RESEARCH ARTICLE OPEN ACCESS

A Survey on Hidden Markov Model (HMM) Based Intention Prediction Techniques

Mrs. Manisha Bharati*, Dr. Santosh Lomte**
*(Department of Computer Science, Dr. BAM U, India)
** (Department of Computer Science, Dr. BAMU, India)

ABSTRACT
The extensive use of virtualization in implementing cloud infrastructure brings unrivaled security concerns for cloud tenants or customers and introduces an additional layer that itself must be completely configured and secured. Intruders can exploit the large amount of cloud resources for their attacks.
This paper discusses two approaches In the first three features namely ongoing attacks, autonomic prevention actions, and risk measure are Integrated to our Autonomic Cloud Intrusion Detection Framework (ACIDF) as most of the current security technologies do not provide the essential security features for cloud systems such as early warnings about future ongoing attacks, autonomic prevention actions, and risk measure. The early warnings are signaled through a new finite State Hidden Markov prediction model that captures the interaction between the attackers and cloud assets. The risk assessment model measures the potential impact of a threat on assets given its occurrence probability. The estimated risk of each security alert is updated dynamically as the alert is correlated to prior ones. This enables the adaptive risk metric to evaluate the cloud's overall security state. The prediction system raises early warnings about potential attacks to the autonomic component, controller. Thus, the controller can take proactive corrective actions before the attacks pose a serious security risk to the system.
In another Attack Sequence Detection (ASD) approach as Tasks from different users may be performed on the same machine. Therefore, one primary security concern is whether user data is secure in cloud. On the other hand, hacker may facilitate cloud computing to launch larger range of attack, such as a request of port scan in cloud with multiple virtual machines executing such malicious action. In addition, hacker may perform a sequence of attacks in order to compromise his target system in cloud, for example, evading an easy-to-exploit machine in a cloud and then using the previous compromised to attack the target. Such attack plan may be stealthy or inside the computing environment, so intrusion detection system or firewall has difficulty to identify it.
Keywords - Hidden Markov model (HMM), user action Recognition, smart home systems, intention prediction, sequence learning, attack plan.

I. INTRODUCTION

Prediction techniques are essential tools to reach an effective decision on countering an attack. There are two main models that can be used for the prediction target namely, (1) Finite-context models, that are applied using Markov Models, MM, and Variable Order Markov Model, VMM. These models assign a probability to a symbol based on the context in which it appears and, (2) Finite-state models. These models are applied using Hidden Markov Models, HMM, which are composed of an observable part called "events," and a hidden part called "states." A state stores information about the past since it reflects changes in the system from the start to the present moment. A transition indicates a state change and is described by a condition that needs to be fulfilled in order to enable the transition. Events are observed with different probability distribution depending on the state of the system. These models provide flexible structure that can model complex sources of sequential data. However dealing with HMM typically requires considerable understanding and insight look into the problem domain in order to restrict possible model architectures. A bad prediction model may result in: (1) reducing network/host performance, (2) wrongly disconnect users from the network/host, (3) high costs for administrators' reestablishing services, and (4) a DoS attack for the network, which will eventually have to be disabled.

Attacker may combine multiple security vulnerabilities into an intelligent attack. For such a target attack, he often adopts persistent attack approach consisting of a sequence of attack behaviors continuously until the target is compromised.

For example, hacker may first attempt to compromise an easy target in a cloud. A compromised machine, either by malicious insider or service hijacking, may abuse cloud computing or attack one or more machines in the cloud, causing more serious damage than in a network environment where machines are distributed and independent.

Intrusion detection system (IDS) and firewall, monitoring network activities at gateway level, are considered as efficient attack prevention mechanisms. The traffic in/out gateway violating pre-defined rules will be alerted or blocked, but not for that inside the perimeter. Hence, suspicious events inside a cloud might not be alerted by IDS or firewall and a sequence of planned attacks might be successful even IDS and firewall is deployed at the gateway level. Therefore, intrusion detection in cloud should examine both inbound and outbound traffic. In a cloud environment, many audit logs are recorded, such as web traffic or system log, and many alert logs are also reported by IDS or firewall. A vast amount of logs requires human and computing resources to filter out false alarms and to identify real attacks. Some attack attempts recorded in a log might not be successful as the target machine does not possess the vulnerability exploited by the attack, like an Apache server is attack-free from IIS vulnerabilities. Therefore, alert or warning from a log might not be able to plot the whole picture. However, multiple logs could indicate if a previous attack is successful as a compromised target may leave some attack trace in different logs. Multiple logs in cloud should be examined and analyzed to identify successful attacks.

