Enabling Multi-Layer Threat Analysis in Dynamic Cloud Environments

Salman Manzoor§, Antonios Gouglidis, Matthew Bradbury and Neeraj Suri
Lancaster University, UK
Email: {s.manzoor1, a.gouglidis, m.s.bradbury, neeraj.suri}@lancaster.ac.uk

Abstract—Most Threat Analysis (TA) techniques analyze threats to targeted assets (e.g., components, services) by considering static interconnections among them. However, in dynamic environments, e.g., the Cloud, resources can instantiate, migrate across physical hosts, or decommission to provide rapid resource elasticity to its users. Existing TA techniques are not capable of addressing such requirements. Moreover, complex multi-layer/multi-asset attacks on Cloud systems are increasing, e.g., the Equifax data breach; thus, TA approaches must be able to analyze them. This paper proposes ThreatPro, which supports dynamic interconnections and analysis of multi-layer attacks in the Cloud. ThreatPro facilitates threat analysis by developing a technology-agnostic information flow model, representing the Cloud's functionality through conditional transitions. The model establishes the basis to capture the multi-layer and dynamic interconnections during the life cycle of a Virtual Machine. ThreatPro contributes to (1) enabling the exploration of a threat's behavior and its propagation across the Cloud, and (2) assessing the security of the Cloud by analyzing the impact of multiple threats across various operational layers/assets. Using public information on threats from the National Vulnerability Database, we validate ThreatPro's capabilities, i.e., identify and trace actual Cloud attacks and speculatively postulate alternate potential attack paths.

Keywords—Cloud security; Cloud functional model; Threat analysis

1 Introduction

Cloud computing supports a variety of service models that offer elastic access to shared pools of resources (e.g., computational, storage, infrastructure) that are provisioned on-demand to meet user requirements. Cloud systems also entail the co-existence of both physical and virtual components that consequently result in a complex threat landscape. The overall effect is evident by the emergence of a diverse and increasing number of attacks and security breaches involving Cloud systems. A few recent examples include attacks that led to the leakage of users' confidential information [1] while other attacks have targeted the availability of the Cloud services [2].

To address security concerns in complex Cloud environments, multiple threat analysis approaches have been proposed that investigate threats at either a systems level [3], in the context of specific assets and technologies [4], or by exploring potential attack surfaces in the Cloud that could be used by attackers to violate security requirements [5]. Examples of asset-based schemes include, among others, threat analysis for evaluating cache side-channel attacks [6], analyzing network attacks [7], web attacks [8] or analyzing the impact of different threats on Cloud storage systems [9]. The alternate graphical model-based techniques, e.g., attack trees and graphs, have been applied to identify attack patterns that could potentially undermine the security of the Cloud. For instance, the authors in [10] developed a model of the Cloud data center and applied attack trees to identify potential paths leading to a security violation. Similarly, in [11], the authors proposed a security assessment methodology targeted specifically at the Cloud users.

1.1 Problem Space and Contributions

While the above-mentioned methodologies offer valuable insights into threat analysis, it is essential to recognize their focus areas and assumptions, i.e., they target identifying threats within individual assets or make assumptions regarding the static nature of interconnections among the assets. While this observation highlights the practical context in which these methodologies operate, it also provides a pointer to their scope and constraints. Specifically, the assumption of static interconnections hinders their effective applicability to Cloud environments, which are inherently dynamic in nature given their support for on-demand adaptive resource provisioning. Furthermore, the limited capabilities of contemporary analysis techniques in incorporating user- and service-specific security requirements within the Cloud threat model leads to incomplete security analyses. Examples of user-specific requirements may stem from service level agreements, authentication and enforcing access control restrictions (e.g., time, location). Similarly, service-specific requirements (e.g., content delivery service) can be set by the application domain (e.g., finance, medical) and result in prioritizing availability over confidentiality. Hence, a threat analysis process is desired that considers the incorporation and prioritization of user-level and service-level security requirements.

§. The research was conducted when the author was affiliated with Lancaster University.

To address these challenges, we propose ThreatPro, a novel threat analysis methodology capable of modeling both the dynamic environment of the Cloud and the security requirements of a user. ThreatPro facilitates Cloud service providers to evaluate the consequence of actual or speculative threats and their progression across the system under a dynamic configuration irrespective of the underlying technologies; and to analyze the impact of multiple threats across different operational layers and services in the Cloud for specific security requirements. As with similar solutions [12], [10], [13], ThreatPro also enables the users to define the scope of their system and the threats to the system. It means that the users will need to decide at what level of abstraction to describe their Cloud system and the types of threats to analyze.

Additionally, to develop a threat analysis methodology that is technology-agnostic, ThreatPro proposes a new information flow [14][1] based model to abstractly capture the functional behavior of the Cloud. This is accomplished by defining a set of transitions and a rule-set specifying the conditions for executing the transitions. In contrast to existing models [10], [5], [16], we emphasize on the interconnection of services and the flow of information rather than performance and computing measurements. Furthermore, we specify rules prescribing the behavior of a threat as additional constraints to the transitions to determine the implication of the threat. By tracing the sequence of transitions, we can not only model the propagation of threats but can also simulate speculative scenarios. Overall, the main contributions of ThreatPro are:

- A Cloud model capable of representing the fundamental operations of a Cloud. This is achieved by abstracting the essential services from real-world Cloud deployments [Section 5].
- A technology-agnostic information flow model based on the Cloud model. The model converts service interactions to a set of rule-based transitions to represent the functional behavior of the Cloud [Section 6]. The rule-based transitions are capable of facilitating the specification and analysis of the information flow models.
- A path-illustrative approach to profile the flow of threats and analyze their impact on targeted services and the propagation of threats across the multiple layers of the Cloud. This assists in identifying paths that lead to the violation of the security requirements, i.e., an attack on the system [Section 7].

1.2 Paper Organization

The remainder of the paper is organized as follows: Section 2 reviews contemporary threat analysis approaches for the Cloud. A progressive overview of ThreatPro's three building blocks is presented in Section 3. In Section 4, the first block of ThreatPro is presented, i.e., services abstraction to represent the functional behavior of the Cloud. In Section 5, the second block of ThreatPro is presented that translates the abstract Cloud model into an information flow model to represent the functional behavior of the Cloud operations. Section 6 concatenates these building blocks to develop the overall threat analysis process including the approach to perform speculative analysis. Section 7 validates the capability of ThreatPro to trace and analyze real-world attacks. Section 8 discusses ThreatPro's capabilities for predictive analysis, its potential for the plug-and-play services, remarks on the automation of the modelling process, and the limitations of this approach, and concluding remarks are presented in Section 9.

1. By information flow we encapsulate system execution and the flow of information between components within a system. This differs from data flow [15], which specifically focuses on which data is transferred between different system components.

2 Related Work

We now provide an overview of contemporary threat analysis approaches. For simplicity, the approaches are broadly categorized into (1) *asset-based techniques* — used to explore potential threats in specific assets, and (2) *graphical security models* — used to identify potential attack paths leading to a security requirement violation.

2.1 Asset-based Threat Analysis

Asset-based TA aims to uncover threats and their impact on discrete assets (e.g., components, services, interfaces, data) typically without factoring in operational considerations. Some recent works have demonstrated the value of TA in evaluating cache side-channel attacks [6] to explore the possibility of using the cache to compromise the confidentiality of tenants hosted on the same physical machine. A number of TA approaches exist that target specific technologies. For example, the authors analyze the impact of different threats in Cloud brokerage systems in [9]. On the other hand, the application of model checking to verify the violation of security property has been demonstrated in [7]. The primary objective was to analyze network attacks violating the defined security property. Similarly, modeling the behavior of an application and applying probabilistic model checking to investigate the impact of elasticity on security requirements was investigated in [16]. Furthermore, the outcome of the analysis can be used as feedback to fine-tune the behavior of the Cloud for governing its elasticity. A risk assessment approach is proposed in [17] for access control mechanisms in the Cloud. The objective was to show the effectiveness of role-based access control on the risk assessment of the asset.

These schemes either investigate specific hardware vulnerabilities in their evaluation [6] or consider specific systems (e.g., CloudRAID) in their assessment [9]. Similarly, characteristics of the Cloud operations are studied in to analyze the interplay between elasticity and security, such as data loss or data leakage [16]. However, this analysis is limited to only the elasticity aspect of the Cloud.

2.2 Graphical Security Models

Multiple graphical security models have been developed to visually trace and identify attack paths and patterns that could potentially undermine the security of the Cloud. Primarily, these have been in the form of attack trees and attack graphs. Modelling a Cloud data center and applying attack trees to identify potential paths have been investigated in [10]. Similarly, the quantification of the user's security requirements is proposed in [18]. A risk assessment framework for a sensor environment deployed in the Cloud was presented in [19]. The objective was to illustrate the cause-effect relationship and apply security measures that minimize the attack's impact. On the other hand, concepts from requirement engineering have been utilized in [20] to propose a methodological approach to elicit a user's security and privacy requirements and select the appropriate Cloud provider. The approach performs a cost-benefit analysis for the users thereby enabling them to make an informed decision about migrating to the Cloud.

The application of attack and defense trees has been detailed in [21]. The approach investigated the interplay between attacks and the respective countermeasures and proposed a framework to assess the associated risks of the applied countermeasures. The work in [22] proposed a graphical security model using Bayesian attack graphs to quantify the likelihood of the network compromise which feeds into an attack mitigation plan. This enables system administrators to make an informed decision by considering the trade-off between the attack and the mitigation strategy. A reference model of the Cloud incorporating the security controls and best practices was developed in [23] to assess the security posture of the Cloud offerings for confidentiality and integrity. This was achieved by estimating probabilities of advanced persistent threat infiltration in the Cloud. The underlying technique utilized a Bayesian network model that examines attack paths and assesses their impact on both confidentiality and integrity requirements.

In [24], the authors address the limitation of attack graphs that are restricted to a snapshot at the current time by developing a time-independent model. Therefore, the model can be utilized irrespective of the state of the network. The fundamental premise behind the model is the analysis of the security of multiple network states considering the time duration of each network state and the visibility of the network components to create an attack graph that is representative of the network without limiting it to a specific time window. Similarly, in [25], the authors leverage the software-defined network technology to develop an attack graph-based that exhibits dynamic behaviour by incorporating a moving target defence technique. The model relies on shuffling a host's network configurations (e.g., MAC/IP/port addresses) when an alert has been generated that identifies the presence of an attacker in the system. The shuffling is performed to reduce the likelihood of lateral movement from the compromised network node.

Overall, these schemes leverage attack graphs/trees to explore potential paths that identify a security violation. Furthermore, quantifying the risks associated with each path is fundamental to many of these schemes, enabling system administrators to prioritize the paths and the mitigation strategy accordingly. On the other hand, these schemes assume that the attack paths are static and the functional behavior does not create new interconnections at run-time. This assumption does not hold in the inherently dynamic Cloud environment, where new interconnections might be introduced at run-time through VM migration or by instantiating a new VM.

2.3 Synopsis

As identified in Sections 2.1 and 2.2, both asset-based TA and graphical security models are effective TA techniques. However, their effectiveness is limited in analyzing threats considering the holistic view of the Cloud's dynamic operations. For instance, asset-based schemes consider assets in isolation without operational factors and reveal threats pertinent to the specific asset. On the other hand, graphical models assume that the interconnection among assets is static and lacks the capability to analyze threats in a dynamic environment. In this paper, we propose ThreatPro that can incorporate (1) the asset's operational environment, (2) dynamic interconnections across resources/services, and (3) specification of the user's security requirements, to provide a comprehensive threat analysis process applicable to the Cloud.

3 Building Blocks of ThreatPro

The ThreatPro methodology is developed as a progression of three building blocks (i.e., functional Cloud model, information flow model and threat analysis) as depicted in Figure 1. In the following, we overview each of these blocks prior to detailing their operations in Sections 4, 5 and 6 respectively.

