Connectivity Extraction in Cloud Infrastructures

Pernelle Mensah, Samuel Dubus, Wael Kanoun, Christine Morin, Guillaume Piolle, Eric Totel

Connectivity Extraction in Cloud Infrastructures

Pernelle Mensah*†‡, Samuel Dubus*
and Waël Kanoun*
*Cybersecurity Paris Saclay
Nokia Bell Labs
Nozay, France
firstname.name@nokia-bell-labs.com

Christine Morin†
†Myriads
Inria
Rennes, France
christine.morin@inria.fr

Guillaume Piolle‡†
and Eric Totel‡†
‡CIDRE
CentraleSupélec
Cesson-Sévigné, France
firstname.name@centralesupelec.fr

***Abstract*—For management and security purposes, cloud providers should know the connectivity graph between virtual machines. Since traditional methods used in physical networks produce incomplete results and are hardly usable in the Cloud, we propose to use information provided by a Cloud Management Software and an SDN controller, to compute the connectivity graph in those environments. Our approach shows an exact, complete and up-to-date connectivity graphs computation on a representative infrastructure, in reasonable time.**

I. Introduction

Enterprise networks complexity renders the knowledge of the connectivity an essential tool to facilitate management tasks and enhance network security. That interconnection knowledge can provide useful indications to speed up the identification of points of failure in case of network outage, and can also be used to perform proactive impact analysis of devices or links failure on the users. From a security standpoint, it represents one of the building blocks for an attack graph generation, used to perform vulnerability chains construction [13], [17], [18].

In physical infrastructures, the acquisition of the topology has been vastly addressed in existing works with either passive or active discovery methods. Those methodologies present issues such as their intrusiveness, extensive probing traffic and path redundancy due to the repeated interrogation of the same interfaces across several queries. With the Cloud, in which virtualization attacks and virtual infrastructure dynamic nature are introduced, new methods need to be developed. Indeed, the dynamic and customizable nature of virtual machines deployed in the Cloud, renders difficult the installation of agents dedicated to topology retrieval. On the other hand, given the Cloud economic model, providers benefit from optimizing traffic consumption, since resources are charged according to usage, hence probing traffic should be minimal.

To the best of our knowledge, we present in this paper the first method to address the retrieval of an up-to-date topology and connectivity in cloud environments. We designed a module using cloud technologies to retrieve an updated topology and connectivity, while avoiding limitations of physical infrastructure methods. In an environment with a Cloud Management Software (CMS) and an SDN (Software-Defined Networking) controller interfaced with the CMS, our module retrieves the current topology and builds the associated connectivity, when plugged into a running cloud. This represents the **static topology and connectivity retrieval**. Secondly, the module listens to change events generated inside the infrastructure and within the SDN controller in order to update the topology and connectivity previously built during the static steps: this represents the **dynamic topology and connectivity retrieval**. Required information for topology and connectivity being retrieved from two sources, we handled occasional conflicts in network states. Section II presents the state of the art relative to topology extraction in regular infrastructures, and Section III, the environment, its model and our solution's implementation. We present our conclusions in Section IV.

II. State of the Art on Topology Discovery

We present the state of the art regarding topology discovery techniques, organized into passive and active methods. Passive and active methodologies applicability to the Cloud is analyzed to uncover limits and requirements for an efficient topology and connectivity retrieval. Focus is on topology, i.e. the architecture, since limited details, if any, are given for connectivity retrieval approaches in the studied papers, i.e the protocols and direction of communication between machines.

A. Passive Discovery Methods

Passive methods are based on a non-intrusive observation of the network traffic to detect devices and reconstruct equipment topology. Passive measurements can be carried out by the deployment of specialized hardware such as network taps [10] at strategic locations in the network and binding them to traffic analyzers, however with significant costs. Other methodologies involve port mirroring, incurring an additional workload on the switches concerned, as each packet on the monitored ports is copied and sent to a monitoring host [19]. Flow export protocols such as sFlow or NetFlow can also be leveraged to reconstruct the topology. They provide access to information pertaining to layer 2, 3 and 4 of the OSI model. Few methods in the literature rely solely on a pure passive methodology for topology reconstruction. However, Eriksson and al. [9] used passive measurements to infer structural properties in the Internet. By observing the hop-count vectors between sources and passive monitors, they are able to cluster sources sharing network paths, according to similarities discovered.

This article has been published as a contribution to the 13th International Conference on Network and Service Management (CNSM 2017)

B. Active Discovery Methods

There are two kinds of active methods: agent- and monitor-based approaches.

