Intrusion Optimal Path Attack detection using ACO for Cloud Computing

M. Jaiganesh [€], *G.ShivajiRao* [£], *P.Dhivya* [¥], *M.Udhayamoorthi* [Ω] and *A.Vincent Antony Kumar* [β]
[€], [£], [¥], [Ω] Karpagam College of Engineering, Coimbatore, Tamilnadu, India.
[β] PSNA College of Engineering and Technology, Dindigul, Tamilnadu, India
[€]*jaidevlingam@gmail.com*, [£]*shivajirao88@gmail.com*, [¥]*p.dhivya@kce.ac.in*, [Ω]*udhayamoorthi.m@kce.ac.in*, [β]*avakumar91@psnacet.edu.in*

Abstract. As the cloud infrastructure is simultaneously shared by millions of consumers, heinous use of cloud resources are also increasing. It makes ways to attackers to set up attacks by exploiting the vulnerabilities. And obviously, these attacks are leading to severe disasters as innocent consumers are unknowingly sharing cloud resources with harmful attackers. To prevent the occurrence of cloud attacks, attack graph based framework is proposed in this paper. Here, an attack path sketches an attack scenario by a streak of threats ranging in severity rating that shows how popular a particular cloud network service is in comparison. In a dynamic cloud environment, the proposed framework can disclose an optimal attack path thereby preventing cloud attacks. In cloud system the infrastructure is shared by potentially millions of users, which benefits the attackers to exploit vulnerabilities of the cloud. An instrument for analyzing multi-stage, multi-host assault scenarios in networks is the attack graph. It might not be possible for the administrator to patch every vulnerability n a large number of assault paths in an attack graph. The administrator might not be able to fix every vulnerability. To identify the most preferred or ideal assault path from a particular attack graph in a setting Ant Colony Optimization (ACO) algorithm is used.

1 Introduction

This approach suggests a framework for applying the Ant Colony Optimization (ACO) technique to determine the best assault path in a dynamic setting [1, 3]. Customized ACO algorithms are used to identify an optimal attack path given an attack graph and the exploits' severity scores. Complete attack graphs need exponential time to generate and have inherent scalability issues. Additionally, such an attack graph has issues with readability and visual representation, as well as redundant nodes and edges [6]. Attack graphs based on the explicit premise of monotonicity have been used in these challenges. Attackers will not be forced to give up their privileges once they have accrued a certain number of them. Only successful paths that lead to a desired node—referred to as the attacker's goal—are present in a limited attack graph.

An indicative virtual based cloud system is formulated with private virtual machines and public virtual servers [4]. This system is used for assessing the security realization. This scenario is portrayed in Figure 1. Two cloud servers, namely Server 1 & Server 2 are attached to the internet by means of an external firewall.

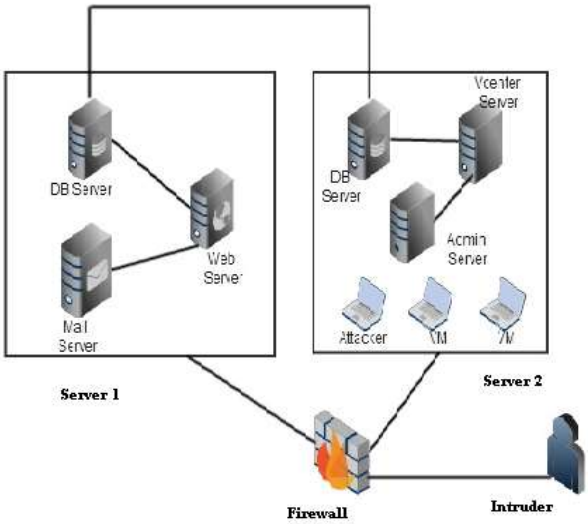


Figure.1 Sample virtual topology

In proposed system the attacker's next step is found out using ACO with Temporal Group Score (TGS) as a pheromone trail for ants [9]. So that on increasing the number of ants and epochs of the method, the ants perform various paths to find the attacker. Reaching the threshold will take less time if it is set at a lower value than if it is set at a higher value [7].