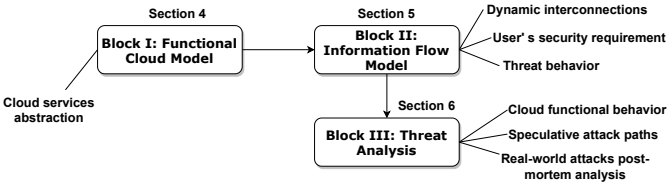

Fig. 1: Blocks of ThreatPro

3.1 Block I: Functional Cloud Model

A number of delivery models exist for the Cloud, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), primarily emphasizing the functionality and performance in these models. Furthermore, a considerable body of research exists for modeling and analyzing the behavior of an application in the Cloud [26], [27], [28]. However, ascertaining threat propagation requires modeling the functional

behavior of the Cloud to capture the interaction across services, and investigating the interplay between the services' interactions and the threat progression. Despite that, work related to modeling the Cloud functionality is very limited. Among the primary functions of the Cloud IaaS, is offering and managing virtual resources as VMs [29], [30]. These VMs are created through virtualization technology, an enabling technology to share a physical host with the VMs. [31]. Thus, we define an abstract model for the Cloud emphasizing the interactions of services during the life-cycle of a VM [32]. Generally, the main stages of a VM's life-cycle are VM creation, storage assignment, server selection for deployment, VM execution, and VM deletion. Furthermore, VM migration and VM snapshot may occur during its life-cycle. The service interactions during the life-cycle of a VM are conceptualized after surveying multiple open-source Cloud computing environments [33], [34] and Cloud deployments adopted by market leaders such as Amazon, Google, and Microsoft. The model, depicted in Figure 2, exhibits a 3-layer architecture of the Cloud consisting of the control layer, infrastructure layer and storage layer, where each layer performs distinct functions. The model is flexible and can be extended to include vendor-specific services at each layer. In this paper, we focus the modelling on the functionality of launching a VM as it is a fundamental offering of the Cloud.

3.2 Block II: Information Flow Model

The second building block of ThreatPro is a technology-agnostic information flow model [14] of the Cloud operations. This entails abstracting the technology and vendor-specific characteristics to create a transition system governed by rules that trigger transitions following the fulfillment of the respective preconditions. For example, the authentication credentials provided by the user are a precondition to trigger different transitions depending on the validity of the credentials irrespective of the underlying authentication technology used to check these credentials. In the case of valid credentials, a user is directed to a dashboard/interface to access their VMs. On the other hand, invalid credentials lead to an error message, and the user is requested to reenter credentials. Thus, defining the pre-conditions and rules that govern the triggering of transitions and passing of the information among the services represent the functional behavior of the Cloud. Furthermore, we incorporate the security requirements of the users in the information flow model to support the prioritization of threats that violate specific requirements. We argue that the security requirement of an application varies depending on the functionality of the application. For example, a content delivery application might set the availability of the data as a high priority while an application dealing with financial records might consider confidentiality as its primary requirement. Therefore, considering such security requirements is critical since it helps to identify threats that may lead to their violation.

3.3 Block III: Threat Analysis

The third block of ThreatPro assesses the impact of threats to Cloud services. We assess the impact of multiple threats at different levels of abstraction, e.g. considering threats at multiple services/layers and the possibility of a threat's combination to violate a security requirement of the user. Furthermore, we investigate the progression of a threat in the Cloud's dynamic environment where resources migrate from one physical host to another or new resources can be instantiated. ThreatPro is also able to perform a speculative analysis to examine the potential of a threat to compromise a security requirement. Following this overview, the subsequent Sections 4, 5 and 6 detail each constituent block of ThreatPro to result in a holistic threat propagation analysis process for the Cloud.

3.4 Threat Model

Given these models, it is also necessary to consider the threats that are intended to be analysed by ThreatPro. In this paper we focus on modelling attacks on confidentiality and availability in cloud systems. Adversaries may have a variety of different goals [35], however, due to ThreatPro facilitating exploring a broad range of threats we do not define specific adversary goals. Instead in Section 7 we will look at case studies of exfiltrating sensitive data and resource exhaustion to deny availability. The key is that the attacks of interest can be modelled in conjunction with ThreatPro's functional and information flow models.

4 ThreatPro's Block I: Defining the Functional Model of the Cloud

Following the overview in Section 3.1, this section details the first block of ThreatPro, i.e., how to represent the Cloud's functional behavior as a model. The reasons for developing such a model are twofold. Specifically, there is a lack of both (1) a generalized Cloud model applicable to the spectrum of Cloud offerings, and (2) approaches that can analyze the interplay between the functional behavior of the Cloud and the attack paths. In order to develop such a model, we first extracted common services from multiple open-source Cloud computing environments [34], [33] and major stakeholders in the Cloud market, such as Amazon, Microsoft and Google. There are obvious differences in terms of the Cloud architecture and network configurations adopted by each vendor. For instance, the controller node could be distributed across the data center. However, these differences are technology and optimization-driven and consequently beyond the scope of this paper.

The Cloud model presented in Figure 2 depicts a generalized 3-layered (Control, Infrastructure and Storage) architecture focusing specifically on the Cloud's functionality to be agnostic to the technologies implementing the functionality. Each demarcated layer performs a

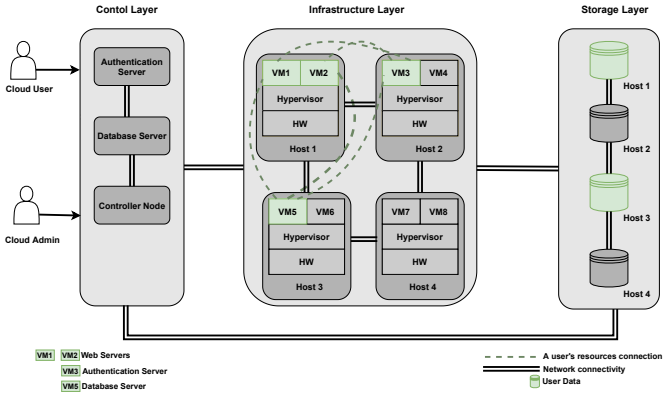

Fig. 2: Multi-layer architecture of the Cloud

specific task in the life cycle of a VM. The role of the control layer is to authenticate users and enable them to request new VMs. The infrastructure layer receives the request, creates the respective VM, and links it with the existing resources of the user. The storage layer provides storage capabilities for the data. We provide details of each layer's functionality in the following sections.

Control Layer: it consists of an authentication server, database server and a controller node, orchestrates the managing and scheduling of the Cloud resources — physical and services — for the Cloud administrator and the users.

Infrastructure Layer: As the name suggests, this layer represents the actual physical hardware of the Cloud for binding the VMs to physical hosts. The core functionality of the layer is provided by a hypervisor [36] that runs on top of the hardware/OS along with other VM management tools.

Storage Layer: This layer provides storage capacity and delivers data when requested. This layer is also responsible for providing consistency among different data backups. As the placement of the VMs across different hosts is permitted, the data could also be distributed across different hosts.

These 3 layers collectively outline the operations of any generalized Cloud system. As VM management (creation, migrations and deletion cf. Section 3.1) is the basic Cloud functionality, ThreatPro utilizes a VM-centric approach for threat propagation and analysis. In the following, we focus on the operations involved in creating a VM to illustrate the information flows across the operational layers of the Cloud prior to building ThreatPro's information flow model in Section 5.

4.1 Information Flow in Launching a VM

As mentioned, the authentication service is the user's interface to the Cloud. A user can only launch or request a VM after being successfully authenticated. The details of subsequent transitions at each layer are as follows:

Control layer transitions: Once authenticated, a user is transferred to a dashboard presenting the allocated VMs and the possibility of requesting additional VMs. If the user decides to launch a new VM, the requested VM configurations (e.g., CPU, RAM) are compared with the assigned quota. A valid request leads to the invocation of the scheduler service that determines a potential host for the requested VM. The VM configuration and the selected host are then passed to the infrastructure layer.

Infrastructure layer transitions: The infrastructure layer receives the VM request and invokes image repository service for the operating system and the network service for the networking capabilities (e.g., Virtual Network Interface Card (VNIC), IP addresses). Furthermore, the infrastructure layer interfaces with the storage service for allocating storage for the VM.

Storage layer transitions: The primary responsibilities of the storage service are assigning storage to the VM and keeping the data among the backups consistent. This step is optional in case the user does not select the storage capacity for the VM.

VM: After the configuration is finalized, the hypervisor instantiates the VM and it is added to the database against the corresponding user.

The aforementioned is an overview of the services interaction to create a new VM. It should be noted that the Cloud provider can initiate the VM instantiation or migration to optimize the workload without the user's input but in compliance with the Service Level Agreement (SLA) signed between the user and the Cloud Service Provider (CSP). The next section translates this model into an information flow model that focuses on the service interaction and the flow of information among the services.

5 ThreatPro's Block II: Defining the Information Flow model

Following on the overview from Section 3.2, this section details the second building block of the ThreatPro methodology, i.e., the development of an information flow model of the Cloud. Requirements for the information flow model are the following: (1) the model should support expressing the functional behavior of the Cloud as well as the threats in a technology-agnostic style, and (2) there should be the ability to identify violations from the sequence of events by determining the modifications in the operations of the Cloud caused by spurious input to the system.

These specifications are achieved by defining rules and constraints that determine the triggering of transitions after their respective preconditions have been fulfilled. Consequently, we begin with a basic transition system representing functional behavior and rules determining the states' transition. Subsequently, we leverage the rule-based transition system to represent a login system for user authentication and eventually represent the Cloud functional behavior. Furthermore, we express a threat's behavior as an instantiation of the rule-based transition system to use as spurious input to the system.

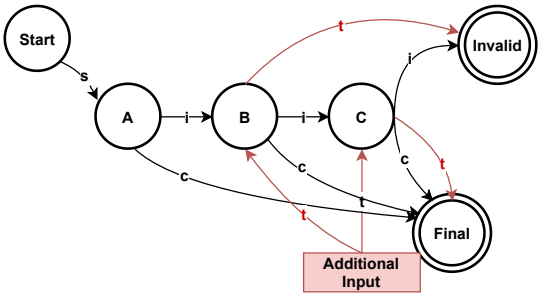

Fig. 3: An abstract example of a transition system

5.1 A Basic Transition System

Figure 3 presents an example transition system to demonstrate how a system's functionality can be represented. The received input at each state, depicted on the arcs, enables transitioning between the states. The transition system forms the basis for analyzing the proper functioning of the system and provides the capability to identify modifications in system actions caused by spurious inputs. The creation of such transitions is imperative for Cloud systems. Although it is possible to generate and reuse a library of profiles of transition systems, it remains a limitation that an arbitrary number of system profiles may be required in practice. In the following, we do not delve into the process of automating the creation, storage and utilization of these profiles. Such a mechanism warrants a dedicated consideration beyond the scope of this work. Instead, we elaborate on how such a transition system can be used. We now describe the rules for transitioning between states which eventually lead to a terminal state (`Final` or `Invalid`).

5.2 Normal Behavior

There are multiple paths that represent the normal operation of the system. Any modification in these paths might be considered a threat to the system.

- Path 1: $\texttt{Start} \xrightarrow{s} \texttt{A} \xrightarrow{c} \texttt{Final}$
- Path 2: $\texttt{Start} \xrightarrow{s} \texttt{A} \xrightarrow{i} \texttt{B} \xrightarrow{c} \texttt{Final}$
- Path 3: $\texttt{Start} \xrightarrow{s} \texttt{A} \xrightarrow{i} \texttt{B} \xrightarrow{i} \texttt{C} \xrightarrow{c} \texttt{Final}$
- Path 4: $\texttt{Start} \xrightarrow{s} \texttt{A} \xrightarrow{i} \texttt{B} \xrightarrow{i} \texttt{C} \xrightarrow{i} \texttt{Invalid}$

Paths 1, 2, 3 and 4 demonstrate the correct functional behavior of the transition system, i.e., the paths start from the state `Start` and terminate to either the `Invalid` or the `Final` state. The inputs start, invalid, and correct are respectively denoted by $\{s, i, c\}$ and are used to trigger different paths depending on the input provided to the system. For instance, in path 1, an input triggers the state `Start` which passes on s as information to state `A`. The received input initiates multiple paths from state `A`, for instance, the input corresponding to a correct value c leads to the `Final` state. Conversely, an invalid input i at state `A` moves the system to state `B` and the same process is followed at state `B`. However, at state `C`, an invalid input i terminates the system at the invalid state.

