Secure migration of virtual SDN topologies

Fabien Charmet Gregory Blanc
{fabien.charmet,gregory.blanc}@telecom-sudparis.eu
Télécom SudParis, Institut Mines-Télécom, CNRS SAMOVAR UMR 5157, Evry, France

***Abstract*—With the emergence of Software Defined Networks (SDN), new virtualization techniques have appeared (e.g., FlowVisor [14]). Traditional hypervision has attracted a lot of attention with respect to resource sharing and multi-tenancy. Cloud providers have usually a solid knowledge on how to manage computing, memory and storage resources, but often lack the ability to properly manage network resources. Thanks to OpenFlow, a widespread SDN southbound interface protocol, virtualizing the network infrastructure has become possible. However, network virtualization also comes with its own security issues ([5], [6]). In this paper, we focus on the security aspects related to the migration of virtual networks. After providing a brief overview of the technological scope of our work, we review the state of the art of the migration of virtual resources. Finally, we conclude with our current results and the prospective outcomes we expect to obtain.**

***Index Terms*—Software Defined Networking, Virtualization, Network Migration, Security**

I. Introduction

Virtualization is the cornerstone of performances optimization for both hosts and networks, entailing important cost reductions for service providers. Cloud computing itself heavily relies on this technology. Host virtualization makes it possible to provide multi-tenant isolated services, while network virtualization allows Cloud providers to easily reconfigure the topology in order to meet dynamic workload requirements and ensure elasticity [11], thus allocating part or the entirety of the physical network resources to the virtual network (VN). One of the emerging technologies that enables network virtualization is Software Defined Networking (SDN), where data plane and control plane are decoupled. In a SDN architecture, the control plane is managed by a software element called the controller, that interacts with applications and reconfigures the physical switches and routers to meet dynamic needs. A central operation in network virtualization is migrating the VN, which consists in remapping the virtual nodes and links over the physical network. Migration is a common operation used to manage the network in case of failure, workload adaptation and even as a security counter-measure (Moving Target Defense [12]) However, to the best of our knowledge, no research work has yet focused on the impact of the VN migration on security properties in a SDN environment. It is crucial to maintain a certain level of security during the migration to prevent data leakage or corruption. The remainder of this paper is structured as follows. Section II presents the technological background of this work. Section III presents a state of the art of the recent works on migration and security topics. Then, section IV details the components in the architecture. Finally, section V presents the firsts results we published.

II. Background

In this section, we detail the different technological aspects underlying this work. We focus on Software Defined Networks (SDN) as the networking technology, and more precisely we work with OpenFlow as it is the leading interfacing protocol leading the research and the industry.

A. SDN & OpenFlow

With the growth of Internet infrastructures and needs, typical networking technologies tend to fall behind in terms of scalability and resource sharing, among other aspects. From that observation, McKeown et al. proposed OpenFlow [10], a simple way to decouple the data plane from the control plane, with the objective to ease and accelerate the development of network protocols and applications, such as traffic engineering or security. The OpenFlow-enabled architecture features a specific component responsible of controlling the data plane, aptly named controller. Placed at the control plane, it instructs the networking elements of the data plane on deploying and enforcing the paths to be taken by the traffic. OpenFlow provides a vendor-agnostic API that allows for heterogeneity and hide the underlying implementation to the end user, thus easing the use of networking applications. In OpenFlow, the routing information on how a packet should be handled is called a *flow rule*. Based on a match-action pattern, every packet of a flow matching the criteria of a rule will be processed according to the associated action (modification/forwarding/drop).

B. Virtualization

Virtualizing networks consists in abstracting the topology view and routing policy from the physical network topology. The abstracted view is presented to the viewer and the virtual resources will be mapped with the underlying infrastructure. There are four different types of mapping: *one-to-one*, *one-to-many*, *many-to-one*, *many-to-many*.

One-to-one mapping consists in having only one virtual element mapped with one physical element. This is a counter-intuitive abstraction, as it defeats the purpose of sharing resources by virtualizing them.

