AttackDive: Diving Deep into the Cloud Ecosystem to Explore Attack Surfaces

Salman Manzoor, Jesus Luna and Neeraj Suri
Technische Universität Darmstadt, Germany
Email: {salman,jluna,suri}@deeds.informatik.tu-darmstadt.de

***Abstract*—A multitude of issues affect the broader adoption of Cloud computing, with security arguably being amongst the most significant. To address security concerns, the process of threat analysis is advocated to assess potential attacks that can undermine the security goals. However, conducting threat analysis for the Cloud is a non-trivial task given the plethora of attack surfaces entailed in the multiple layers of the operational stack and the resource/customer interfaces. Consequently, contemporary Cloud threat analyses approaches primarily focus on specific services/layers without analyzing the malicious behaviors over the complete multi-layered Cloud ecosystem. Hence, the need is of a comprehensive Cloud threat analysis approach that can (a) analyze the spectrum of malicious behaviors stemming from the vulnerable service interactions across the multi-level operational stack, and (b) correspondingly enumerate the attack surface exploitability by varied types of attackers.**

We achieve such a holistic Cloud threat analysis via a novel multi-level modeling of Cloud operations to obtain a comprehensive behavioral profile of its underlying services. Our proposed approach, using Petri Nets, targets the identification of core operational states to enumerate the normal sequence of Cloud operations along with the triggers that provide the state transitions. The obtained states transition enumerate comprehensive multi-level state space baseline of "normal" sequences and also constitutes to identify multi-level vulnerabilities not recognizable by the traditional single-level threat analysis.

***Index Terms*—Cloud security; Petri Nets; Cloud threat analysis**

I. Introduction

As the usage of the Cloud proliferates, the corresponding need for the often lacking security assurance (including the threat analysis) from the Cloud Service Providers (CSPs) also grows. From the perspective of the CSP, providing security assurance is not trivial given the multitude of possible attack surfaces to consider. These arise not only from the traditional threats entailed in the classical Internet protocol stack, but also from the newer Cloud-specific architectures.

One of the advocated methods to provide security assurance, and to alleviate the customer's security concerns, is to conduct threat analysis i.e., evaluating the system for vulnerabilities that can be exploited by adversaries. Threat Analysis (TA) [1], [2] is an approach to investigate potential attacks that can undermine security goals. However, due to the plethora of services/technologies involved in the Cloud operational stack and interfaces across varied resources and customers, the current Cloud threat analysis approaches typically focus on a particular service or consider only a particular technology [3]–[5]. Hence, most TA schemes reveal attack surfaces that are pertinent only to that particular service/technology, and without covering the holistic operations of the Cloud. Consequently, the current TA approaches fail to analyze behavioral repercussions of the vulnerable service/technology across the varied levels of operational stack in the Cloud ecosystem.

Hence, a comprehensive threat analysis of the Cloud is required that can (a) analyze malicious behaviors stemming from the interactions of vulnerable services across the multi-level operational stack, and (b) correspondingly enumerate the multi-level attack surface exploitability by attackers.

Consequently, the goals of this paper address two aspects. The first is to develop a comprehensive multi-level model of the Cloud operations to elaborate the lifecyle and the interaction of services involved in the Cloud ecosystem. We consider the Cloud operations in launching a Virtual Machine (VM) to (a) profile the behavioral characteristics of Cloud services and their interconnections, and (b) analyze the detailed flow of information among the services over varied Cloud levels to launch a VM. This information is utilized in our proposed approach to identify the base Cloud operational states, enumerating the "normal" sequence of Cloud operations along with the triggers that provide the state transitions. The Cloud model, developed using Petri Nets, forms the basis to identify multi-level vulnerabilities not recognizable by traditional/single-level threat analysis.

Consequently, the second paper aspect is to investigate the anomalous sequences of operations from varied attacker profiles e.g., insider/outsider attackers. On this background, the main contributions of the paper are the following:

1) Development of a comprehensive multi-level Cloud operational model and the rules to detail the information flow between services and transitions.
2) Development of an approach to analyze the behavioral properties of the services to a) characterize the baseline sequences of Cloud operations, and b) identify anomalous sequences over varied attacker profiles.