5.3 Incorporating Malicious Inputs to the System

The rules determine the functional behavior despite the different underlying technologies. The rules can be added (or removed) to introduce new (or speculative) specifications or constraints from users/systems. In Figure 3, additional inputs are introduced to both states `B` and `C` to analyze their corresponding impacts on the behavior of the system. For example, at state `B`, an input t can modify the state and result in transitioning subsequently to an invalid state instead of the state `C` or `Final`. Thus, a rule-based transition system highlights manipulation caused by malicious inputs and enables the speculative (what-if) analysis. The complete paths for both the malicious input are given below.

- Path M1: $\texttt{Start} \xrightarrow{s} \texttt{A} \xrightarrow{i} \texttt{B} \xrightarrow{t} \texttt{Invalid}$
- Path M2: $\texttt{Start} \xrightarrow{s} \texttt{A} \xrightarrow{i} \texttt{B} \xrightarrow{i} \texttt{C} \xrightarrow{t} \texttt{Final}$

5.4 Representing a Transition System

We have demonstrated the benefits of using a rule-based transition system to enumerate the behavior of a system and to speculate on the behavior by adding spurious constraints. We leverage this rule-based transition system concept to develop an information flow model of the Cloud depicting its functionality. There exist multiple methods to model the functionality of a system. In the following, we detail two prominent alternatives of labelled transition systems and Petri nets.

5.4.1 Labelled Transition System (LTS)

LTS has been extensively applied to model the Cloud operations, including the modeling of client-Cloud interactions [37], [38], [39]. The benefit of using such models is to elaborate the behavior of a system and identify a potential violation of the specified property using a model checker. To this end, the complete model and the property specification are provided to a model checker that generates a counterexample identifying the property violation. The specified property is often a safety/liveness property, but the process can be replicated for specific security properties. On the other hand, LTS becomes cumbersome for concurrent systems due to the state explosion problem [40]. Further, the states and the associated actions in LTS are global, i.e., the complete state information is required to recognize the firing of a transition. A state cannot be distributed into multiple local states with different preconditions to trigger a transition locally if a certain precondition is satisfied. Moreover, these models are deterministic, while modeling the Cloud requires triggering of transitions at certain time intervals to replicate e.g., VM migration.

5.4.2 Petri nets

An alternative to an LTS are Petri nets, which can describe the functional behavior of distributed systems. Petri nets have been used to model the workflow of concurrent systems [41], resource management in

the Cloud [42], and fault detection in distributed systems [43]. A difference between Petri nets and LTSs are that the states can be distributed locally as places in the former enabling them to hold different information required for a transition. Moreover, the transitions are fired locally and non-deterministically without requiring a global view of the system. Furthermore, the Petri nets support time-driven firing of the transitions, i.e., firing the transition at a specific time instance. Similar to LTS, Petri nets also encounter the issue of state explosion [40].

5.5 ThreatPro's Requirements

We have described the possible options for modeling the behavior of a system, and now we proceed to elicit the specific requirements for modeling the Cloud. The Cloud is a distributed and concurrent system, and modeling its functional behavior entails assigning information to each place[2] and passing on either a complete or a subset of information according to the triggering event. Furthermore, certain events might create an impact both locally and globally. For example, a threat targeting a service affects that service, but can also progressively target the interlinked services. On the other hand, performing a speculative analysis requires assigning constraints (threats preconditions) to different services to analyze their consequence on the benign operation of the Cloud. An additional requirement is the capability to model time-driven events. For example, a VM can instantiate, decommission or migrate at run-time according to the workload. These requirements favor the use of Petri nets to model the information flow. A brief overview of Petri nets is presented before demonstrating its use in developing the information flow model of the Cloud.

A typical Petri net has two elements, places and transitions[3], depicted as circles and bars, respectively, as shown in Figure 4. A transition signifies the occurrence of an event and the place holds the tokens (information) that enables the transition. The conditions that govern the flow of tokens are represented on the arcs between input and output places. The pre-conditions are represented on the arcs that connect places to transitions and the output flow (post-condition) from a transition governs the flow of token (information). A transition is fired only if both pre- and post-conditions are satisfied. A token from an input place is transferred onto the respective output place after the transition is triggered.

In this paper, we use a variant of Petri nets called High-Level Petri nets (HLPN) [44], which provide further flexibility in assigning multiple tokens of different data types to a place. Moreover, in HLPN, a subset of the token (information) can be passed onto the next place depending on the triggering condition. For example,

2. A place holds the token in Petri nets, in other words, places are comparable to states in transition system.

3. We use three different fonts to make it clear what type of item within a Petri net is being referred to. These are: a `Place` in the Petri net, an Input provided, and a TRANSITION that can be taken.

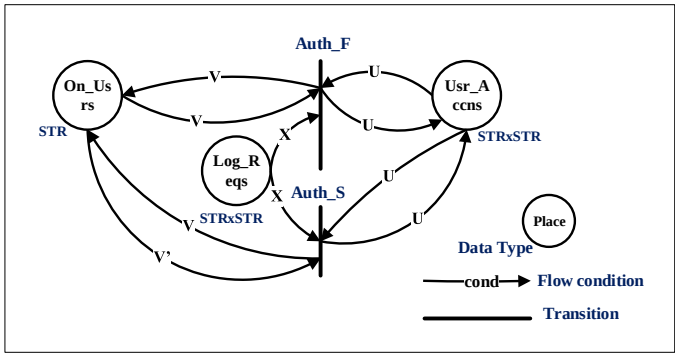

Fig. 4: Login system using HLPN

TABLE 1: Description of Places in Figure 4

Place	Description	Mapping
$\phi(Log\_Reqs)$	Login credentials	$\mathbb{P}(Usernames \times Passwords)$
$\phi(Usr\_Accns)$	Sever credentials	$\mathbb{P}(Usernames \times Passwords)$
$\phi(On\_Usrs)$	Online Users	$\mathbb{P}(Usernames)$

the authentication service holds both usernames and passwords and passes on only the username to the next place that provides a list of the user's existing VMs. Furthermore, the constraint can be time-driven. For instance, after a certain time interval, a VM migration process can start requiring a new VM instance creation and the model needs to capture such dynamic interconnections. These dynamic interconnections are captured in the model through time-driven firing of the transition. Moreover, the transitions are fired locally without contemplating the global state of the system. This enables the model to capture new VM instances requested during the VM run place or concurrent VM requests from the same user.

5.6 Instantiation of the Cloud Login System

In the previous section, we have explained a basic transition system and rules that determine the functional behavior of the system through the flow of information among the states. We also described the advantages of using HLPN for the development of the information flow model. This section leverages the rule-based transition system to create an authentication system for the Cloud before translating the complete Cloud model (cf., Figure 2). This authentication system is shown in Figure 4, where there are three places (`Log_Reqs`, `Usr_Accns` and `On_Usrs`) described in Table 1 and two transitions (AUTH_F, AUTH_S). The transition AUTH_F represents failed authentication due to invalid credentials, while AUTH_S depicts a successful authentication. The firing of these transitions follows the rules in Equations (1) and (2). The action taken when the predicate in Equation (1) holds is shown in Equation (3). This is represented in Guarded Command Language [45] in the form Name ::= Predicate $\rightarrow$ Statements, where the list of statements are executed when the guard predicate holds.

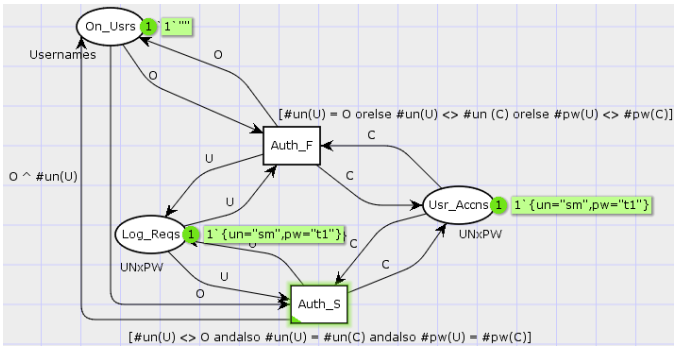

Fig. 5: Snippet of CPN tools of the Login system

$$R(\text{AUTH\_S}) = \exists u \in \mathsf{U} : u \in \mathsf{C} \wedge u.\text{username} \notin \mathsf{O} \quad (1)$$

$$R(\text{AUTH\_F}) = \forall u \in \mathsf{U} : u \notin \mathsf{C} \vee u.\text{username} \in \mathsf{O} \quad (2)$$

$$\text{SuccessfulLogin} ::= R(\text{AUTH\_S}) \rightarrow O' := \mathsf{O} \cup \{u.\text{username}\} \quad (3)$$

The transition AUTH_S in Figure 4 is fired if the necessary preconditions are fulfilled, i.e., the username and password provided by the user match the username and password stored at the user accounts and the user is not already online. These preconditions are represented on the arcs using: (i) the set of users U attempting to log in, where $\forall u \in U : u = (u.\text{username}, u.\text{password})$ represents the username $u.\text{username}$ and password $u.\text{password}$ provided by a user, (ii) the set C of credentials known to the server, where $\forall c \in C : c = (c.\text{username}, c.\text{password})$ represents the username $c.\text{username}$ and password $c.\text{password}$ known by the server, and (iii) set O represents the usernames that are already online. A successful authentication of the user transfers them to the list of online users by adding the new user to the set O, for which the updated set is denoted by O'. On the other hand, a violation in any of the conditions results in the firing of the transition AUTH_F instead. The predicate $R(\text{T})$ denotes if a specific transition T is taken. We show the implementation of these predicates in Listing 1 of our supplemental material which was performed using CPN tools [46][4].

Figure 5 shows a snippet of the CPN tools after defining the places, transitions and the guards to the respective transitions. For instance, the place `ON_Usrs` holds the users that are online and currently it is empty. The `Log_Reqs` currently has a single token (information) with the username `"sm"` and password `"t1"`. This is compared against the stored credentials at `Usr_Accns`. Therefore, the data type of both the places is `UNxPW`. A `place` can hold multiple tokens and the green circle shows the exact number of tokens the place currently holds. To distinguish tokens from each other, a separator `++` is used in the CPN tools. The AUTH_S is highlighted to indicate that the transition is enabled. In Petri nets the transitions are enabled after all the input places to the transition have at least one token but the transition is only fired after both the transition guard and the output condition of the transition are satisfied. The firing results in taking the respective tokens from the input places and adding them to the output places in compliance with the output condition. A weightage can be assigned to the output condition which then determines the number of tokens moved from the input places. Furthermore, a timing delay can also be applied to the transition which would restrict the firing of the transition until the assigned time period has elapsed. In the case of AUTH_S, the transition guard is to match credentials and the output condition is to add the user to the `On_Usrs` place. Once these conditions are fulfilled, the user becomes online and is added to `On_Usrs`.

It is evident that rules-based information flow is independent of the underlying technology since any appropriate technology could be used to validate credentials. The subsequent section expands the authentication system by introducing additional Cloud functionality and eventually representing the Cloud behavior using HLPN. Consequently, the resulting information flow model is agnostic to specific underpinning technologies.

5.7 Instantiation of the Cloud Functional Behavior

We extend the authentication system by adding additional services from the Cloud model (cf., Figure 2) and eventually, translating the Cloud model to an HLPN model which is shown in Figure 6. The description of places and their data types are mentioned in Table 2. The function $domain(\text{V})$ takes a HLPN place `V` and returns the set of all possible values that `V` could have.

We revisit the instantiation of the VM from the perspective of creating rules to govern the flow of information among the services and replicating the functional behavior of the Cloud.

1) Transitions T1.1a/T1.1b/T1.2 determine credential validity. A successful authentication leads to a dashboard enabling the user to access their existing VMs.
2) Transitions T1.3a/T1.3b are triggered after a user initiates the VM creation process and provides properties for the VM (e.g., CPU, RAM, disk space). These properties are checked for compliance with the associated quota of the user.
3) Transition T1.4 is fired after the scheduler service determines a potential data center and a host to run the requested VM.
4) Transition T1.5 is triggered after multiple services provide the respective tokens (information). For example, a disk image is provided from the repository and the network service initializes a virtual network interface card and assigns MAC/IP addresses. These configurations are pushed onto the hypervisor configuring the VM instance accordingly.
5) Transition T1.6 is fired after it receives the final configuration and the VM has started executing successfully. The `VM` place in Figure 6 shows the terminating place of the Cloud model.