II. BACKGROUND AND LITERATURE REVIEW

The potential impact of intrusions in cloud systems steadily increases because of the huge amount of cloud resources that an intruder may control and use to implement further attacks. Furthermore, the deficiencies of the current intrusion detection technology hinder its adoption in clouds.

Most of these technologies suffer of single point of failure and none of these solutions use an autonomic response, risk metric, or prediction features. In this section, we highlight these features in details.

Widely adopted mechanisms to implement a fault tolerant system [6] includes (1) replication of software agents, (2) redundant processing components, (3) integrity checks for self-healing, (4) reconfigurable hardware and restructuring architectures, and (5) fault detection using Heartbeat messages. The proposed framework adopts 2, 3 and 5 as the most effective mechanisms to achieve the self-resilience and the fault tolerance capabilities.

Alerts correlation and risk assessment processes play a critical role in both the detection and prediction phases. The detection component produces a large number of alerts that disturbs the intrusion response component and this can increase the impact on the network and results in a DoS.

There are two different approaches for alerts correlation namely: (1) the alert filtering approach that selects just true alerts from the raw ones that are generated by detection components and it causes false negatives in prediction but prevents the application of high impact reactions to the network by the response component, (2) the alert severity modulation that modulates the quality of alerts and generates prediction alarms for the most interesting steps of multi-step attacks and consequently it improves the prediction accuracy.

A planned attack normally is performed in a long term time frame with persistent and stealthy attacks to avoid violation of IDS rules, such as two password guessing constantly in a long duration which triggers no alert and will not be discovered by IDS. To determine if a machine is under attack, the proposed approach extracts and analyzes the logs related to the observing machine to identify whether an attack sequence exists. This study adopts hidden Markov Model (HMM) to model the sequence of anomaly behaviors. As mentioned above that different attack strategies may leave traces in different logs. An attack plan often lasts for a long duration, so the detection should infer and correlate various logs in a period of time. A successful attack consists of at least three stages: (1) reconnaissance: gathering information from a target machine, such as scan or password guess; (2) intrusion: intruding/exploiting the target with the vulnerability found; (3) attacking: using the compromised machine to attack others.

The model consists of three states corresponding to the three stages described above, and the observations from the analyzed logs are shown in the second layer, where each observation requires the correlations of logs with extracted features listed in the lowest layer. Each machine initially is at state Reconnaissance, as each is under the threat of being scanned or discovered. A machine whose state transited from the initial state to the next state, Intrusion, indicates that an intrusion happened after the target has been scanned or attempted login. The logs related to the target are analyzed and the observed events will be obtained to apply the proposed HMM to see if the state has been transited. Attack sequence will then be identified as described above.

III. ACIDF PREDICTION AND EARLY-WARNINGS

The basic idea underlying the prediction model is that most intrusions consist of many stages and each early stage prepares for the later one. ACIDF integrates heterogeneous IDSs to generate more accurate and synthetic alerts. The output of the IDSs usually includes a large number of alerts as stream data which usually unordered and changes frequently. Using traditional techniques with such data is a big challenge. The HMM algorithm is one of the best ways to tackle this weakness. HMM works well dealing with streaming inputs. It is fast and can be used to predict future sequences [8, 9]. We adapt the HMM to provide the predictability and early-warning feature to ACIDF. In this model, the sequence of events that match attacks signature rules in the correlation tree represents a series of state transitions with a certain probability where each event is not directly visible but output dependent on the event is visible, the output in this case is the attack phase or state. To build this model, we consider four main issues, (A) formally defining the model using some notation of, (B) the implementation of the model, (C) the training of the model, and (D) the evaluation of the model.