1) Agent-based Approaches: They rely on agents deployed in each device to audit, and often use the SNMP protocol, with SNMP agents installed in routers, switches or end-hosts. Network management tools allow the automated discovery of routers, subnets and layer-3 topology. Breitbart and al. [6] propose an algorithm based on standard SNMP information to construct layer-2 topology. Lowerkamp and al. [14] have extended this work by integrating incomplete database knowledge and non-cooperative (without SNMP) equipments such as hubs. Even when relying on standard protocols such as SNMP, difficulties arise due to vendors specificities. Indeed, implementation can be extended across platforms, and inconsistencies in table indexing schemes may occur. This leads to additional challenges when processing data originating from multiple sources. Besides, agent-based approaches lead to intrusiveness into infrastructure devices, due to the need for an agent in customers' equipment.

2) Monitor-based Approaches: Monitor-based approaches are more flexible than the previous ones: they use a dedicated set of probing hosts, responsible for performing topology acquisition by leveraging protocols and applications already available in the users' devices (i.e. ICMP, traceroute, ping). They do not require the use of customers' resources, since they are independent from their hardware. Skitter [16], tool developed by the Center for Applied Internet Data Analysis (CAIDA), and the Test Traffic Measurement (TTM) [11] from RIPE Network Coordination Center (NCC) are extensive tracing systems that have been used for Internet topology discovery at the IP level. They leverage traceroute mechanisms between 24 to 200 hundreds monitors to reconstruct the topology. Donnet and al. [8] determined that traceroute-based tools for discovery can be inefficient, as they have to deal with the redundancy induced from repetitively probing the same interfaces. Hence, to decrease probing traffic, they opted for Doubletree [8], an algorithm allowing to reduce simultaneously intra- and inter- monitor duplicated data, by starting the probing at an intermediate distance between monitor and destination, and performing backward (destination-rooted tree) and forward (monitor-rooted tree) probing schemes. Monitors share paths they already probed to their destinations, to avoid their peers to take the same ones.

C. Challenges Faced in the Cloud

Since monitors need to be located on the sources' path in passive discovery methods, this approach is hardly adaptable to the Cloud in which communication between VMs located on the same hypervisor occur, without reaching the physical network, leading to a knowledge gap in the topology discovery. On the other hand, active discovery methods tend to impose a heavy load on the network, incurring non-billable bandwidth consumption and producing traffic potentially flagged as malicious, which brought down to the Cloud scale is not desirable. Agent-based methods are not suitable given the need to install an agent on each device, especially when it is not directly under the control of the cloud provider. Due to multi-tenancy, traditional methods are not able to attribute each machine to their owner, which is a crucial addition in a security context. These limitations can be addressed in the Cloud by leveraging a centralized store, used as knowledge base in the following related works. Indeed, Madi and al. [15] focus on virtualized infrastructures and tackle the verification of compliance properties. They analyze data sources coming straigth from the virtualized environment, compared with data from the Openstack platform and an SDN controller to check proper instantiation. Bleikertz and al. [5] [4] aim to validate the correctness of instances configuration from an isolation perspective in the context of the Cloud. A differential analysis is performed when changes occur, by comparing the newly obtained model and a policy to detect potential failures. The goal of these works and ours differs. While the same tools are used in our setting and Madi and al.'s, rather than aiming to verify configuration correctness across the diverse layers of the Cloud and remaining at the level of the topology as they do, we plan to retrieve both the topology and the connectivity in real time and represent them in an exploitable format. Besides, their choice of processing the data retrieved in batch mode distances us from the real time property we expect from a connectivity builder. On the other hand, Bleikertz and al. consider the information flow in the infrastructure and address the dynamic evolution of their analysis. However the probes introduced for data retrieval are hypervisor-specific, in an environment without SDN controllers.

III. Building the Connectivity in a Cloud Environment

To retrieve the infrastructure's connectivity, we define the context considered. We then introduce the resulting challenges incurred, and an environment model to help the design of the implemented algorithm.