4. Each of the following Petri net models were implemented using CPN tools and the implementation can be found at https://github.com/salman-manzoor/Threatpro.

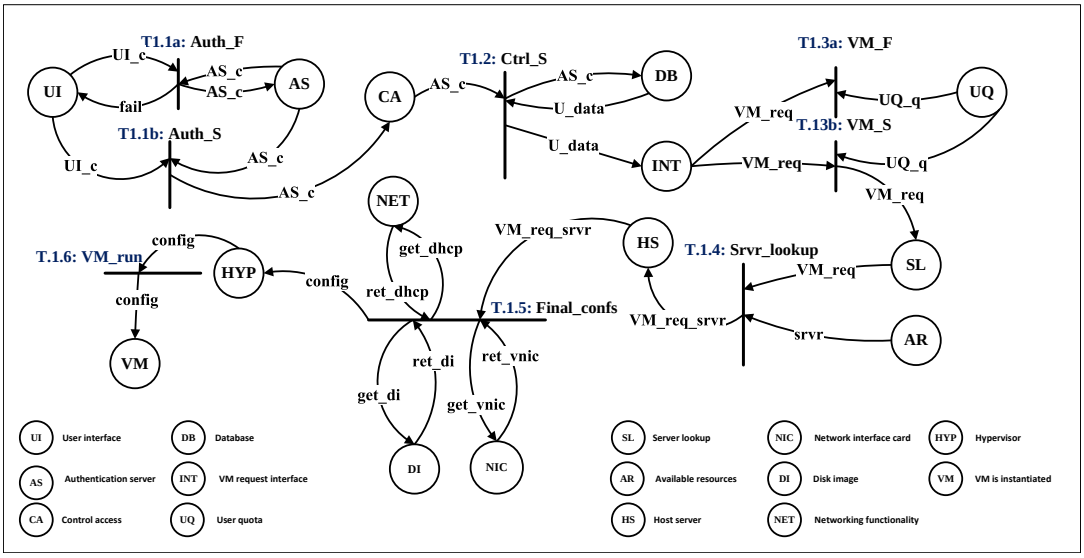

Fig. 6: Transforming Cloud Model to HLPN

TABLE 2: Description of Places in the Cloud Model

Place	Description	Domain
UI	Interface to enter credentials	$\mathbb{P}(Usernames \times Passwords)$
AS	Authentication server storing credentials	$\mathbb{P}(Usernames \times Passwords)$
CA	Access restrictions	$\mathbb{P}(Usernames)$
DB	Stored list of VMs	$\mathbb{P}(Usernames \times VMs)$
INT	Interface to run VMs	$\mathbb{P}(Username \times CPU \times RAM \times Disk \times Arr)$
UQ	Users quota and configurations	$\mathbb{P}(Username \times CPU \times RAM \times Disk)$
SL	Potential server for the VM request	$\mathbb{P}(Usernames \times CPU \times RAM \times Disk)$
AR	Available resources to launch the requested VM	$\mathbb{P}(Loc \times DC)$
HS	Receives hosting server and VM config.	$\mathbb{P}(Loc \times DC \times Usernames \times CPU \times RAM \times Disk)$
NIC	MAC address	MAC
NET	Assigns dynamic IP	$\mathbb{P}(IP \times MAC)$
DI	Holds VM disk image	$\mathbb{P}(DI)$
HYP	Receives configuration and launches the VM	$\mathbb{P}(CPU \times RAM \times Disk \times IP \times MAC \times DI)$
VM	VM is started on the server	$\mathbb{P}(Loc \times DC \times Usernames \times CPU \times RAM \times Disk \times DI \times IP \times MAC)$

We define rules that govern the flow of tokens (information) from input to output places. A new token is generated each time a user tries to login triggering transitions AUTH_F and AUTH_S to determine the validity of the user's credentials, UI_c is the set of provided credentials and AS_c is the set of credentials stored at the server. These credentials are used in Equations (4) and (5) to check the validity of the user's credentials.

$$R(\text{AUTH\_F}) = \forall u \in \mathsf{UI\_c} : u \notin \mathsf{AS\_c} \tag{4}$$

$$R(\text{AUTH\_S}) = \exists u \in \mathsf{UI\_c} : u \in \mathsf{AS\_c} \tag{5}$$

Equation (4) represents that the credentials provided by the user are invalid, and therefore the user is requested to reenter the valid credentials. On the other hand, the valid credentials trigger AUTH_S transition, and correspondingly, access privileges are granted to the user. The user is transferred to an interface to access the assigned VMs or request new VM instances. Equations (6) and (7) determine the success or failure of the VM request considering several factors, including the quota associated with the user. The VM_req stores the configurations of the requested VM such (CPU, RAM and Disk) which are checked for compliance against the allocated quota of the user. The users quota are stored in UQ and UQ_q is the quota of the specified user.

$$\begin{aligned} R(\text{VM\_F}) = \forall d \in \mathsf{VM\_req} : (&d.\text{username} \neq \mathsf{UQ\_q}.\text{username} \vee \\ &d.\text{cpu} \neq \mathsf{UQ\_q}.\text{cpu} \vee \\ &d.\text{ram} \neq \mathsf{UQ\_q}.\text{ram} \vee \\ &d.\text{disk} \neq \mathsf{UQ\_q}.\text{disk}) \end{aligned} \tag{6}$$

$$\begin{aligned} R(\text{VM\_S}) = \exists d \in \mathsf{VM\_req} : (&d.\text{username} = \mathsf{UQ\_q}.\text{username} \wedge \\ &d.\text{cpu} = \mathsf{UQ\_q}.\text{cpu} \wedge \\ &d.\text{ram} = \mathsf{UQ\_q}.\text{ram} \wedge \\ &d.\text{disk} = \mathsf{UQ\_q}.\text{disk}) \end{aligned} \tag{7}$$

Equation (6) determines the invalidity of the VM request due to a lack of access privileges for additional VM or if the configurations of the requested VM do not comply with the associated quota. The compliance of the requested VM invokes the scheduler service that selects an appropriate server to instantiate the requested VM. Furthermore, the server selection triggers multiple services to configure the VM. For instance, the disk image service provides a guest operating system for the VM. The network service provides networking capabilities to the VM, i.e., initiating a virtual network interface card, assigning a MAC address, and determining the mapping between the machine's virtual and physical interfaces. NET is responsible for leasing IP addresses and the

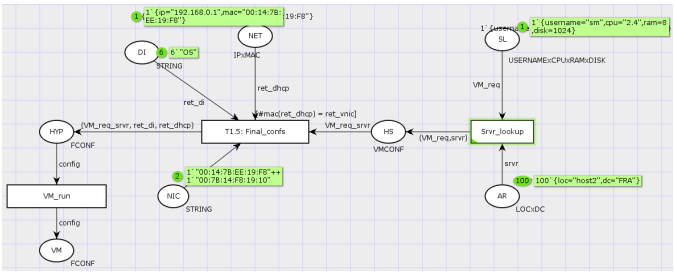

Fig. 7: Snippet of CPN tools of the Final Configurations

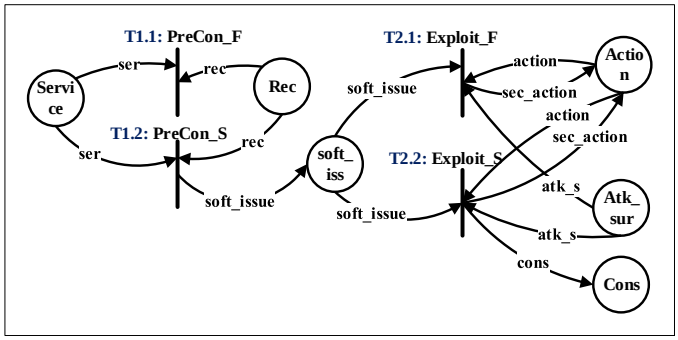

Fig. 8: Modeling a threat's behavior using HLPN

corresponding IP address mapping to the MAC address. These configurations are pushed onto the hypervisor, which executes the VM on the physical hardware. These configurations follow Equation (8) for triggering the respective transition. In Equation (8), we use $++$ to denote tuple concatenation and $:=$ to denote assignment, resulting in an updated variable.

$$\begin{aligned} R(\text{FINAL\_CONFS}) = \exists \text{im} \in \mathit{domain}(\texttt{DI}) : \text{im} = \textsf{ret\_di} \wedge \\ \exists \text{vn} \in \mathit{domain}(\texttt{NIC}) : \text{vn} = \textsf{ret\_vnic} \wedge \\ \exists \text{dh} \in \mathit{domain}(\texttt{NET}) : \text{dh} = \textsf{ret\_dhcp} \wedge \\ \text{dh.mac} = \text{vn.mac} \wedge \\ \textsf{config} := \textsf{VM\_req\_srvr} ++ (\text{im}) ++ \text{dh} \end{aligned} \tag{8}$$

$$\begin{aligned} \text{Final\_confs} ::= \exists \text{im} \in \mathit{domain}(\texttt{DI}) : \text{im} = \textsf{ret\_di} \wedge \\ \exists \text{vn} \in \mathit{domain}(\texttt{NIC}) : \text{vn} = \textsf{ret\_vnic} \wedge \\ \exists \text{dh} \in \mathit{domain}(\texttt{NET}) : \text{dh} = \textsf{ret\_dhcp} \wedge \\ \text{dh.mac} = \text{vn.mac} \rightarrow \\ \textsf{config} := \textsf{VM\_req\_srvr} ++ (\text{im}) ++ \text{dh} \end{aligned} \tag{9}$$

The implementation of Equation (8) in CPN tools is shown in Listing 2 of our supplemental material and the respective snippet of the transitions and places in CPN tools is shown in Figure 7. The place `SL` receives VM configurations and the server lookup is initiated to select the server that can run the requested VM. The selected server and the VM configurations are passed onto the place `HS` which temporarily holds this information. The variable VM_req_srvr in Figure 7 holds both the VM configurations and the server information which is passed to the FINAL_CONFS transition. Further inputs to this transition are from (i) `DI` which provides an operating system for the VM, (ii) `NET` provides IP and MAC addresses, and (iii) `NIC` maps the provided MAC to the network interface card. The transition guard compares vnic with ret_dhcp and a valid guard leads to the firing of the transition. The final configurations (VM_req_srvr, ret_di, ret_dhcp) are passed onto the hypervisor which runs the VM as per the received configurations.

This section explained the functional behavior of the Cloud as a rule-based transition system irrespective of the underlying technologies. The rules determine the information flow among the services for the proper functioning of the Cloud. On the other hand, a threat's input can alter the behavior of the Cloud leading to malfunctioning. Thus, the following section defines the behavior and characteristics (e.g., preconditions, consequence, etc.) of a threat that are given as the spurious input to the Cloud to analyze the threat's impact on the functional behavior of the Cloud.

TABLE 3: Description of Places in Figure 8

Place	Description	Mapping
`Service`	Targeted services.	$\mathbb{P}(Services \times Issues)$
`Rec`	Reconnaissance step input.	$\mathbb{P}(Services \times Issues)$
`soft_iss`	Potential issues in the target.	$\mathbb{P}(Services \times Issues)$
`Action`	Action to exploit the issues.	$\mathbb{P}(Action)$
`Atk_sur`	Attack surface.	$\mathbb{P}(Atk\_sur)$
`Cons`	The consequence of the threat.	$\mathbb{P}(Cons)$

5.8 Instantiation of a Threat's Behavior

The previous sections described the normal functional behavior of the Cloud similar to the basic transition system (cf., Figure 3 in Section 5.1) in a technology-agnostic manner. As previously described, in Figure 3, the paths to the terminal states are modified by additional inputs. This section presents threats as the additional inputs to the Cloud. We define a threat's behavior by representing the necessary conditions required for one or more threats to exploit a vulnerable service. It is worth noting that we do not need to explicitly define the type of the threat actor or threat vectors (e.g., local, network) since any modeled vulnerability will eventually be exploited, if its conditions are met. Moreover, modeling the behavior of threat(s) facilitates assessing the impact of a threat on a particular service and consequently track its progression across the system. The threats are given as input to the Cloud model, and the consequence of the threat dictates the next place in the Cloud model. Furthermore, in combination with the CPN tools [47], the HLPN can be simulated to enumerate benign behavior to validate the functionality of the Cloud and conversely investigate the attack paths generated due to the threat. The instantiation of a threat using HLPN is shown in Figure 8 and Table 3 describes the places used in the HLPN model along with their description and data types. The significance and utilization of these places in defining the threat behavior are explained in the following.