3.1. THE FORMAL DEFINITION OF THE MODEL

Assume that the cloud system can be modeled by *N* different states, i.e., *S* = { *s1, . . . ,sN* } representing different security conditions. The security state of the cloud system changes over time and the sequence of states occurred in the cloud system is denoted as *X = x1, . . . ,xt,* where *xt* ∈ *S*. The cloud system is monitored by *K* host-based and network based IDS sensors. A sensor *k* generates observation messages from the observation symbol set = *{1, . . . , M}*, where *M* is the number of messages for sensor *k*. The sequence of observed messages is denoted as *O* = o*1*, . . . , o*t* , where o*t* ∈ *V* is the observation message received at time *t*. The HMM consists of a state transition probability matrix *P*, an observation probability matrix *Q*, and an initial state distribution vector *π* and is denoted by a tuple (*P, Q, π*). The state transition probability matrix *P* describes the probabilities of transitions between the states of the model. Each entry, *pij*, describes the probability that the model will transfer to state *sj* at time t + 1 given that it is in state *si* at time t, i.e., *pij* = *p*(*xt* +1 = *sj*| *xt* = *si*), 1 ≤ *i*, *j* ≤ *N*. The observation probability matrix *Q* describes the probabilities of receiving different observations given that the system is in a certain state. Each entry, *qn(m)*, represents the probability of receiving the observation symbol *m* from sensor *k* at time t, given that the system is in state *sn* at time t, i.e., *qn(m)*= *P(t* = *m*| *xt = sn),* 1 ≤ *n* ≤ *N*, 1 ≤ *k* ≤ *K*, 1 ≤ *m* ≤ *M*.

3.2. THE PREDICTION MODEL IMPLEMENTATION

The basic idea underlying our proposed prediction model is to employ a HMM to track the evolution of the attack in the system. That way, while an attack is in progress, the state changes and we can trigger appropriate responses based on a predefined confidence level threshold, which would result in a lower false positive rate. The prediction component has all the detailed information about the malicious activity such as severity, confidence level, and the cost of asset targeted. The following sections describe the prediction components in details and give a practical example for the model.

3.2.1. THE PREDICTION COMPONENTS

1) States: the system is assumed to be in one of the following 4 states: Hale (H): indicates that system is working well and there is no malicious activity or any attempt to break into the system, Investigate (I): indicates that malicious activities are attempted against the system, Attack (A): indicates that intrusion has been started and is now progressing, and Penetrate (P): indicates that intrusion successfully compromised the system. The graph shown in Fig. 1 defines the relationship among these states.

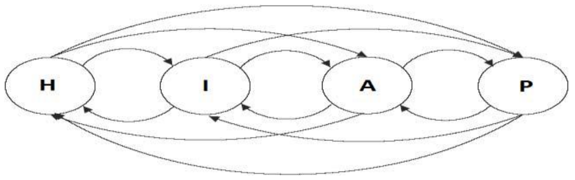

Fig. 1. The relation between the proposed HMM states

2) Observations: *O* = o*1*, . . . , o*K* , are alerts from the detection sensors. Observations cause the system model to move among states. We consider the severity of these alerts as observation and each alert have four priorities reflects the state of the system: Low, Medium, High, and Very high or (*L*, *M*, *H*, *V*). The alert severity function is described later inthis section.

3) State Transition Probability Matrix (*P*): the state transition probability matrix describes the probability of moving among states. The following steps describe how to build the HMM states and to calculate the transition possibilities.
a) Construct a signature sequence vector to contain the sequences of signatures that define each attack.
b) List all possible combinations of the signatures that may be shared by more than one attack in the signature sequence vector. At the same time let every possible instance represents a state in HMM, then refine these states to construct a minimal state set,c) Calculate the transition possibility between states using the Forward-Back Propagation] training algorithm to find, given an output sequence or a set of such sequences, the best set of state transition and output probabilities. The idea is to derive the maximum likelihood estimate of the HMM parameters given the set of output sequences.

4) Observation Transition Probability Matrix (*Q*): the observation transition probability matrix describes the probability of moving among observations.

5) Initial State Distribution Vector (*π*): it describes the probability of states when our framework starts.

6) Alert Observation Probability Matrix (*Å*): describes the probability of having a specific alert in a specific state.
This matrix helps in computing the alert severity function as we will explain later. *Å* is built based on the training data in the attack dataset.

7) Assets Cost Matrix (*C*): Each of the states of the system is associated with a cost vector, indicating the potential consequences of the state in question. E.g., A cost vector of the database server in the cloud system for the four defined states (H, I, A, P) can be defined as *C(DBServer) = {0, 3, 7, 25}*. A group of these vectors constructs the final *C* Matrix.