The paper is organized as follows. Section II models Cloud operations using Petri Nets with the primary focus on services interaction. In Section III, we validate our Cloud model and analyze malicious behaviors in the Cloud. Section IV reviews contemporary threat analysis approaches for the Cloud.

Research supported in part by EC H2020 MSCA-ITN NECS GA #675320 and ESCUDO-CLOUD GA #644579

II. Designing and Modeling of the Cloud

In this section, we model the Cloud operations by considering the essential services and their interaction in launching a Virtual Machine (VM). As the VM is an elemental component behind Cloud services, hence our focus is on the services involved in launching it. We surveyed multiple open source Cloud computing environments [6], [7] and identified the core services involved in the process of launching an instance of a VM in these environments. We profile the behavioral characteristics of the services with a focus on information flow and interactions among these services. We utilize the acquired information to develop our multi-level Cloud model using High Level Petri Nets (HLPN). The HLPN forms the basis to analyze behavioral and structural properties of the Cloud under normal operations and under varied attackers manipulations. The obtained model is shown in figure 1, with the description and data types of the HLPN places listed in Table I. Furthermore, we define the rules (pre/post conditions) that govern the flow of information among these places. These rules enable the firing of transitions and HLPN place token (information) from the input place(s) to the respective output place(s).

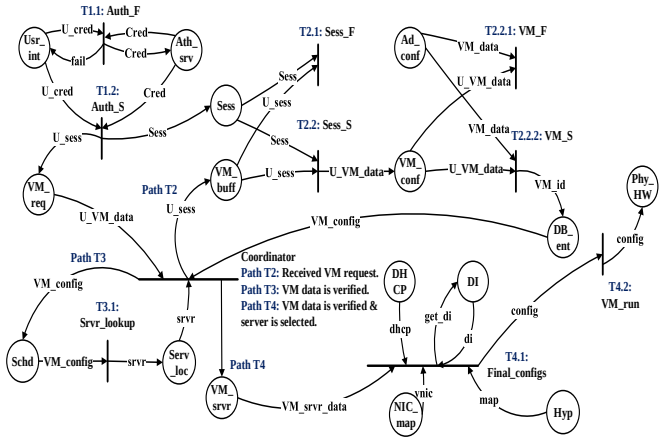

Fig. 1: Petri Nets based Cloud Model

TABLE I: Description and Data Type of the Places in Cloud Model

Place	Description	Data type
Usr_int	Interface to enter user credentials (user_namexPassword).	$String \times String$
Ath_srv	Authentication Service from the CSP.	$String \times String$
VM_req	VM requested instance (CPUxRAMxDisk).	$Int \times Int \times Int$
$Sess$	Session details of the customer (Sess_IDxSess_PolicyxSess_Exp).	$Int \times String \times String$
VM_buff	Temporarily stores VM request value and session details for verification.	$Sess \times VM\_req$
VM_conf	Holds VM data and configurations.	$Sess \times VM\_req$
Ad_conf	Holds quota and access policy of the customer and administrator configurations.	$String \times String \times String$
DB_ent	Initial entry of the VM in the database (VM_IDxVM_req).	$Int \times VM\_req$
$Schd$	Receives VM details to find a potential server.	DB_ent
$Serv\_loc$	Selection of the server for the VM (LocationxData_Center).	$String \times String$
VM_srvr	Receives server and VM details.	$Serv\_loc \times Schd$
Hyp	Receives all the configurations and launches the VM.	$DI \times DHCP \times NIC \times VM\_srvr$

Before describing the details of the model, we first overview the process of initiating a VM instance in the Cloud. The VM initiation begins with authenticating the customer (T1.1 in Figure 1 - top left). Following a successful authentication, the customer provides the desired set of properties for the VM (e.g., CPU, RAM, disk space) and thereafter, the scheduler service is invoked to determine a potential physical machine for the VM. Once the scheduler selects and provides server details to the coordinator, the coordinator then invokes multiple services to, assign disk image, initialize a virtual network interface card and assign MAC/IP addresses. These configurations are pushed onto the hypervisor which configures the VM instance accordingly and initiates the VM on the selected server.

In the following section, we detail the service interactions and model their behavioral characteristics by defining rules that govern the flow of information between the services.