The Common Vulnerability Scoring System (CVSS) assesses the principal attributes of vulnerability and calculates a numerical score to indicate its severity [10]. This numerical value can then be interpreted into a qualitative representation (such as low, medium, high, and critical) to help consumers appropriately judge and estimate their security processes. The 3 metrics of CVSS are shown in Figure 1. Attributes along with possible values are shown in table 1.

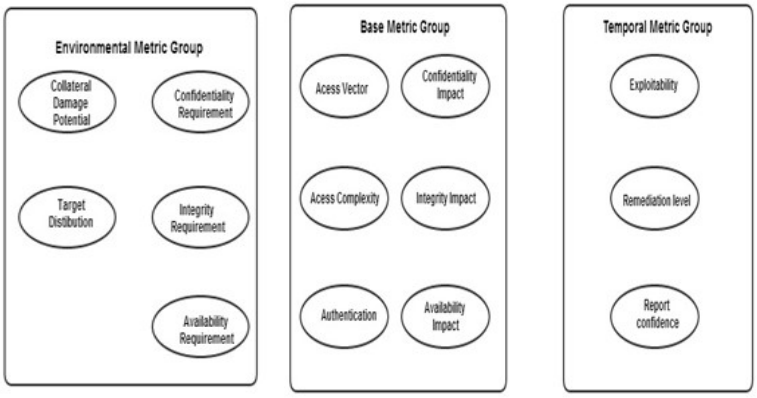


Figure.2 CVSS Metrics

Table 1: Attributes of Base metric group

Attribute	Description	Possible values
Access Vector (AV)	reflects how the vulnerability is exploited	Local (L), Adjacent Network (A), and Network (N)
Access Complexity (AC)	measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system	High (H), Medium (M), and Low (L)
Authentication (AU)	measures the how many times an attacker must authenticate for exploiting vulnerability	Multiple (M), Single (S), and None (N)
Confidentiality Impact (C)	measures the impact to confidentiality of a successfully exploited vulnerability	None (N), Partial (P), and Complete (C)
Integrity Impact (I)	measures the impact to integrity of a successfully exploited vulnerability	None (N), Partial (P), and Complete (C)
Availability Impact (A)	measures the impact to availability caused by a successfully exploited vulnerability	None (N), Partial (P), and Complete (C)

2 System Model

In every metric group, an equation is expressed for its measurement which produces a score between 0 and 10. This evaluation is based on a sequence of observations and assessments by security experts. Here, score 10 means the utmost vulnerability. An example assessment is shown in table 2.

Table 2: Base vector metrics

METRIC	EVALUATION	BASE SCORE
AV	[Native]	(0.395)
AC	[High]	(0.35)
AU	[None]	(0.704)
C	[Ample]	(0.66)
I	[Ample]	(0.66)
A	[Ample]	(0.66)

Attack Graph Generation

The illustration of an attack by the occurrence of numerous vulnerabilities is appropriately given by attack graphs. By means of a set of security-based constraints, system states are portrayed. For example, vulnerability is a specific host and the connection between multiple hosts. Transition from one state to another state represents vulnerability exploitation.

MULVAL (Multihost, multistage, Vulnerability Analysis)

Vulnerability data in databases, host's configuration details and all other related data are usually encoded in the form of Datalog facts. An attack graph expresses crucial attack paths and appropriate counter measures. Here, each node depicts precondition/consequence of an exploit. Knowledge about known vulnerabilities and connectivity details are furnished in an attack graph. So, potential threats and attacks can be envisioned appropriately in order to obtain the present security status of the system. If aa event is recognized as a potential attack, It can take steps to stop

it from infecting the cloud system or implement particular safeguards to lessen its effects. A Scenario Attack Graph (SAG) is a tuple SAG= (V, E) as shown in figure 3.