A. Context

We consider a cloud infrastructure in which we adopt the standpoint of the cloud provider. Networking is handled by an SDN controller for dynamic network configuration. This interaction is implemented via an existing application in the SDN controller, responsible for configuring the network as defined by the CMS. As a result, the CMS network configuration view is contained in the SDN controller. While the CMS presents management interfaces to both the cloud provider and the tenants, the SDN controller is exclusively managed by the cloud provider. The administration of the virtual infrastructures is delegated to the CMS, which interacts with the SDN controller to provision the tenants' networks. Multi-tenancy allows each tenant to have its own virtual infrastructure made up of a set of VMs interconnected by virtual networks. Beyond the scalability and volatile nature of the tenants' infrastructure which are characteristics of the Cloud, and need to be considered in the solution, the combination of CMS and SDN poses an additional challenge. Indeed,

SDN enables administrators to deploy applications in the SDN controller. Those applications can then, according to the programmed logic, reactively modify the flow rules on virtual switches and directly affect the topology, without providing any feedback to the CMS. It results in inconsistent topology views between the Cloud Management System and the SDN Controller. On one hand, the CMS aggregates data necessary to build the topology and connectivity, i.e. the hypervisors and their capabilities, the virtual machines' location (physical hosts), their owners as well as the networks built by the tenants and the security rules enforced. On the other hand, the SDN controller allows to determine flow rules generated by providers' applications installed on top of it, and independent from the CMS configuration. Flow rules are equivalent to routing rules authorizing or forbidding connections between virtual machines based on traffic patterns and SDN applications logic. From a network connectivity standpoint, they are the concrete realization of security policies and are ordered according to arbitrary priorities given by the developer of the applications running on the SDN controller, resulting in a hierarchical ordering of the flow rules distributed into consecutive tables installed on the virtual switches. Starting from the first table, packets are matched against flow rules in decreasing order of priority and only the action in the flow rule corresponding to the first match in the table is enforced. Examples of actions can be *drop*, *forward* or *jump to table*. In our context, one of those applications is interfaced with the CMS, and implements the necessary flow rules with an arbitrarily defined priority. A similar behavior is observed for the other applications installed on the SDN controller. Only the flow rules installed with a higher priority than the ones installed via the SDN application interfaced with the CMS are unknown to the CMS and impact the resulting connectivity. Indeed, in case of positive match with an incoming packet, the actions requested in those rules will superseed lower priorities ones (in particular, the ones from the CMS).

B. Overview of the Connectivity Extraction Process

To obtain a consistent view of the connectivity, we use data from both the CMS and the SDN controller. We rely on the CMS to obtain its vision of the topology and the connectivity, connectivity being later modified by an identification via the SDN controller of higher priority rules installed. Addressing the discrepancies between the CMS and the SDN controller requires the following steps:

1) Building the initial topology and connectivity as viewed by the CMS, using the CMS databases;

2) In the SDN controller, leveraging the provided APIs to identify the applications interfaced with the CMS and register the priorities of the flow rules they provision in each table;

3) Via the SDN controller, listing the flow rules installed, and retaining only the flow rules with higher priorities than the ones identified in Step 2. Let *FR* be this collection of flow rules;

4) Processing each flow rule in *FR*, and based on the combination of layer 3 and 4 protocols data, querying the connectivity graph to obtain the endpoints and links to modify.

This results in a coherent static connectivity graph, reconciling the CMS and SDN controller views. For this view to be maintained considering flow rules changes in the SDN controller, a monitoring application reacts to every rule update, addition or removal, determining whether the flow rule should belong to *FR*. When it does, Step 4 is repeated for the concerned flow rule.

C. Cloud Environment Model

From any Cloud Management System used in virtual infrastructures, the same building blocks can be extracted to set up the topology: *Hypervisor*, *Virtual Machine*, *Security Rule*, *Security Group*, *Tenant*, *Virtual Port*, *Virtual Router*, *Subnet* and *Network*. Networks and subnets refer to the virtualized context, while routers interconnect VMs belonging to distinct subnets. Security groups represent a collection of security rules that are applicable to virtual machines. They contain the IP range, port range, protocol (TCP, UDP or ICMP) and direction of the traffic authorized on the virtual machines. An additional option indicating traffic allowed considering the originating security group can also be provided. Let H, VM, SR, SG, T, VP, VR, S and N be the sets representing the collection of hypervisors (physical nodes), virtual machines, security rules, security groups, tenants, virtual ports, virtual routers, subnets and networks respectively. We define predicates classified in two categories: topology-related and connectivity-related. In their expression, $h \in H$, $\{vm,x,y\} \in VM$, $t \in T$, $secr \in SR$, $secg \in SG$, $vp \in VP$, $vr \in VR$, $s \in S$, $n \in N$. The topology-related predicates are the following:

- *instantiate(h,vm)*: means that the hypervisor h is the host of the virtual machine *vm*,
- *own(t,X)* where $X \in VM$ *or* $X \in S$ *or* $X \in SG$: means that the tenant t is the owner of the element X,
- *belongTo(s,vp)*: means that the virtual port *vp* belongs to the subnet *s*,
- *isLinkedTo(s,n)*: means that the network n is linked to the subnet *s*,
- *isAttachedTo(vp,X)* where $X \in VR$ *or* $X \in VM$: means that the element X is attached to the virtual port *vp*.