A. Information flow in the Cloud

The process of initiating a VM follows a specific set of services that interact to successfully launch a VM. We define the rules that describe the information flow and processing of the information in the services to trigger the related transitions as shown in figure 1. As the request is initiated through authentication, hence a new token is generated each time the customer tries to log in via the Cloud interface. This initiates the execution of transitions in the path $T1$ and transitions $Auth\_F$ and $Auth\_S$ determine whether the customer's credentials are valid or invalid. These transitions are then mapped according to the rules shown in equations 1 and 2.

$$R(Auth\_F) = \forall cred \in Cred : U\_cred[1] \neq cred[1] \vee U\_cred[2] \neq cred[2] \vee U\_cred \neq cred \quad (1)$$

$$R(Auth\_S) = \forall cred \in Cred : U\_cred[1] = cred[1] \wedge U\_cred[2] = cred[2] \wedge U\_cred = cred \quad (2)$$

Equation 1 depicts a mismatch in the credentials provided by the customer and stored at the provider. Thus, the customer is again required to enter the correct credentials. The transition $Auth\_S$ is fired after the customer provides valid credentials. Consequently, a session is initiated and access privileges are granted to the customer based on the access policy and quota of the customer. The next place VM_req allows the customer to enter the desired VM data such as CPU, memory, and storage. These values are passed to the coordinator whose primary job is to manage and share respective information with the next place that satisfies the precondition(s). The conditions for next places and their description are shown in Table II.

TABLE II: Condition for Selecting Next Place from Coordinator

Next Place	Condition	Description
$T2/VM\_buff$	Transition $T1.2$ is fired.	VM data is available.
$T3/Schd$	Transitions $T2.2$ and $T2.2.1$ are fired.	Session is active and VM data is verified.
$T4/VM\_srvr$	Transitions $T2.2$, $T2.2.1$ and $T3.1$ are fired.	VM data is verified and the data center and server is selected.

Since the condition for path $T2$ is satisfied, the next place VM_buffer receives and stores VM and session details temporarily for the next transitions to validate the session and verify the VM properties. The rules for successful session and VM validation are shown in equations 3 and 4.

$$R(Sess\_S) = \forall sess \in Sess : U\_sess[1] = sess[1] \\ \wedge U\_sess[2] = sess[2] \wedge U\_sess[3] < sess[3] \quad (3)$$

Equation 3 is then fired if; (a) the session is active, and (b) the access policy of the customer matches the policy assigned by the administrator. Thus, a successful session verification leads to equation 4, which validates the VM configurations.

$$R(VM\_S) = \forall VM\_data \in VM\_data : U\_VM\_data[1] \\ \in VM\_data[1] \wedge U\_VM\_data[2] \in VM\_data[2] \\ \wedge U\_VM\_data[3] \in VM\_data[3] \quad (4)$$

Equation 4 successfully validates VM request by verifying access privileges for the requested VM and validating configuration of the requested VM with the quota of the customer. Thereafter, Eq 4 assigns a unique value for the identification (ID) of the VM request. This ID enables the rest of the services to distinguish between multiple VM requests from the same customer. The combined information of the VM and the session $(VM\_ID \cup U\_VM\_data \cup U\_Sess)$ is then written in the database.

The verification of the VM data enables the condition for path $T3$. Consequently, the scheduler service is invoked to find an appropriate data center location and an appropriate server. The selection of the server enables the second condition (cf., Table II) of the path $T4$, hence, multiple services are invoked with the requisite information to assign the varied configurations. The rule of Fi_Conf transition is shown in Equation 5 and the outcome of firing the transition is $config := (VM\_config \cup srvr \cup map)$.

$$R(Fi\_Conf) = \exists im \in DI : get\_di = im, \mid im \rightarrow map[1], \\ \mid \forall mac \exists vnic : ((mac \leftrightarrow dhcp[1]) \leftrightarrow map[2]) \quad (5)$$

The disk image service is responsible for providing a VM image from the repository. The network service initiates a virtual network interface card, assigns a MAC address and allocates a mapping between the virtual and the physical interfaces of the machine. DHCP is responsible to assign and manage IP address and a mapping between the IP and the MAC address of the virtual instance. These configurations are pushed onto the hypervisor, which then starts the VM on the physical hardware according to the received configurations.

These rules enable the flow of information in the Cloud ecosystem and describe the behavioral interactions of the services. Having established how the services interact with each other in the Cloud ecosystem, the next step is to validate our Cloud model and utilize the state space analysis to determine normal behaviors and to identify behavioral changes in the presence of a vulnerable service.