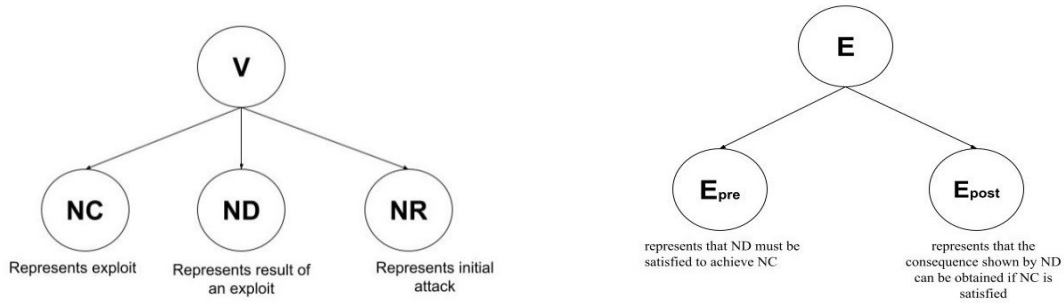


Figure 3: Vertices and Edges of SAG

3 Intrusion Optimal Path Attack detection

Giving the exploits retrieved from a few well-known public domain datasets severity scores in terms of Base Score (BS), Temporal Score (TS), and Environmental Score (ES) is one way to quantitatively analyze attack graphs. The exploit's conditional probability can be obtained by combining these three scores [12]. With the use of this score, an administrator could be able to calculate the most severe attack path and assess the level of severity in relation to a network service that is exploitable and vulnerable [14]. The way that vulnerability changes over time, however, is rarely taken into consideration. An exploit's level of threat changes in response to the release of fresh patches or the availability of additional technical information about the associated vulnerability. As a result, scores fluctuate. A methodology for creating dynamic environments whose vulnerability severity may vary over time has been presented in this work. An innovative method known as Ant Colony Optimization (ACO), which is based on a soft computing technology, has been introduced [16]. It takes an attack graph and the individual exploit scores to generate an ideal attack path dynamically [13]. More than 37,000 publicly disclosed vulnerabilities have CVSS ratings found in a few well-known public domain sources, such as NVD, Nessus, and Bug Traq [5]. The Base Score (BS), Temporal Score (TS), and Environmental Score (ES) are the three areas of concern that are measured by the CVSS assessment for every vulnerability. Three temporal metrics are available for TS: Report Confidence (RC), Remediation Level (RL), and Exploitability (E). Temporal Group Score (TGS) $= (\mathrm{E} \times RL \times RC)$

TGS usually lies within the range between 0.67 and 1.0.

$$\text{Temporal Score (TS)} = \text{BS} \times \text{TGS}$$

It is not required to apply the environmental score or the temporal score. However, in a real-world setting, the administrator always plans to use vendor-specific updates to address any vulnerability that are occasionally found [2]. A straightforward method is used to translate each exploit's CVSS Temporal Score into probability scores (p(e)).

$$p(e) = {}^{TS}/_{10}$$

The conditional likelihood of an exploit is how this probability is expressed. While certain attack pathways could require less work to exploit than others, some might not. However, once an attack channel is effectively utilized by a group of attackers, as time goes on, other attackers may decide to take the same route until the network administrator takes precautionary action. The term "optimal attack path(s)" refers to this collection of attack paths that are regularly exploited [8]. The

situation described above is similar to the Ant Colony Optimization (ACO) method, in that the pheromone deposit concentration changes with time. This is similar to situations in dynamic network environments where fresh technical details or vendor-specific updates might change the threat posed by exploits that comprise an attack vector. According to this work, the best attack path is the one that attracts the most attackers from a colony of attackers and whose probability of selecting that path finally approaches [1, 10 and 11].

4 Algorithms for Path Selection

The method is divided into two shared memory-using processes, Process1 (see method 1) and Process2 (see Algorithm 2). Figures 4 and 5 depict them, respectively. Process 1 declares a data structure in the shared memory that contains the probability value (Pi,j) as well as BS, E, RL, RC, and TGS (14).Semaphores protect data integrity as processes enter the crucial area.

```
Input: Attack Graph with initial configuration
Output: Optimal Attack Path
For each attacker k of colony do
   AttackPath = NULL;
   For i=0 to nodes do
         color[i]=White ;
   End
   start=0;
   Append(start);
   color[start]=Black;
   while start! = goal do
         wait(semid1);
         Read the values of E, RL, and RC from shared memory;
         Calculate the current TGS value;
         Calculate the probability of selecting the neighboring nodes;
         For each neighbor j of currentnode do
               If heus[j].TGS > threshold then
                     flag = 1;
               End
         End
         If flag = 1 then
               Find next node based on highest TGS value;
         End
         Else
               Find next node using RandomWalkAlgorithm;
         End
         color[start]=Black;
         Append(start);
         Update TGS ;
         signal(semid2);
   End
   Display AttackPath;
End
```