8) The Output or emission probability Matrix (*Y*): It represents how likely the output result is for each sequence of attack states. It is an empty matrix that collects the final output probabilities.

9) Alert Severity Function: It describes the severity of each alert at specific state *s*. We model this severity function based on Eq.1 as shown in Eq.2 and 3. The computed severity is mapped to one of the four priorities (*L, M, H, V*) to reflect the state of the system as we will explain later in the prediction algorithm.
$ARs = (ACs * AP * DRs)/ NFs$ (2)= $(ACs * (CSeverity * NOccurance / AFrequency) * DRs) / NFs$ (3) *Where,*

- ♣ ARs: Alert Risk at a specific state *s*,
- ♣ ACs: Asset Cost at a specific state *s*. AC is computed using the *C* vector and it represents the potential consequences of the state *s* on the asset in question
- ♣ *AP: Alert Priority. It* is computed based on *CSeverity, NOccurance,* and *AFrequency* as shown in Eq.3,
- ♣ *CSeverity:* Current alert severity defined by the firing IDS.
- ♣ *NOccurance:* Number of occurrences of current alert in a specified correlation time slot defined in the correlation process,
- ♣ *AFrequency:* Acceptable frequency of this alert per day based on the training data computed from the attack dataset.
- ♣ DRs: Detection Reliability at a specific state *s*. It is computed according to the alert position corresponding to *s in* Matrix *Å*.
- ♣ NFs: A fixed Normalization Factor that is computed according to the maximum values appeared during training phase for ACs, AP, DRs, and Maximum Alert Risk (*MR*) where ARs belongs to the range (0-MRs). All these values are computed

for each state independently. Thus, $NFs = (Max(ACs) * Max(AP) * Max(DRs)) / MRs$.

10) HMM Prediction Algorithm

The Pseudo Code for the prediction algorithm and the alert risk modulation approach is shown in Algorithm 1.

Algorithm1: HMM Prediction and Alert Risk Modulation

The algorithm starts by computing the alert risk and then mapping this risk to one of the 4 defined risk levels.

```
1.Algorithm HMM_Prediction & Alert_ Risk_ Modulation
2. Inputs: Alert, Accept Alert Freq, P, Q, π, Å, C, Cur_ Obs, Obs_prob,
s, Asset, n, Threshold, L, M, H, V.
3. Begin
4. $AC\ s$= Compute Asset Cost (Asset ,s, C)
5. A Frequency = Choose Acceptable Alert (Alert, Accept Alert Freq)
6. $DRs$=Compute Detection Reliability (Alert, s, Å)
7. $NFs$=(Max( $AC\ s$)* Max(AR)* Max( $DRs$)) / $MRs$.
8. $ARs$= ($AC\ s$ * (CSeverity * NOccurance / AFrequency) * $DRs$) / $NFs$
9. IF ($ARs$ ≤L) Then // Alert Risk Level is low (L= 0.25)
10. Obs_prob =1
11. Else IF ( $ARs$ ≤M) Then // Alert Risk Level is Medium (M=0.50)
12. Obs_prob =2
13. Else IF ( $ARs$ ≤ H) Then // Alert Risk Level is High (H=0.75)
14. Obs_prob =3
15. Else // Alert Risk Level is Very High (V>0.75)
16. Obs_prob =4
17. End If
18. sum_tmp =0
19. sum_final=0
20. IF (Cur_Obs=1) Then // Initial Observation
21. For (i=1 to n)
22. Tmp[Cur_Obs, i] = π[i] * Q[i, Obs_prob]
23. sum_tmp = sum_tmp + Tmp[Cur_Obs, i]
24. End For
25. For (i=1 to n)
26. Final[Cur_Obs, i] = Tmp[Cur_Obs, i] / sum_tmp
27. End For
28. Else // Other Prediction Observations
29. For (i=1 to n)
30. For (k=1 to n)
```

```
31. sum_final= sum_final + Final[Cur_Obs-1, k]*P[k,i]
32. End For
33. Tmp[Cur_Obs, i]= Q[i, Obs_prob] * sum_final
34. sum_final=0
35. sum_tmp = sum_tmp + Tmp[Cur_Obs, i]
36. End For
37. For (i=1 to n)
38. Final[Cur_Obs, i] = Tmp[Cur_Obs, i] / sum_tmp
39. End For
40. End IF
41. IF (Final[Cur_Obs, n]≥ Threshold) Then //Check Prediction
42. Print("Intrusion Prediction rate is high=",(Final[Cur_Obs, n])
43. End IF
44. End
```

If the observation status variable, Cur_Obs, is equal to one, the algorithm uses vector π otherwise it uses matrix *P*. Finally, it fires an alert if the final prediction probability is higher than a defined prediction threshold.

