system. For the first 10 window size, run time rapidly increases up to 45 seconds. After that, the execution time increases gradually to 50 seconds.
Figure 24. Computational Complexity

![Accuracy Rate](image1)

Figure 25. Comparison of accuracy performance

![Resource Utilization](image2)

Figure 26. Comparison of Resource Utilization Performance

![Resource Allocation](image3)
system attains 8% and 16% greater values by DNS and AICM. Thus the proposed VM allocation scheme efficiently allocated the resource (VM) while the system detects the attack.

Figure 28 shows the throughput of the proposed work compared with existing techniques and achieves better throughput of 91.8%. The packet throughput of the existing techniques including DNS, DL, AICM obtains 90%, 88%, 83% respectively. This indicates that the proposed work has enhanced performance as compared with the existing methods.

Figure 29 shows the comparison of the proposed method with existing methods in terms of energy consumption and the proposed scheme utilizes highly reduced energy when compared with other existing techniques.
CONCLUSION

Amid numerous strengths afforded by cloud computing, it also fosters security concerns that hamper the rapid adoption of cloud computing. In the current situation, the number of suitable attacks is immense, and the existing methods identify the attack nodes and perform recovery solutions. This research developed a novel method that analyzes APT behaviors and the path by which the outbreak could occur. Thus the weight allotment-based path detection is accomplished along with considering its effect on a neighboring node. Based on this weight allocation, the proposed system accurately migrates the VM’s efficiently tracking the underutilized hosts. Then the data stored are accessed securely with passwords-based storage generation and percentage categorization reducing the search overheads. Consequently, the experimental results show that the accuracy of the proposed system achieves 99% and has higher TPR with reduced FPR. In the future, the queueing time needs to be reduced to improve the throughput of the proposed work further.

FUNDING AGENCY

Publisher has waived the Open Access publishing fee.
REFERENCES