The connectivity-related predicates are the following:

- *contains(secg,secr)*: means that the security group *secg* contains the security rule *secr*,
- *isEnforcedOn(secg, vm)*: means that the security group *secg* is enforced on the virtual machine *vm*,
- *areConnected(x,y)*: means that the communication is possible between x and y for at least one combination of protocol, addresses and ports.

areConnected is a predicate partly deduced from the others. Two virtual machines *areConnected* if they are either on the same subnet or on subnets linked by routers, and their security rules allow communication for at least one combination of addresses, ports and protocol. Subnet information is provided

by *belongTo* and *isLinkedTo*, while reasoning on security rule applicability is permitted by the predicates *isEnforcedOn* and *contains*. The content of the rules themselves is then interpreted in order to derive connectivity among virtual machines.

D. Topology and connectivity graph construction

In this section, we provide more details on the topology and connectivity graph construction algorithm. The topology and connectivity builder process the CMs and SDN controller's data. The SDN controller comprises an SDN monitoring application, listening to flow rule events generated by other applications. Rule characteristics are stored in an SDN rule database. In parallel, the topology and connectivity builder runs in a dedicated server. It creates a static topology and connectivity by processing the CMS database and stores the obtained representation into its own graph database. Secondly, it listens to CMS-generated events to update its view. The events tracked are the creation, deletion, update and status change of virtual elements. Table I shows an excerpt table focusing on the creation event and illustrating its impact on the topology and connectivity, considering changes occurring on virtual machines. In this table, arrows represent the creation of an edge whose type is given by the label written above.

1) Static Phase: During the static phase, the aim is to establish a baseline topology and connectivity. We begin with the static topology, built with data from the Cloud Management Database. The network topology is built first, it is comprised of the interactions between virtual machines, virtual ports, subnets, networks, virtual routers, security groups, security rules and tenants. In order to optimize its generation, we leverage the multi-tenant nature of the Cloud: the network topology of each tenant is independently built by concurrent threads, allowing to parallelize the task. Once the network topology of each tenant is obtained, the relationship is established with the cloud provider's physical infrastructure by creating an *instantiates* predicate between the tenants' virtual machines and their corresponding hypervisors. After generating the static topology, the connectivity is constructed by identifying groups of machines able to communicate, due to their interconnection with routers or their belonging to a same subnet. Each machine cluster is then processed to determine the effective communication as stated by security rules contained in the security groups they depend on. This phase generates *areConnected* links as viewed by the CMS. Additionally, since flow rules provisioned by SDN applications are hierarchical, we identify the ones with a higher priority than the rules provisioned by the CMS application in the SDN controller, by querying the Flow Rules registry in the SDN controller. Indeed, these are the rules yielding discrepancies between the views from the CMS and the SDN controller. We develop a Monitoring Application installed on the SDN controller, responsible for identifying and registering these rules in a separate SDN rule database. Parameters contained in each rule allow to match the related CMS *areConnected* links and modify them according to the SDN view of the connectivity.

2) Dynamic phase: During the dynamic phase, an event listener intercepts topology-related notifications generated by the Cloud Management System to store them in a queue. Events are then processed by queue consumers to update the topology and connectivity graph with the changes induced by those notifications. The events are processed to modify the topology, as well as the connectivity reported in the graph. The SDN applications are also continuously monitored, in order to register any impactful modification caused by a rule creation, update or removal. The information gathered by this monitoring application allows to update an SDN rules database and modify the *areConnected* links accordingly.