Figure 4: Algorithm 1. Process 1

Process 1 chooses an adjacent node by comparing the TGS values with the threshold once a certain node has been explored. Every time, Process 1 looks for a new attack path, and this is done until all of the attackers in the colony have run out of resources. It is evident that all following iterations converge to a single path after a reasonable number of iterations. Process 2 entails each node's TGS values continuously declining over time

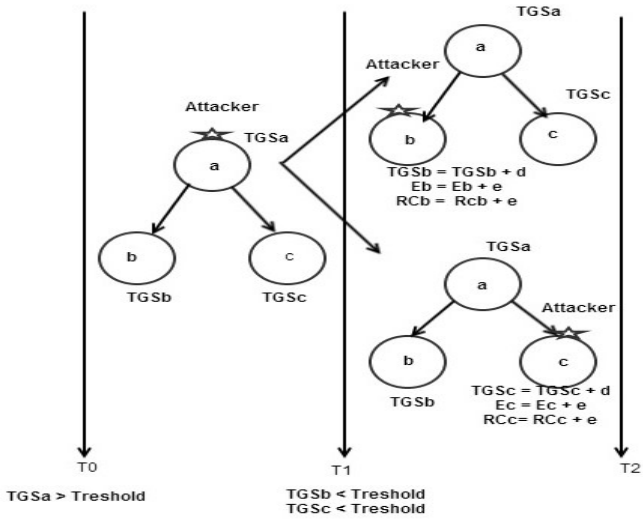
```
Input: Previous TGS value
Output: Updated TGS value
Settimer (timeslice)
While true do
        If CheckTimer(timeslice) = 1 then
            wait(semid2)
            Decay TGS values
            signal(semid1)
        End
End
```

Figure 5: Algorithm 2. Process 2

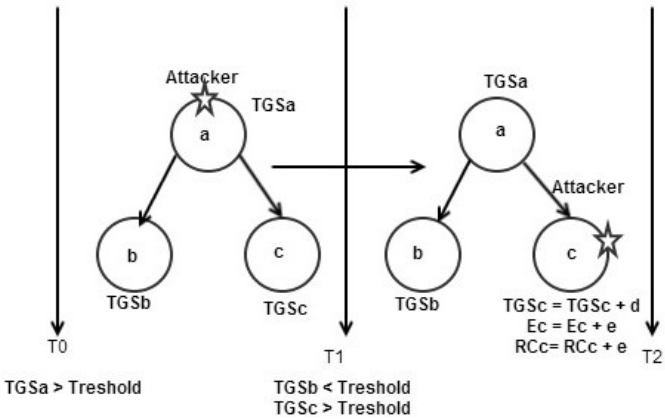
5 Results and Discussion

A straightforward three-node attack graph, shown in figure 6, can be used to explain how the values of the various TGS attributes (E, RL, and RC) fluctuate in a dynamic environment. It illustrates three distinct scenarios that show how the values of E, RL, and RC change over time. The three exploits in the attack graph are its nodes, A, B, and C. The TGS value for A is greater than the Threshold in Figure 6(a) during Time Slice T0. As a result, the attacker (shown by the little circle) can choose to execute B or C after executing vulnerability A during time slice T0.

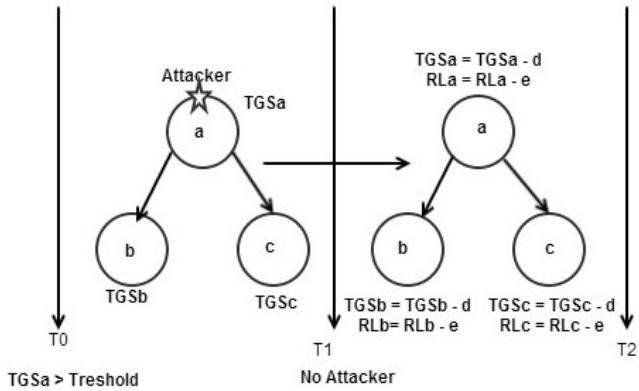
Nodes B and C, or TGSB and TGSC, have TGS values that are below the threshold. The Random Walk Algorithm will be used to determine which node will be chosen next. Therefore, B or C is the next exploit that the attacker may use in the next time slice, T1. In all scenarios, there will be a δ increment in TGSB or TGSC once related E and RC values have increased by a certain amount. A situation when the TGS value for exploit C over the threshold is shown in Figure 6(b). As a result, the attackers that come after will choose node C as the next node in time slice T1. These increase the TGSC value by a certain amount. A case where no attacker has arrived during time slice T1 is depicted in figure 6(c). Consequently, a δ amount decrease in the relevant RL values results in a corresponding reduction in the TGSA, TGSB, and TGSC values. scores based on the CVSS. There are a few well-known public websites where you can find the data required to calculate the likelihood scores. When an exploit's prerequisites are met, its conditional probability is measured by this score.