5.9 Reconnaissance Step

This step uncovers potential weaknesses in a system that could be exploited by an attacker. Examples weaknessess include: the installation of a vulnerable version

of software or a misconfigured service. Additionally, this step explores the necessary preconditions to exploit the weakness. The reconnaissance step can be done using different tools but for our purposes, the Vulnerability Database [48] suffices since our purpose is to collect weaknesses in the services as a triggering condition of a transition and consequently track the progression of the threat in the system. Equations (10) and (11) determine if the preconditions of the potential weakness are met.

$$R(\text{PRECON\_S}) = \exists r \in domain(\text{Rec}) : r \in \text{ser} \tag{10}$$

$$R(\text{PRECON\_F}) = \forall r \in domain(\text{Rec}) : r \notin \text{ser} \tag{11}$$

If Equation (10) is true then then a service with a potential issue discovered during the reconnaissance step exists. The absence of such an exploitable weakness instead fires PRECON_F as Equation (11) is true.

5.10 Exploit Step

This step is triggered if a service has an existing issue that could be exploited. This requires an attacker to utilize an action specifically designed to exploit the specific weakness. An absence of such an action indicates an open window of compromise. The rules governing the exploit step are described in Equations (12) and (13).

$$\begin{aligned} R(\text{EXPLOIT\_S}) = {} & \exists i \in domain(\texttt{soft\_iss}) : i = \text{iss} \wedge \\ & \exists a \in domain(\texttt{Action}) : (a = \text{act} \wedge a = \text{iss.issue}) \wedge \\ & \exists as \in domain(\texttt{Atk\_sur}) : as = a \end{aligned} \tag{12}$$

$$\begin{aligned} R(\text{EXPLOIT\_F}) = {} & \forall i \in domain(\texttt{soft\_iss}) : i \neq \text{iss} \vee \\ & \nexists a \in domain(\text{Action}) : (a = \text{act} \vee a = \text{iss.issue}) \vee \\ & \nexists as \in domain(\texttt{Atk\_sur}) : as = a \end{aligned} \tag{13}$$

A successful exploit might affect the normal operations of a system. For instance, a Denial of Service (DoS) would limit the availability of the service. These consequences are represented as the `Cons` in Figure 8. On the other hand, if the consequence of the threat is to bypass authentication then the consequence of the threat is the next available place for the attacker after circumventing the authentication service.

The implementation of threat's instantiation in the CPN tools is given in Listing 3 of our supplemental material and the respective snippet from the CPN tools is shown in Figure 9. For simplicity, we only show the success cases of the rules, i.e., implementation of Equation (10) and Equation (12). The figure shows that the service *Auth* is vulnerable to token mismanagement and is discovered during the reconnaissance phase. Following the discovery, the respective action is taken from the `Action` place which holds actions at an attacker's disposal. An action can be used to exploit multiple issues or it might be the case that exploiting an issue requires multiple independent actions. Therefore, these actions are not tied to specific services or issues. A successful exploit leads to the `Cons` place that holds the impact of the exploit. As mentioned before, the level of granularity depends on the user, i.e., a user can mention a vulnerable

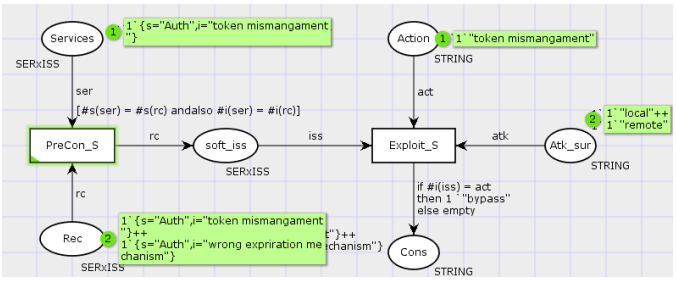

Fig. 9: Snippet of CPN tools depicting threats behavior

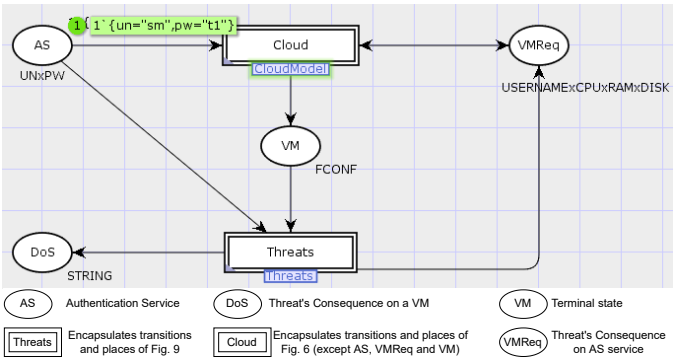

Fig. 10: Link between threats and the Cloud Model

service or software version as well as the corresponding action for that specific vulnerability. However, for our purpose, the description from the NVD suffices as our objective is to perform threat analysis and show the propagation of threats in the Cloud.

We have shown the Cloud model and the instantiation of threat behavior using Petri nets and their implementation in CPN tools. However, the correlation across the Cloud and the threats warrants consideration. This is shown in Figure 10, where after successfully bypassing the authentication server (`AS`), the next place available to the attacker is `VMReq`. `VMReq` is the same as `INT`[5] in Figure 6. On the other hand, a running VM can be targeted with threats causing a denial of service which is shown with the `DoS` place. The functionality of both the *Threats* and the *Cloud Model* in Figure 10 is hidden. These are termed hierarchical Petri nets, and the hierarchy highlights the connection among different blocks while hiding individual block's places and transitions. For instance, the *Threats* block encompasses the places and transitions represented in Figure 8. These hierarchical Petri nets make the model modular and enable adding new modules (e.g., extending the Cloud model by adding new services such as billing, etc) or removing existing modules (e.g., focusing only on specific services such as the authentication mechanisms in the Cloud) simpler. The `VM` is the terminating place of the model.

This section described the necessary blocks to model the Cloud, which captured service interactions representing the system behavior. The information flow model

5. **INT** is a reserved keyword in CPN tools and hence cannot be used as name for a place.

and the threat behavior are defined using HLPN, which allows us to assign multiple constraints to each service and trigger the transition after the satisfaction of preconditions. In the following sections, these blocks are used in the CPN tools to (1) validate the benign operation of the Cloud, (2) perform speculative attack scenarios when threat conditions are satisfied, and (3) perform post-mortem analysis of real-world attack scenarios.

6 ThreatPro Block III: Threat Analysis

The details of the first two building blocks of ThreatPro, i.e., the Cloud model and the information flow model are described in the previous sections (cf., Sections 4 and 5). The Cloud model is an abstraction of services from real-world deployments, while the information flow model governs the flow of information among the services using transitions that are triggered after their respective conditions are satisfied. Section 5.8 comprehensively detailed the threats and their required preconditions in the form of constraints to transitions, so this section builds on these blocks to perform threat analysis in the Cloud. However, before proceeding to threat analysis, we first validate the correct behavior of the Cloud. Specifically, we examine if the Cloud always terminates to the `VM` place each time a user requests a new VM or starts an existing VM. Consequently, allowing us to enumerate all the execution paths that lead to the correct terminal place. The terminal place is `VM` for both (a) starting an existing VM or (b) launching a new instance of the VM. Thereafter, we insert additional constraints acting as threats to different services in order to investigate paths leading to violations of security requirements.

Using an HLPN to build the information flow model facilitates the use of CPN tools [47] to simulate the model and enumerate the Cloud behavior. The simulation allows for the analysis of Cloud's behavior when no adversary is present, i.e., given a valid VM request the terminating place should always be the `VM` place. CPN tools also supports triggering transitions at certain time intervals which facilitates modelling dynamic Cloud behavior. This is accomplished by triggering new events (e.g., launching a new VM, migrating a VM, or fulfillment of a threat's preconditions) after a certain time period has elapsed in the simulation. This establishes the handling of the dynamic behavior of the Cloud by discerning the impact of the new events in the model. In the following sections, we utilize CPN tools to generate places enumerating the Cloud's benign behavior and also its behavior when inserting threats to different services in order to perform threat analysis.

6.1 Enumerating the Cloud behavior

We begin by validating the behavior of the Cloud without threats to understand its normal operations. We achieved this by simulating the HLPN in Figure 6 using CPN tools. Figure 6 dictates that `VM` should be the terminating place when a user requests a VM instance. Using

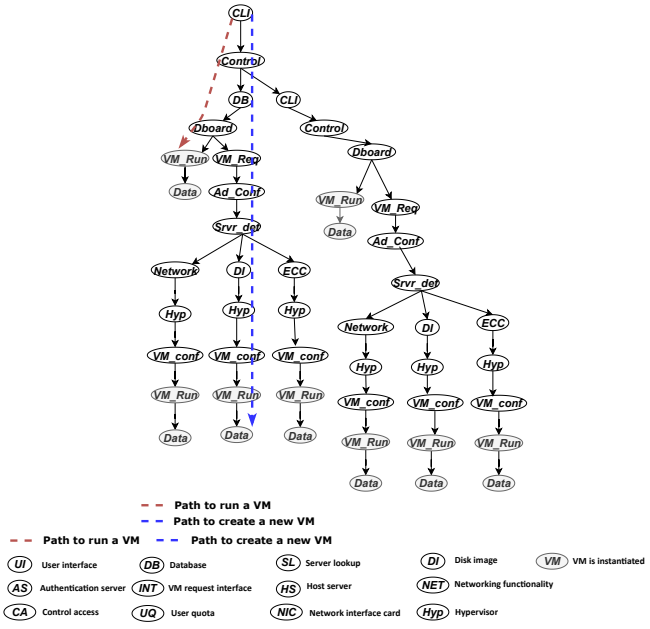

Fig. 11: Example of valid execution paths in the Cloud

CPN tools, we generate the sequence of places for the scenario where a valid user requests a VM. In this valid request, the execution always terminates at the `VM` place. An illustration of a subset of valid paths is shown in Figure 11 where those paths all terminate at the `VM` place. There are some paths that show `VM+Data` instead of `VM` to represent the scenario in which a user had requested storage capacity along with a VM. This is simply used to differentiate between VMs with and without storage. These paths correspond to the instantiation of the Cloud behavior presented in Section 5.7.

In Figures 11 to 14 that represent executions in the Cloud environment, invalid paths and unsuccessful transitions are omitted as the purpose of these figures is to show the validity of the Cloud model through simulation, i.e., a valid request should always terminate at `VM`. In these figures `VM+Data` is shown to indicate that there is storage attached to the requested VM. The storage for VM is optional and hence, it is only shown for some VMs rather than all the instantiated VMs.

6.2 Threat analysis

We now perform the threat analysis by adding constraints (e.g., threat conditions at different services) to the HLPN and simulating the Cloud behavior in the presence of these threats. The threats are added at different layers and services to investigate both the cause-effect relationship and to analyze their impact on the Cloud's functional behavior. To demonstrate the generalization of our approach, we perform speculative analysis using vulnerabilities from the national vulnerability database [48] to identify corresponding attack scenarios. This analysis aims to identify potential paths that attacker could use to undermine a security requirement.

The selected vulnerabilities in our analysis serve as a proof-of-concept. However, we acknowledge that some threats can have varying dependencies and prerequisites that may shape the execution of attacks, which may result in multifaceted attack scenarios. Automating such a process is not within the scope of this paper, and should be considered as future work.

We use the vulnerabilities in Table 4 to demonstrate the effectiveness of using ThreatPro to analyze the potential impact of threats at different layers of the Cloud and the potential of a threat to progress in the Cloud. The attack graph generated from these vulnerabilities is shown in Figure 12. The multiple paths violating security requirements are explained below, where each path enumerates an attack.

Path 1: A successful exploitation of vulnerabilities in path 1 of Figure 12 leads to attaining additional resources in the Cloud from a disabled user. It is accomplished by exploiting CVE-2013-4222/CVE-2012-4457 to request a new authorization token of the disabled user which is used to access the victim's resources. A precondition of the attack requires authentication of the user which could be achieved by exploiting either vulnerability CVE-2013-2006 at the `CA` or CVE-2015-3646 `DB` service.