TABLE I
EXAMPLE OF ELEMENTARY ACTIONS PERFORMED ON VIRTUAL MACHINES IN CLOUD ENVIRONMENTS AND THEIR EFFECTS ON THE TOPOLOGY

Common actions	Effects
Create	Node creation: vm Node attribute modification: vm.state=active vm $\xleftarrow{owns}$ tenant vm $\xleftarrow{isEnforcedOn}$ secgroup vm $\xleftarrow{instantiate}$ hypervisor Generation of a port create event Node creation: port subnet $\xleftarrow{belongsTo}$ port port $\xleftarrow{isAttachedTo}$ vm vm $\xleftarrow{areConnected}$ vm_x, where vm_x is an existing machine able to communicate with vm

IV. CONCLUSION

In this paper, we have identified the topology and connectivity extraction as a building block for risk management solutions in Cloud environments. By modeling the virtual environment and leveraging technologies available such as the CMS and the SDN controller, we designed a non-intrusive approach allowing to obtain an up-to-date view of tenants' architectures. This approach also addresses potential discrepancies between the states of the Cloud controller and the SDN controller, thus leading to an accurate representation of the connectivity. On the other hand, the connectivity built using data extracted from the CMS and the SDN controller has an optimal exhaustiveness in our context. Indeed, our approach being oblivious to potential software firewalls configured by tenants in their virtual machines, it results in an over-approximation of the tenants' virtual machines connectivity. We may report configured (via CMS or SDN), but non-effective connections between VMs due to a lack of visibility into tenants' virtual machines. However this is an acceptable approximation in a risk management context, as no potential connection link is left out of the representation. Besides, connectivity data is obtained reasonably fast as confirmed by performance experiments run on the algorithm. However, their results are left outside of the scope of this paper. It is hence suited to the dynamic nature of the Cloud. For future work, the most expensive operations lying in the establishment of the static connectivity, we aim to parallelize its construction to reduce building time, as we only did it for the static topology.

REFERENCES

[1] Neo4j. https://www.neo4j.com/.

[2] ONOS. http://onosproject.org/.

[3] State Of The Cloud Report. Technical report, RightScale, 2017.

[4] Sören Bleikertz, Thomas Groß, Matthias Schunter, and Konrad Eriksson. Automated Information Flow Analysis of Virtualized Infrastructures. In *Proceedings of the 16th European Conference on Research in Computer Security*, ESORICS'11, pages 392–415, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] Sören Bleikertz, Carsten Vogel, and Thomas Gross. Cloud Radar: Near Real-Time Detection of Security Failures in Dynamic Virtualized Infrastructures. In *Annual Computer Security Applications Conference (ACSAC)*, 2014.

[6] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous IP networks: the NetInventory system. *IEEE/ACM Transactions on Networking*, 12(3):401–414, June 2004.

[7] Benoit Donnet, Timur Friedman, and Mark Crovella. Improved algorithms for network topology discovery. In *International Workshop on Passive and Active Network Measurement*, pages 149–162. Springer, 2005.

[8] Benoit Donnet, Philippe Raoult, Timur Friedman, and Mark Crovella. Efficient algorithms for large-scale topology discovery. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 327–338. ACM, 2005.

[9] Brian Eriksson, Paul Barford, and Robert Nowak. Network Discovery from Passive Measurements. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 291–302, New York, NY, USA, 2008. ACM.

[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-level traffic measurements from the sprint IP backbone. *IEEE Network*, 17(6):6–16, November 2003.

[11] Fotis Georgatos, Florian Gruber, Daniel Karrenberg, Mark Santcroos, Ana Susanj, Henk Uijterwaal, and René Wilhelm. Providing active measurements as a regular service for isps. In *PAM*, 2001.

[12] Salim Jouili and Valentin Vansteenberghe. An empirical comparison of graph databases. In *Social Computing (SocialCom), 2013 International Conference on*, pages 708–715. IEEE, 2013.

[13] Richard Paul Lippmann and Kyle William Ingols. An annotated review of past papers on attack graphs. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2005.

[14] Bruce Lowekamp, David O'Hallaron, and Thomas Gross. Topology Discovery for Large Ethernet Networks. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 237–248, New York, NY, USA, 2001. ACM.

[15] Taous Madi, Suryadipta Majumdar, Yushun Wang, Makan Pourzandi, and Lingyu Wang. Auditing security Compliance of the Virtualized Infrastructure in the Cloud: Application to OpenStack. In *6th ACM Conference on Data and Application Security and Privacy ACM CODASPY 2016*, 2016.

[16] D McRobb, K Claffy, and T Monk. Skitter: Caida's macroscopic internet topology discovery and tracking tool, 1999.

[17] S. Noel, S. Jajodia, and A. Singhal. Measuring security risk of networks using attack graphs. *International Journal of Next-Generation Computing*, 1(1), 2010.

[18] X. Ou and A. Singhal. *Quantitative Security Risk Assessment of Enterprise Networks*. Springer, 2012.

[19] William Tu, Priya Thangaraj, Jui-hao Chiang, and Tzi-cker Chiueh. Automated service discovery for enterprise network management, 2009.