(a) Case 1



(b) Case 2



(c) Case 3

Figure 6: Variation of *E*, *RL*, *RC* with time

6 Conclusion

The proposed framework demonstrates the achievement of finding the trust rate of the virtual machines in the cloud environment. An innovative method utilizing the Ant Colony Optimization (ACO) technology is showcased to identify the best assault path from a specific cloud or an attack graph. As exploits vary dynamically over time, this framework recognizes the scenario appropriately. From the behavior of the virtual clients it traces the malware injected client with the exploit rate of the system. Here, the optimal attack path is defined as the one that the attacker's colony finds most appealing. An exploit's Temporal Group Score (TGS) is correlated with ACO's

pheromone update phenomenon. The ACO-based technique effectively detects optimal attack paths as it approaches the ideal solution.

References

1. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," ACM Comm., vol. 53, no. 4, pp. 50-58, Apr. 2010.

2. C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," Journal of Network and Computer Applications, vol. 36, no. 1, pp. 42–57, 2013

3. Karen Scarfone, Murugiah Souppaya, Paul Hoffman (2011), "Guide to Security for Full Virtualization Technologies", computer security NIST Special Publication 800-125.

4. Chun-Jen Chung ,Tianyi Xing (2013), "NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems",IEEE transactions on dependable and secure computing, vol. 10, no. 4

5. G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "BotHunter: Detecting Malware Infection through IDS-driven Dialog Correlation," Proc. 16th USENIX Security Symp. (SS '07), pp. 12:1-12:16, Aug. 2007.

6. G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," Proc. 15th Ann. Network and Distributed System Security Symp. (NDSS '08), Feb.2008.

7. M. Jaiganesh and A. Kumar, Fuzzy ART based User behaviour trust in cloud computing, Adv. Soft Comput. 324 (2014) 341–348, 2016.

8. P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph based network vulnerability analysis," Proc. 9th ACM Conf. Computer and Comm. Security (CCS '02), pp. 217-224, 2002.

9. X. Ou, W.F. Boyer, and M.A. McQueen, " A Scalable Approach to Attack Graph Generation," Proc. 13th ACM Conf. Computer and Comm. Security (CCS '06), pp. 336-345, 2006.

10. X. Ou, S. Govinda vajhala, and A.W. Appel, "MulVAL: A Logic- Based Network Security Analyzer," Proc. 14th USENIX Security Symp., pp. 113-128, 2005.

11. Mitre Corporation, "Common Vulnerabilities and Exposures, CVE," http://cve.mitre.org/, 2012.

12. National Institute of Standards and Technology, "National Vulnerability Database, NVD," http://nvd.nist. gov, 2012.

13. Gutjahr, W.J.: Aco algorithms with guaranteed convergence to the optimal solution. Information Processing Letters 82(3), 145–153 (2002)

14. Nishant, K.Sharma , P. :Krishna, V.," Load Balancing of Nodes in cloud using Ant Colony Optimization", Computer Modeling and Simulation, 14th International Conference on 28-30 March 2012, 3-8

15. Stutzle, T., Dorigo, M.: A short convergence proof for a class of ant colony optimization algorithms. IEEE Transactions on Evolutionary Computation 2002, 358–36 (2002).

16. Ghosh, N., Nanda, S., Ghosh, S.K. (2010). An ACO Based Approach for Detection of an Optimal Attack Path in a Dynamic Environment. In: Kant, K., Pemmaraju, S.V., Sivalingam, K.M., Wu, J. (eds) Distributed Computing and Networking. ICDCN 2010. Lecture Notes in Computer Science, vol 5935, 509–520.