Path 2: Exploiting CVE-2014-5251 at the control service allows attackers to bypass access restrictions and potentially discover restricted projects. However, in combination with CVE-2018-14432, an attacker can escalate the impact to retain the access of these restricted projects with an expired authorization token. Alternatively, an attacker in combination with CVE-2016-0757 at `SL` might be able to change the VM's configuration. This path specifically shows that combining vulnerabilities from different services can increase the overall impact and therefore, the potential of a threat's progression should be considered in the threat analysis process.

Path 3: Similar to Path 2, this path has multiple potential consequences depending on the vulnerabilities exploited. In path 3a, the vulnerability CVE-2014-9623 at the disk image service is exploited to bypass the storage quota and thus enabling attackers to upload a large image file causing a denial of service. However, path 3b illustrates alternative paths in which the vulnerability is combined with a hypervisor vulnerability (CVE-2014-0134), resulting in either reading the configuration file of the physical server, breaching the confidentiality, or potentially causing the VM to migrate. The latter case opens up new attack surfaces such as exploiting CVE-2018-04635 during VM migration which could allow attackers to intercept network traffic, or CVE-2013-7130 facilitating attacker's access other users' data.

These attack surfaces are introduced due to the elastic behavior of the Cloud. Since this analysis happens at run-time ThreatPro is able to identify these attack paths. Other threat analysis tools that only consider a static view of the system would only incorporate changes in the system after re-execution. These tools might require a large number of re-executions in order to process all the changes that elastic Cloud behavior may introduce.

Speculative Analysis: The speculative analysis allows the exploration of the potential paths an attacker could use to accomplish their objectives. Moreover, this facilitates a proactive approach to threat mitigation and prioritization of threats according to their impact or the threat's degree of centrality in the path. In the following section, we perform a post-mortem analysis of two cases that violate different security requirements, to demonstrate the effectiveness of ThreatPro in identifying threat progression in the system as well as disclosing alternative attack paths through speculative analysis.

7 Validation: Real-world Case Studies

The previous sections outlined the processes of ThreatPro in conducting actual and speculative threat analysis to identify attack paths. To validate ThreatPro, in this section, we use multiple CVEs related to real-world attacks to enumerate the attack paths used to compromise the system. In addition, ThreatPro is able to conduct a post-mortem analysis on these attacks by introducing speculative conditions and exhibiting alternative potential cases of violation of the security requirements. In essence, these potential attack paths determined through speculative analysis highlight ThreatPro's predictive capabilities for identifying alternate possible attacks.

We now present two case studies of actual Cloud attacks to illustrate ThreatPro's methodology. The first attack is the Equifax attack on breach of confidentiality [49] where attackers exfiltrated confidential data of Equifax's customers. The second attack is a resource consumption attack which impacts the application's availability [50].

7.1 Case I: Confidentiality as a Requirement

The first attack scenario covers the violation of a confidentiality requirement. We review the Equifax data breach where attackers successfully ex-filtrated the financial and private records of approximately 148 million users, making it one of the largest data breaches and an attack with one of the largest financial settlements [51]. Furthermore, this case specifically highlights the significance of multi-layer attacks where supposedly negligible issues at different layers were combined to create an aggregated impact. Although threat analysis techniques are useful to determine these issues individually at each service, ThreatPro provides the capability of assessing the outcomes of the threats and their possible combination in the system. This is achieved through modeling the functional behavior to determine a threat's possible progression in the system. A brief analysis of the attack is presented which illustrates the path taken by attackers to access the confidential data of the users. We refer readers to [49] for a complete analysis of the data breach.

1) Attackers exploited a vulnerability in the web portal granting them access to the web server.

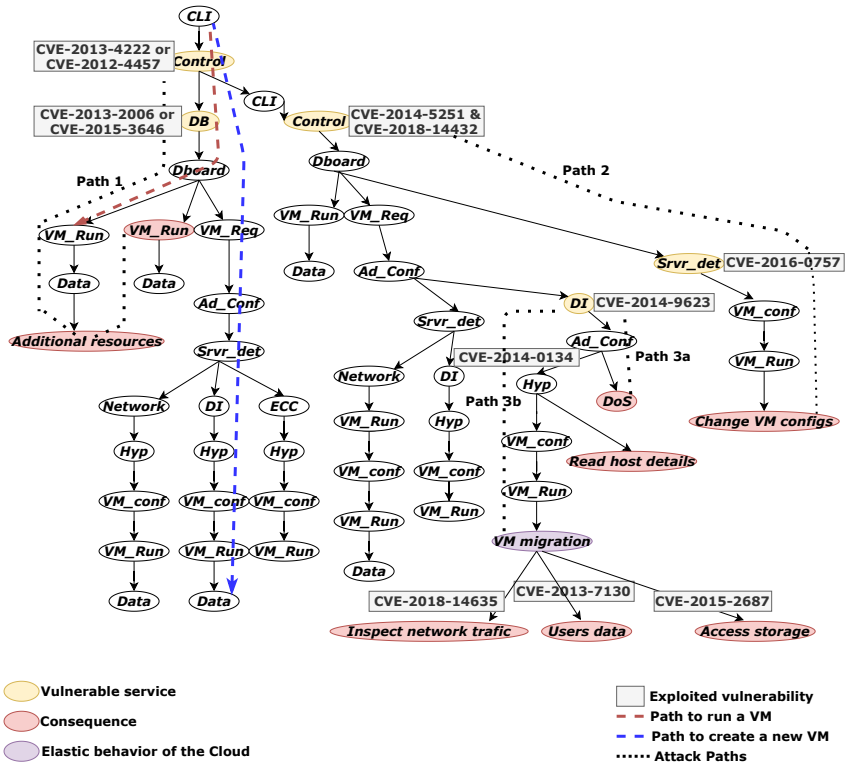

Fig. 12: Attack Paths based on the selected vulnerabilities

TABLE 4: List of vulnerabilities from NVD. The first column in the table is the CVE entry, while the second and third columns show the targeted service and its corresponding HLPN place. The last three columns show the vulnerability's consequence on Confidentiality, Integrity, and Availability (CIA). A full impact with ✓ and a partial impact is indicated with P. Where a partial impact means that a subset of data was revealed to an adversary (confidentially) or a subset of data was corrupted (integrity).

CVE#	Service	Place	C	I	A
2012-4457	Authentication	AS	✓		
2013-2006	Authentication	AS	✓		
2013-4222	Authentication	AS	✓		
2013-7130	Compute	HYP	✓		
2014-0134	Compute	HYP		✓	
2014-2573	Neutron	NET	P		
2014-9623	Glance	DI		P	
2015-2687	Compute	HYP	✓		
2016-5362	Neutron	NET	✓		
2016-0757	Cinder	SL	✓		
2018-14432	Cinder	CA	✓		
2018-14635	Neutron	NET	P		

2) User names and passwords were saved in plain text facilitating attackers to penetrate further into the system using these credentials.
3) Networks and systems were not segmented properly allowing attackers to move laterally across the network and systems without any restriction.

This attack is an example of attackers moving across the services/layers and eventually reaching restricted states of the system due to the presence of negligible issues at each service/layer. A proper partitioning of the network/systems and encrypting the credentials at rest would have limited the impact of the attack. However, the combination of these issues across different services/layers amplified the impact of the attack. ThreatPro generates the sequence of steps that enable attackers to access the data which are shown in Figure 13.

Figure 13 shows the attacker compromised the web server running on the VM at host 1 by exploiting the publicly known vulnerability CVE-2017-5638. This allowed attackers to gain access to the VM resources and the storage of the unencrypted credentials which facilitated penetrating further into the system by using these credentials. Systems/networks were not properly segmented allowing attackers to use the credentials on VMs running at different hosts, e.g., host 2 in Figure 13. We now demonstrate the capability of ThreatPro in revealing alternative attack paths at the attacker's disposal.

7.1.1 Speculative Analysis

Figure 13 shows the potential issues that were exploited by the attacker, however, the speculative analysis of the

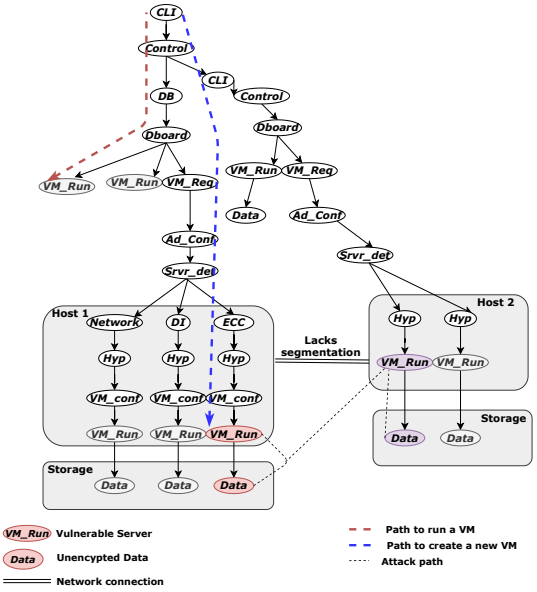

Fig. 13: Attack Path: Equifax data breach (Section 7.1)

Equifax data breach reveals that the attackers have alternative attacks paths at their disposal to accomplish their goals. For instance, if the network is partitioned properly, an alternative route for the attacker could be to intercept network traffic by exploiting CVE-2016-5363/CVE-2016-5362 at the network service. Thus, speculative analysis is useful to determine the alternative paths exploitable by an attacker in case a mitigation strategy is deployed.

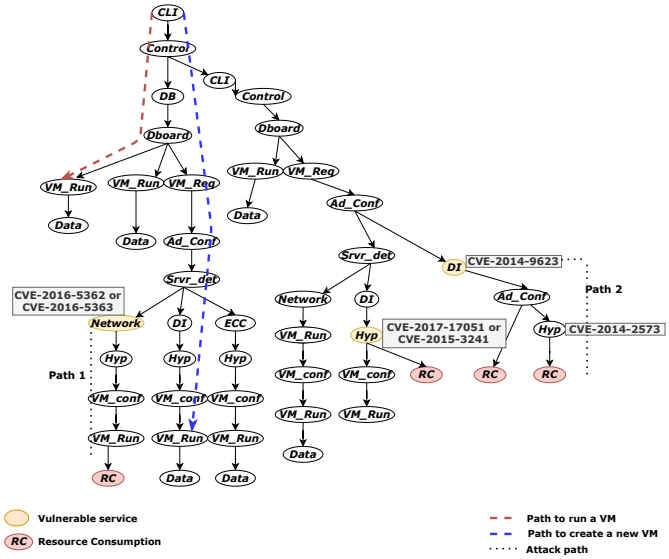

Fig. 14: Attack Path: resource consumption attack (Section 7.2)

7.2 Case II: Availability as a requirement

The second attack illustrates the use of ThreatPro in determining the paths violating the availability requirements of an application. Specifically, this attack entails exhausting the resources to limit the availability of an application and eventually causing a denial of service. These attacks typically target content delivery applications where timely delivery of content is the primary objective [52], [53]. Recently, Amazon reported that it has thwarted the biggest attack on its services [50]. The documented information is limited in these cases to avoid leakage of propriety information that could potentially be used in future attacks. However, using the threats published in the NVD, ThreatPro is able to depict scenarios where an attacker can target individual services or discover a combination of vulnerabilities to cause exhaustion of the resources. These attack paths are shown in Figure 14 and are explained below.

7.2.1 Paths 1 and 2

Using CVE-2016-5362 or CVE-2016-5363 at the network service, an attacker can intercept traffic and cause a resource consumption attack. This vulnerability allows the interception of traffic destined for other hosts and thus, could potentially be used to intercept snapshots of the VM during a migration and consequently enable attackers to exhaust resources. Alternatively, in path 2, exploiting CVE-2014-9623 at the disk image service combined with a vulnerability at the hypervisor (CVE-2014-2573) leads to a resource consumption attack. Furthermore, exploiting either CVE-2017-17051 or CVE-2015-3241 at the hypervisor also leads to exhausting resources by repeatedly rebuilding instances with new disk images.