III. Validation of the Cloud Model

In this section, we validate the proposed Cloud model and also analyze the behavioral properties of the services using CPN tools [8]. These tools simulate and validate the HLPN to verify the model properties under a variety of conditions. One of the characteristic feature of the CPN tools is the ability to perform state space analysis of the given model. In the following sections, we demonstrate the effectiveness of CPN tools by developing a state transition schema consisting of all "normal" behaviors and state transitions of potential misbehaviors resulting from services manipulation.

A. State Space Analysis of the Cloud Model

We begin our behavioral analysis by simulating the Cloud model and validating the sequence of transitions under the rules described in Section II-A. These rules describe the benign interaction of services under normal Cloud operations. For example, equation 6 illustrates one of the possible correct sequence of operations.

$$R(crct\_op) = \exists U\_cred \in cred, \mid VM\_data \in Ad\_conf, \mid \\ im \in DI, \mid ((mac \leftrightarrow dhcp) \leftrightarrow im) \quad (6)$$

In order to demonstrate the comprehensive "all" sequences of operations, we utilize the state space analysis from the CPN tools. The analysis results in a large number of states and their interconnections, making illustrating the complete state diagram to be impractical. Therefore, we outline a partial state diagram of the model in figure 2[1] and analyze the structural and behavioral properties in this state diagram.

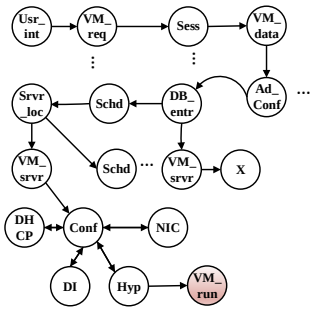

Fig. 2: Services behavior in Cloud IaaS

Figure 2[2] shows state transitions that are trivial to the normal Cloud operations and also the state transitions that are invalid and hence should not lead to a successful instantiation of a VM. As an example of the nefarious behavior for the sate DB_ent would be to interact directly with VM_srvr state and this interaction should not lead to further transitions. This prevents the state DB_entr to pass wrong information (e.g., server details and VM configurations) directly to VM_srvr under normal interactions.

However, in presence of a vulnerability or a service misconfiguration, anomalous behaviors may arise that are invalid but could still lead to the instantiation of the VM. To investigate this, we classify states according to their accessibility to the

[1]The CPN tools assign numeric numbers to the nodes in state space which makes the analysis and diagram difficult to comprehend. Thus, we map these numeric numbers to the respective state description of the IaaS model to make it understandable for the reader.

[2]Figures 2 and 3 are extracted from the Cloud model in Figure 1

attacker and partition state space accordingly. In this paper, due to space limitations, we investigate malicious behaviors resulting from an insider attacker manipulations.

B. Insider Attacker

An insider attack is orchestrated by entities responsible for managing the Cloud operations. They usually have elevated access to the services and also possess intimate knowledge of the infrastructure. This makes an insider attack have high potential for damage, and we analyze their potential misbehaviors from an insider perspective. Equation 7 depicts the interaction of misconfigured *Ad_Conf* with other services while figure 3 depicts this interactions graphically.

$$R(wr\_ad) = \exists U\_cred \in cred, | \, Ad\_conf[1] = VM\_data, | \, Ad\_conf[2] = Sess, | \, Ad\_conf \rightarrow VM\_srvr, | \, VM\_srvr \rightarrow Conf \quad (7)$$

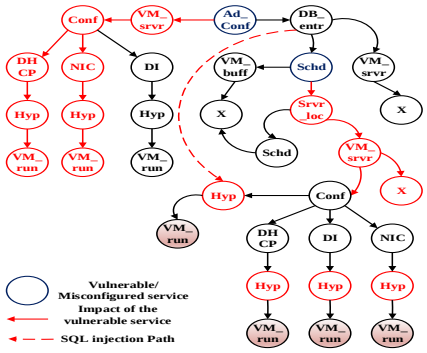

Fig. 3: Insider attacker state space

As evident from Figure 3, the behavior of the transitions changes when an insider attacker misconfigures the *Ad_Conf* service. The misconfiguration in the *Ad_Conf* leads to the state *VM_srvr* which violates the normal sequence as the *VM_srvr* must always be followed by *Srvr_loc*. The misconfigured *Ad_Conf* also affects *Conf* which is responsible for the VM configurations (i.e., mapping between the virtual and physical network interfaces, and assigning IP/MAC address).