IV. Attack Sequence Detection (ASD)

The test data is collected from a campus network. In order to better represent the applicability of the proposed system, two types of logs, audit logs and alert logs, are included in the preliminary experiment, one from web traffic (audit log) and the other from IDS (alert log). Logs of consecutive five weeks are collected with average of four millions of web requests each day and ten thousands of IDS alerts each week. To identify attack sequence, the logs of the first week are observed and the suspicious machines in state Reconnaissance are extracted for further analysis. The logs of the following weeks are analyzed and the subsequent attacks related to the suspicious machines are identified. he preliminary results show that the proposed system can identify attack plans with persistent and low frequency attack activities.

V. CONCLUSION

In this paper, we have presented a cloud based IDS, ACIDF, that provides autonomic and prediction capacities that enable it to work efficiently with cloud environments. ACIDF enables cloud consumers to protect their cloud applications and data from various types of harmful cyber-attacks.

Also ASD study examines the stages of an attack plan and analyzes logs to identify attack sequences. Hence usage of Hidden Markov model, suitable for predicting attacks before they are obvious and also for recognizing time sequence events to detect attacks. The preliminary results show that both the proposed detection models are efficient to identify attacks.

REFERENCES

[1] Online Risk Assessment and Prediction Models For Autonomic Cloud Intrusion Prevention Systems: 978-1-4799-7100-8/14 /$31.00 ©2014 IEEE

[2] Hisham A. Kholidy, Fabrizio Baiardi, Salim Hariri, "DDSGA: A Data-Driven Semi-Global Alignment Approach for Detecting Masquerade Attacks", in IEEE Transactions on Dependable and Secure Computing, accepted and in printing in May 2014.

[3] Top Threats to Cloud Computing V1.0 https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf

[4] W. Lee, S. J. Stolfo, and K. W. Mok, "Mining Audit Data to Build Intrusion Detection Models," Fourth international conference on knowledge discovery and data mining, New York, AAAI Press 66-72, 1998.

[5] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," ACM Trans. Inf. Syst. Secur., vol. 3, no. 4, pp. 227–261, Nov. 2000.

[6] C. C. Lo, C. C. Huang, J. Ku, "A Cooperative Intrusion Detection System Framework for Cloud Computing Networks," ICPPW '10 Proceedings of the 2010 39th International Conference on Parallel Processing Workshops, IEEE Computer Society, pp. 280-284, Washington DC, USA, 2010. ISBN: 978-0-7695-4157-0.

[7] S. Sridhar, M. Govindarasu, and C Liu. Risk Analysis of Coordinated Cyber Attacks on Power Grid. Control and Optimization Methods for Electric Smart Grids – Power Electronics and Power Systems, Vol. 3, Part 3, pp 275-294, 2012.

[8] C. Zhou, C. Leckie, and S. Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. Computer and Security, Vol. 29, pp 124-140, 2010.

[9] S. Zargar, H. Takabi, and J. Joshi. DCDIDP: A Distributed, Collaborative, and Data-driven Intrusion Detection and Prevention Framework for Cloud Computing Environments. Proceedings of International Conference on Collaborative Computing: Networking,

[10] P. Kumar, Nitin, V. Secgal, K. Shah, S. S. P. Shukla, and D. S. Chauhan, "A Novel Approach for Security in Cloud Computing using Hidden Markov Model and Clustering," Information and Communication Technologies (WICT), pp.810-815, 2011.

[11] wordiQ.com, “Hidden Markov model – Definition,” http://www.wordiq.com/definition/Hidden_Markov_model

[12] B. Bauer and K.-F. Kraiss, “Towards an Automatic Sign Language Recognition System Using Subunits,” Proc. Gesture Workshop, pp. 64-75, 2001.