7.2.2 Speculative analysis

Performing speculative analysis reveals alternative paths that might result in exhausting a resource, e.g., the vulnerabilities CVE-2017-17051 and CVE-2015-3241 can be used to exploit the functionality of a hypervisor to exhaust resources by repeatedly building the same instance. This causes double allocations and repeating the process causes a DoS as the resources get exhausted.

These attack scenarios illustrate that a proactive approach is required to analyze a threat in the Cloud to explore possible attack paths that can be exploited by attackers. ThreatPro can be used to perform speculative cause-effect analysis to determine the impact of a threat at a single service and to analyze the impact of multiple threats towards violating a security requirement.

8 Discussion

ThreatPro provides a methodology to perform threat analysis in dynamic Cloud environments. It is based on the manual specification of an information flow model that is developed using HLPNs. Services and threats are represented as rules or constraints, which are added to the model and evaluated against security requirements. As stated in Section 5.4, in contrast to LTSs, HLPNs leverage distributed places and allows actions to be applied locally. Hence, any impact of new rules or constraints is determined locally at the targeted service. Consequently, a threat's propagation starts from the targeted service instead of the system's starting point. The latter assists in performing cause-effect analysis of new services and allows Cloud Service Providers (CSPs) to identify security implications introduced by a service. The process can be repeated for threats at different services, and thus, assist CSPs to track the propagation of threats in the Cloud. In the following, we discuss how ThreatPro can perform predictive analysis and how to add new services to the information flow model.

The modelling of the dynamic interconnections is primarily achieved by launching new VM instances while the previous ones are either in a running state or at a later stage of their creation (e.g., final configurations at the hypervisor). In Petri nets, the actions to places are local and hence, multiple requests can be launched concurrently. Furthermore, VM requests can be be restricted to instantiate only after certain time period has elapsed. The aim of ThreatPro is to speculatively evaluate the consequences of threats by ascertaining their potential to propagate across different layers of the dynamic Cloud environments. The capability for exploratory analysis can form the basis of automating this in future work.

The performance analysis and scalability of ThreatPro is constrained by CPN tools. When considering large-scale Cloud environments, with thousands of components and services, computational complexity can be a challenge. Nevertheless, CPN tools, by encapsulating system behavior hierarchically and offering parallel processing, provides scalability to the approach. While the exact performance depends on analysing the model and function definitions, CPN tools can handle the intricacies and scale of modern Cloud environments, helping Cloud service providers make informed decisions [54].

8.1 Predictive Analysis

In Sections 6 and 7, we presented how ThreatPro can perform speculative threat analysis and post-mortem analysis of security requirements such as confidentiality and availability. However, ThreatPro can be extended handle attacks where information is missing or a countermeasure has been applied. In the Equifax data breach example, exploring possible attack paths after hardening the network or mitigating the vulnerability at the web server shows the result of the countermeasure. Additionally, when the network is partitioned properly, but the CVE-2016-5363 or CVE-2016-5362 are present, either can be exploited to intercept network traffic from other hosts and for attackers to circumvent network partitioning. The ability to complete paths in case of missing information or to find alternative paths of attacks can empower CSPs to mitigate additional attack paths. This results in eventually moving away from threat analysis being reactive to proactive. Furthermore, mitigation strategies can focus on services that have a higher degree of centrality in attack paths to reduce the impact of attacks.

8.2 Plug and Play Services

As stated in Section 3, Cloud deployments may vary between vendors. In this paper, the adopted Cloud model is an abstraction of common services used in the lifecycle of a VM. However, the model can be extended to include vendor-specific or additional services to enhance the Cloud functionality. To achieve this, new places and their respective transitions and constraints need to be added to the information flow model. As shown in Figure 10, the advantage of hierarchical Petri nets is that it hides the functionality of individual blocks to focus on the interaction between blocks. This makes extension of the model simpler, i.e., new functionality can be added as an independent block and the respective connections can occur on the edge transitions. The added functionality can be simulated to assess its influence on the functional behavior of the Cloud, i.e., if the added functionality leads to a proper terminating place or introduces any issue. Similarly, threats introduced with new services can be added to assess their propagation paths. ThreatPro's methodology remains agnostic to underlying technologies since constraints from both threats and services are at the functional level. In case the functionality has to be removed, all that is required is to disconnect the blocks to restore the previous place of the model.

8.3 Automation

ThreatPro currently relies on the manual generation of the models and threats as inputs, which can be a restriction and prone to human errors; however, it is a common problem in all model-based systems that require human intervention. ThreatPro already supports automated speculative analysis as an output of the tool in Sections 7.1.1 and 7.2.2. Therefore, in this discussion, we focus on automating building inputs to ThreatPro.

8.3.1 Constructing Petri Net Models of Threats

There is an ongoing effort to create a uniform format for vulnerabilities in standardized formats [55]. This data can contain vulnerability preconditions and, to a certain extent, mechanisms to exploit the vulnerability. This information could be used to automate the construction of Petri net specifications for many vulnerabilities (as was performed manually in Section 5.8). However, the vulnerability data is limited, especially regarding a logical specification of the threat and its impact.

8.3.2 Constructing Petri Net Models of System Model

Similarly, automating the generation of Petri net models for system representation may facilitate the adoption of ThreatPro. Expanding beyond the manual methods in Section 5.7, here we explore potential techniques for automatically constructing Petri net models of systems.

One approach for automation may involve utilizing the high-level specifications of systems. Specifications could be represented formally using a domain-specific language or modeling framework which can then be translated into a Petri net. Such methods may ensure that the resulting model accurately captures the intricate interactions and dependencies within a system [56].

Natural language processing (NLP) offers another promising approach for automating the generation of Petri net models from system descriptions. Through NLP techniques, natural language specifications can be analyzed [57], and key elements can be extracted to construct a corresponding Petri net.

Additionally, the automatic construction of Petri net models could be achieved by extracting information from system logs. Analyzing logs generated by systems provides insights into the runtime behavior, resource utilization, and communication patterns. This data can be utilized to dynamically generate Petri net models that capture the evolving dynamics of a system [58].

8.3.3 Automation Limitations

For ThreatPro, the expectation is that threats of interest will need to be manually defined according to the security properties of interest and manually incorporated into a system's analysis. If detailed vulnerability specifications are available in the future, then these could be used to automatically derive threat definitions in ThreatPro to avoid needing to add threats manually. However, the types of threats to consider will likely need to remain a manual process, as how specifications for systems or vulnerabilities are automatically modelled will need to take into account the threats of interest.

8.4 Limitations

The threat landscape is evolving rapidly and coverage for all possible threats is not feasible for a threat analysis technique. ThreatPro focuses on threats that are publicly documented in NVD to perform threat analysis. However, it is also able to incorporate new threats by adding

them as additional constraints or rules to the information flow model, even from other repositories than NVD (e.g., Microsoft's security bulletin [59], Google's open-source vulnerability database [60]). Thus, ThreatPro can be extended to consider novel threats associated with a service and determine the execution paths followed by incorporating them into the Cloud model.

9 Conclusions

This paper introduced ThreatPro as a dynamic Cloud threat analysis methodology that addresses the evolving Cloud security landscape. By integrating the dynamic nature of the Cloud into threat analysis, ThreatPro goes beyond traditional approaches, providing a flexible framework for assessing potential security threats. Our key findings in the paper are the following.

ThreatPro offers a powerful framework for conducting predictive threat analysis in dynamic Cloud environments. It enables CSPs to identify potential attack paths and assess their consequences proactively. This approach helps mitigate security risks before they manifest, moving Cloud security from a reactive to a proactive stance.

ThreatPro's flexibility allows for seamlessly integrating vendor-specific or additional services into the information flow model. This extensibility simplifies the process of incorporating new functionality and assessing its impact on the Cloud's security. ThreatPro's modular approach, with the use of Petri nets, enables the addition of new services without disrupting the existing model.

The rapidly evolving threat landscape presents a significant challenge for Cloud security. ThreatPro is designed to accommodate new threats from various sources, not limited to the NVD. By enabling the inclusion of novel threats associated with services, ThreatPro ensures that Cloud security remains up-to-date and resilient in the face of emerging security challenges.

In summary, ThreatPro offers a comprehensive and proactive approach to enhance Cloud security by enabling predictive threat analysis, customizing the security model with ease, adapting to evolving threat scenarios, and positioning it to effectively tackle emerging security challenges in Cloud environments.

References

[1] B. Edwards, S. Hofmeyr, and S. Forrest, "Hype and heavy tails: A closer look at data breaches," *International Journal of Cybersecurity*, vol. 2, pp. 3–14, 2016.

[2] M. Masdari and M. Jalali, "A survey and taxonomy of dos attacks in cloud computing," *International Journal of Security and Communication Networks*, vol. 9, pp. 3724–3751, 2016.

[3] H. Abusaimeh, "Security attacks in cloud computing and corresponding defending mechanisms," *Intl. Journal of Advanced Trends in Computer Science and Engg.*, vol. 9, pp. 4141–4148, 2020.

[4] D. Sgandurra and E. Lupu, "Evolution of Attacks, Threat Models, and Solutions for Virtualized Systems," *ACM Computing Surveys*, vol. 48, pp. 1–38, 2016.

[5] N. Gruschka and M. Jensen, "Attack surfaces: A Taxonomy for Attacks on Cloud Services," in *Proc. of Intl. Conference on Cloud Computing*. Miami, FL, USA: IEEE, 5–10 July 2010, pp. 276–279.

[6] L. Wang, Z. Zhu, Z. Wang, and D. Meng, "Colored Petri net Based Cache Side Channel Vulnerability Evaluation," *IEEE Access*, vol. 7, pp. 169 825–169 843, 2019.

[7] R. Ritchey and P. Ammann, "Using model checking to analyze network vulnerabilities," in *Proceedings of the Symposium on Security and Privacy*, USA, 14–17 May 2000, pp. 156–165.

[8] D. Akhawe, A. Barth, P. E. Lam, J. Mitchell, and D. Song, "Towards a formal foundation of web security," in *IEEE Computer Security Foundations Symposium*, 17–19 July 2010, pp. 290–304.

[9] K. Torkura, M. Sukmana, M. Meinig, A. Kayem, F. Cheng, H. Graupner, and C. Meinel, "Securing Cloud Storage Brokerage Systems Through Threat Models," in *Proceedings of the International Conference on Advanced Information Networking and Applications*. Krakow, Poland: IEEE, 16–18 May 2018, pp. 759–768.

[10] N. Alhebaishi, L. Wang, S. Jajodia, and A. Singhal, "Threat Modeling for Cloud Data Center Infrastructures," in *Intl. Symposium on Foundations & Practice of Security*. Québec, Canada: Springer International Publishing, 24–25 October 2016, pp. 302–319.

[11] A. Nhlabatsi, J. Hong, D. Kim, R. Fernandez, A. Hussein, N. Fetais, and K. Khan, "Threat-specific security risk evaluation in the cloud," *IEEE Trans. on Cloud Computing*, vol. 9, pp. 1–13, 2018.

[12] P. Wang, W.-H. Lin, P.-T. Kuo, H.-T. Lin, and T. C. Wang, "Threat risk analysis for cloud security based on attack-defense trees," in *Proc. of the International Conference on Computing Technology and Information Management*, Seoul, South Korea, 2012, pp. 106–111.

[13] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based Attack and Defense Modeling: Don't Miss the Forest for the Attack Trees," *Computer Science Review*, vol. 13–14, pp. 1–38, 2014.

[14] V. Varadharajan, "Petri net based modelling of information flow security requirements," in *Proceedings of the Computer Security Foundations Workshop*, Franconia, NH, USA, 1990, pp. 51–61.

[15] T. DeMarco, *Structured Analysis and System Specification*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1979.

[16] A. Naskos, A. Gounaris, H. Mouratidis, and P. Katsaros, "Online analysis of security risks in elastic cloud applications," *IEEE Cloud Computing*, vol. 3, pp. 26–33, 2016.

[17] D. Santos, R. Marinho, G. Schmitt, C. Westphall, and C. Westphall, "A framework and risk assessment approaches for risk-based access control in the cloud," *Journal of Network and Computer Applications*, vol. 74, pp. 86–97, Oct. 2016.