The second service considered for analysis is the *Schd* service, responsible for finding an appropriate server for the requested VM. The misconfigured *Schd* service has a limited impact as it can only change the location of the server. The implication of selecting a wrong server is severe as it may fail to instantiate the requested VM. For completeness, a path in figure 3 is presented for the case of an SQL injection attack in which an insider writes malignant entries directly to the database. This results in skipping a number of transitions and directly pushing the maligned server and VM configurations onto the hypervisor which will then instantiate the VM with wrong configurations on an attacker controlled server.

IV. Related Work

Threat analysis is a well utilized approach to identify software/system threats. However, it is typically constrained to a specific configuration and for a specific attacker profile. The initial efforts led by Microsoft introduced a threat modeling approach called STRIDE [9]. It is an attacker-centric approach, applicable to data flow diagrams to find potential weaknesses and security flaws exploitable by a specific attacker. The approach presented in [10] develops an attack tree of the Cloud and utilizes a "what if" analysis to traverse paths in the tree to determine potential exploit(s). In [11], the authors formally analyzed open source Cloud environments for assessing correctness properties. In [5], the authors characterize vulnerabilities in the hypervisor by considering their impact on the functionality of selected popular hypervisors. The security issues of hypervisors was also analyzed by Tsai et al [3]. Their analysis focused on security issues over VM hopping and VM mobility. In [4], Bugiel et al., analyzed publicly available VM images in the Amazon EC2 repository. Their analysis focused on public interfaces to therein extract private information to launch attacks such as starting a botnet or launching an impersonation attack.

Our work differs from the above as we (a) model the influence of a vulnerable service on other services in the Cloud in a technology agnostic manner, and (b) analyze malicious behaviors for their impact on the full set of Cloud operations considering varied insider/outsider attacker profiles.

V. Conclusion

Our work explored multi-level Cloud attack surfaces by characterizing the fundamental information flows in Cloud services. To achieve this, we comprehensively analyzed Cloud operations to design and model varied levels of the operational stack and interfaces. The resultant Petri Nets model formed the basis to comprehensively identify the operational states for outlining the "normal" sequence of Cloud services, and to determine anomalous sequences of operations resulting from the vulnerable service interaction across the multi-level Cloud operational stack. Correspondingly, we enumerated the multi-level attack surface exploitability by attackers based on their accessibility of the service to explore attack surfaces across varied levels of the operational stack.

References

[1] F. Swiderski and W. Snyder, *Threat modeling*. Microsoft Press, 2004.

[2] S. Myagmar and et al, "Threat modeling as a basis for security requirements," *Proc. of SREIS*, pp. 1–8, 2005.

[3] H. Tsai and et al, "Threat as a service?: Virtualization's impact on cloud security," *IT Professional*, vol. 14, pp. 32–37, 2012.

[4] S. Bugiel and et al, "Amazonia: when elasticity snaps back," *Proc. of CCS*, pp. 389–400, 2011.

[5] D. Perez-Botero and et al, "Characterizing hypervisor vulnerabilities in cloud computing servers," *Proc. of SCC Cloud*, pp. 3–10, 2013.

[6] O. Sefraoui and et al, "Openstack: toward an open-source solution for cloud computing," *IJCA*, vol. 55, pp. 38–42, 2012.

[7] D. Nurmi and et al, "The eucalyptus open-source cloud-computing system," *Proc. of CCGRID*, pp. 124–131, 2009.

[8] K. Jensen and et al, "Coloured petri nets and cpn tools for modelling and validation of concurrent systems," *IJSTTT*, vol. 9, pp. 213–254, 2007.

[9] S. Hernan and et al, "Uncover security design flaws using the STRIDE approach," *MSDN Magazine*, Nov. 2006.

[10] P. Wang and et al, "Threat risk analysis for cloud security based on attack-defense trees," *Proc. of ICCM,*, pp. 106–111, 2012.

[11] S. Malik and et al, "Modeling and analysis of state-of-the-art vm-based cloud management platforms," *IEEE TCC*, vol. 1, pp. 50–63, 2013.