[18] A. Nhlabatsi, K. Khan, J. Hong, D. Kim, R. Fernandez, and N. Fetais, "Quantifying Satisfaction of Security Requirements of Cloud Software Systems," *IEEE Trans. on Cloud Computing*, pp. 1–18, 2021.

[19] A. Sen and S. Madria, "Risk assessment in a sensor cloud framework using attack graphs," *IEEE Trans. on Services Computing*, vol. 10, pp. 942–955, 2017.

[20] S. Islam, M. Ouedraogo, C. Kalloniatis, H. Mouratidis, and S. Gritzalis, "Assurance of security and privacy requirements for cloud deployment models," *IEEE Trans. on Cloud Computing*, vol. 6, pp. 387–400, 2018.

[21] P. Saripalli and B. Walters, "QUIRC: A Quantitative Impact and Risk Assessment Framework for Cloud Security," in *Proceedings of the International Conference on Cloud Computing*. Miami, FL, USA: IEEE, 2010, pp. 280–288.

[22] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using bayesian attack graphs," *IEEE Trans. on Dependable and Secure Computing*, vol. 9, pp. 61–74, 2012.

[23] D. Gonzales, J. Kaplan, E. Saltzman, Z. Winkelman, and D. Woods, "Cloud-Trust—a Security Assessment Model for Infrastructure as a Service (IaaS) Clouds," *IEEE Trans. on Cloud Computing*, vol. 5, pp. 523–536, 2017.

[24] S. Y. Enoch, J. B. Hong, and D. S. Kim, "Security modelling and assessment of modern networks using time independent graphical security models," *Journal of Network and Computer Applications*, vol. 148, p. 102448, 2019.

[25] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Attack graph-based moving target defense in software-defined networks," *IEEE Trans. on Network and Service Management*, vol. 17, no. 3, pp. 1653–1668, 2020.

[26] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *International Conference on Data Engineering Workshops*. Long Beach, CA, USA: IEEE, 1–6 March 2010, pp. 87–92.

[27] F. Machida, E. Andrade, D. Kim, and K. Trivedi, "Candy: Component-based Availability Modeling Framework for Cloud

Service Management Using SysML," in *Proceedings of the International Symposium on Reliable Distributed Systems*. Madrid, Spain: IEEE, 4–7 October 2011, pp. 209–218.
[28] F. Metzger, T. Hoßfeld, A. Bauer, S. Kounev, and P. Heegaard, "Modeling of aggregated iot traffic and its application to an iot cloud," *Proceedings of the IEEE*, vol. 107, pp. 679–694, 2019.
[29] S. Manvi and G. Shyam, "Resource Management for IaaS in Cloud Computing: A Survey," *Intl. Journal of Network & Computer Applications*, vol. 41, pp. 424–440, 2014.
[30] A. Younge, G. Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient Resource Management for Cloud Computing Environments," in *Proc. of the International Conference on Green Computing*. Chicago, IL, USA: IEEE, 2010, pp. 357–364.
[31] R. Smith, "Computing in the cloud," *International Journal of Research-Technology Management*, vol. 52, pp. 65–68, 2009.
[32] X. Jin, Q. Wang, X. Li, X. Chen, and W. Wang, "Cloud virtual machine llifecycle security framework based on trusted computing," *Journal of Tsinghua Science & Technology*, vol. 24, pp. 520–534, 2019.
[33] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "Openstack: Toward an open-source solution for cloud computing," *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, Oct. 2012.
[34] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *Proc. of the Intl. Symposium on Cluster Computing and the Grid*. IEEE/ACM, 2009, pp. 124–131.
[35] S. Manzoor, T. Vateva-Gurova, R. Trapero, and N. Suri, "Threat Modeling the Cloud: An Ontology Based Approach," in *Information and Operational Technology Security Systems*. Springer Intl. Publishing, 2019, pp. 61–72.
[36] A. Desai, R. Oza, P. Sharma, and B. Patel, "Hypervisor: A survey on concepts and taxonomy," *International journal of Innovative Technology and Exploring Engineering*, vol. 2, pp. 222–225, 2013.
[37] Z. Benzadri, F. Belala, and C. Bouanaka, "Towards a Formal Model for Cloud Computing," in *Service-Oriented Computing*. Cham: Springer International Publishing, 2013, pp. 381–393.
[38] K. Bósa, R. Holom, and M. Vleju, *A Formal Model of Client-Cloud Interaction*. Springer Intl. Publishing, 2015, pp. 83–144.
[39] H. Sahli, C. Bouanaka, and A. Dib, "Towards a Formal Model for Cloud Computing Elasticity," in *IEEE 23rd International WETICE Conference*, Parma, Italy, 23–25 June 2014, pp. 359–364.
[40] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, *Progress on the State Explosion Problem in Model Checking*. Berlin, Heidelberg: Springer-Verlag, 2001, pp. 176—-194.
[41] K. Salimifard and M. Wright, "Petri net-based modelling of workflow systems: An overview," *European journal of operational research*, vol. 134, pp. 664–676, 2001.
[42] A. Brogi, A. Canciani, J. Soldani, and P. Wang, *A Petri Net-Based Approach to Model and Analyze the Management of Cloud Applications*. Springer Berlin Heidelberg, 2016, pp. 28–48.
[43] R. Boubour, C. Jard, A. Aghasaryan, E. Fabre, and A. Benveniste, "A petri net approach to fault detection and diagnosis in distributed systems," in *Proceedings of the IEEE Conference on Decision and Control*, vol. 1, San Diego, CA, USA, Dec. 1997, pp. 720–725.
[44] ISO Central Secretary, "High-level Petri nets - Part 1: Concepts, Definitions and Graphical notation," Intl. Organization for Standardization, Geneva, Switzerland, Standard ISO/IEC 15909-1:2019, Aug. 2019. [Online]. Available: https://www.iso.org/standard/67235.html
[45] E. W. Dijkstra, "Guarded commands, nondeterminacy and formal derivation of programs," *Commun. ACM*, vol. 18, no. 8, pp. 453—-457, aug 1975.
[46] K. Jensen and L. Kristensen, *CPN ML Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ch. 3, pp. 43–77.
[47] K. Jensen, L. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, pp. 213–254, 2007.
[48] NIST, "National vulnerability database." [Online]. Available: https://nvd.nist.gov/
[49] P. Wang and C. Johnson, "Cybersecurity incident handling: a case study of the equifax data breach." *Issues in Information Systems*, vol. 19, pp. 150–159, 2018.
[50] J. Porter, "Amazon Mitigated the Largest DDoS Attack Ever Recorded," 2020. [Online]. Available: https://www.theverge.com/2020/6/18/21295337/
[51] S. Cowley, "Equifax to Pay at Least 650 Million in Largest-Ever Data Breach Settlement," 2019. [Online]. Available: https://www.nytimes.com/2019/07/22/business/equifax-settlement.html
[52] Cloudfare, "Famous DDoS attacks: The largest DDoS attacks of all time," 2021. [Online]. Available: https://www.cloudflare.com/en-gb/learning/ddos/famous-ddos-attacks
[53] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in IoT: Mirai & other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[54] CPN Tools, "Performance analysis," 2013. [Online]. Available: https://cpntools.org/category/documentation/doc-tasks-performance/
[55] B. Jordan, R. Piazza, and T. Darley, Eds., *STIX Version 2.1*. OASIS Standard, 10 June 2021. [Online]. Available: https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.pdf
[56] J. Brandt and W. Reisig, "Modeling Erlang processes as Petri nets," in *Proceedings of the 17th ACM SIGPLAN International Workshop on Erlang*, ser. Erlang 2018. ACM, 2018, pp. 6–66.
[57] E. Sarmiento, J. C. Leite, E. Almentero, and G. S. Alzamora, "Test scenario generation from natural language requirements descriptions based on petri-nets," *Electronic Notes in Theoretical Computer Science*, vol. 329, pp. 123–148, 2016.
[58] I. Beschastnikh, Y. Brun, M. D. Ernst, and A. Krishnamurthy, "Inferring models of concurrent systems from logs of their behavior with CSight," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. ACM, 2014, pp. 468–479.
[59] MS, "Microsoft security response center," n.d. [Online]. Available: https://msrc.microsoft.com/update-guide/vulnerability
[60] Google, "Open source vulnerabilities," n.d. [Online]. Available: https://osv.dev/list

Salman Manzoor has a PhD from Lancaster University and is currently employed at Barclays. His research interests are in threat modelling and security assessment. His profile is available at https://ssg.lancs.ac.uk/people/salman-manzoor/.

Antonios Gouglidis is a Senior Lecturer (Associate Professor) at the School of Computing and Communications at Lancaster University in the UK. He is interested in formal methods for security. His professional profile is available at https://ssg.lancs.ac.uk/people/gouglidis/.

Matthew Bradbury is a Lecturer at the School of Computing and Communications at Lancaster University in the UK. He is interested in the security, privacy and trust in resource-constrained and distributed systems. His professional profile is available at https://mbradbury.github.io/.

Neeraj Suri is a Distinguished Professor of Cybersecurity at Lancaster University and an adjunct Professor in the Department of CS at the Univ of Massachusetts at Amherst. His research interests are in the specification and analysis of system level security. His professional profile is available at https://ssg.lancs.ac.uk/people/suri/.

Enabling Multi-Layer Threat Analysis in Dynamic Cloud Environments - Supplemental

Salman Manzoor§, Antonios Gouglidis, Matthew Bradbury and Neeraj Suri
Lancaster University, UK
Email: {s.manzoor1, a.gouglidis, m.s.bradbury, neeraj.suri}@lancaster.ac.uk

Listing 1: CPN ML implementation of Equation (1)

```
colset Usernames = string; (* Type of Usernames is string *)
colset Passwords = string; (* Type of Passwords is string *)
colset UNxPW = record un:Usernames * pw:Passwords; (* Type for multiple fields *)
var un:Usernames; (* Variable of type Usernames *)
var pw:Passwords; (* Variable of type Passwords *)
var U,C:UNxPW; (* Variables of type UNxPW *)
Auth_S = [#un(U)<>O andalso #un(U)=#un(C) andalso #pw(U)=#pw(C)] (* Trans. guard*)
O' = O^#un(U) (* Username is added to online users *)
Auth_F = [#un(U)=O orelse #un(U)=#un(C) orelse #pw(U)=#pw(C)] (* Trans. guard *)
```

Listing 2: CPN ML implementation of Equation (8)

```
colset CPU = string; (* Type of CPU is string *)
colset RAM = int; (* Type of RAM is int *)
colset DISK = int; (* Type of RAM is int *)
colset USERNAMExCPUxRAMxDISK = record un:USERNAME * cpu:CPU * ram:RAM * disk:DISK
var VM_req:USERNAMExCPUxRAMxDISK; (* Variable of type USERNAMExCPUxRAMxDISK *)
colset LOCxDC= record loc:LOC * dc:DC; (* Type of multiple fields *)
var srvr:LOCxDC; (* Type of LOCxDC *)
colset VMCONF = product USERNAMExCPUxRAMxDISK * LOCxDC (* Immutable fields *)
var VM_req_srvr:VMCONF; (* Variable of type VMCONF *)
colset IP = string; (* Type of IP is string *)
colset MAC= string; (* Type of MAC is string *)
colset IPxMAC= record ip:IP * mac:MAC; (* Type of multiple fields *)
var ret_dhcp:IPxMAC; (* Variable of type IPxMAC *)
colset DI = string; (* Type of DI is string *)
var get_di:DI; (* Variable of type DI *)
colset FCONF = product VMCONF * DI * IPxMAC;
var config:FCONF;
Final_confs = [#mac(ret_dhcp) = ret_vnic] (* Trans. guard*)
```

Listing 3: CPN ML implementation of Equation (10) and Equation (12)

```
colset SERVICE = string; (* Type of service is string *)
colset ISSUE = string; (* Type of ISSUE is string *)
colset SERxISS = record s:SERVICE * i:ISSUE;
var ser, rc, iss:SERxISS; (* Variable of type SERxISS *)
var act, atk:STRING;
PreCon_S = [#s(ser) = #s(rc) andalso #i(ser) = #i(rc)] (* Trans. guard*)
Exploit_S = if #i(iss) = act
            then 1`"bypass"
            else empty (* Trans. guard and output condition merged *)
```

§. The research was conducted when the author was affiliated with Lancaster University.