One-to-many mapping is an abstraction also named the "Big Switch" abstraction. The end user only sees one virtual node as a network topology, while the whole processing of packets

is done by several equipments. This use case is convenient when, for instance, a user wants to exploit cloud resources without dealing with the burden of network management and failure recovery. For instance, Ghorbani et al. [9], [7] propose a solution to maintain the correct behavior of a virtual network element using the Big Switch abstraction.

Many-to-one mapping is similar to traditional virtualization. Several virtual resources will be supported by only one physical element (e.g., virtual disk, memory sharing).

Many-to-many mapping is the combination of the previous abstractions. For instance, the hypervisor VeRTIGO [4] provides both abstractions Each end user is allowed to choose which representation of the infrastructure he wants.

III. State of the Art

To the best of our knowledge, there are few works that focus on the security of VN migrations, probably due to the fact that it is a recent field of study. Ghorbani et al. have provided in [8], [9], [7] some leads on defining properties related to the migration, and on verifying and maintaining these throughout the process. On the contrary, literature provides plenty of content when it comes to *live virtual machine migration*.

A. Live VM Migration

For years, virtual machines (VMs) have been the target of a number of attacks, including *side-channel* ones. Simply put, a side-channel attack exploits the weaknesses existing in the implementation of the hypervisor. For instance, Oberheide et al. [13] corrupt the memory pages of a VM during its migration and change the values in the memory of a the current process being executed. During the live migration of a VM, Achleitner et al. have characterized the migration traffic and proven that an attack would then be able to disrupt the migration. They provide a stealth framework that renders the migration undetectable by dynamically generating traffic noise that would change the characteristics of the global traffic. In order to compromise the VM, an attacker needs to be hosted on the same physical equipment as the target VM. Literature describes this as collocation or co-residency. We will use the latter term in the remainder of this paper. Co-residency is studied in [1], [2], [12], [13]. Co-residency is the root cause of side channel attacks [12] and reducing its duration is the best way to prevent information leakage or integrity attacks [1], [2], [13]. Both [2] and [12] propose a migration framework to reduce co-residency as a Moving Target Defense (MTD) technique.

B. Virtual Network Migrations

Different from the above-mentioned works, we consider here the migration works where OpenFlow supports network virtualization. Precisely, we are interested in the works which purpose is to maintain particular properties throughout the migration. For instance, LIME [8] provides transparency during the migration. They consider a formal approach to determine whether or not the migrated virtual network behaved as if it hasn't been migrated. This approach considers the observable events that can occur during the migration. If a migrated network does not produce any event a non migrated network would not, then the migration is transparent. This approach is then applied to another problem, the Big Switch correctness. In [9], [7], the behavior of a one-to-many mapping is discussed. The one-to-many mapping implies that the traffic may take a path in one way and use a different one on the way back. This could lead to discrepancies at the application level, due to the delay in the traffic and differences of entry points. In order to prevent that, COCONUT [7] provides algorithms to deploy flow rules on the different network elements that will preserve the correctness of the behavior.

IV. Secure migration of virtual networks

In the previous sections, we have presented different works based on the migration of VMs or the consistency of virtualization. However, to the best of our knowledge, there is no work that focuses on the security properties of a VN migration. In this section, we present the global architecture of our work, and our objectives.

A. Problem Statement

Attacks on the migration process, that transfers virtual resources from a set of physical resources to another, have been demonstrated in [1]. Therefore, it becomes necessary to ensure a certain level of security of the migration of a virtual network, in order to prevent any information leakage or service disruption. In our work, we consider the modeling of security properties to characterize the migration process. In particular, we studied a subset of these, how to preserve these and how to verify the fact that they are actually preserved.

B. Road map

The problem statement highlights three main steps for our work, namely formalization, preservation and verification.

Formalization: We propose different definitions for the properties we want to preserve, for instance confidentiality, integrity, etc. We use first-order logic predicates to model the conditions representing each property. The parameters related to a networking environment are of the following nature: user, information, physical network device/link, virtual network device/link. The predicates can represent a user accessing an network element or the authorization given to a user to access a particular

Preservation: The security properties we want to preserve require a different treatment for each one of them. It is possible to implement these treatments in sub-modules where each would take care of a specific task. A module would be in charge of generating traffic noise to hide the migration of the virtual networks. Another would provide alternatives in case a change in the infrastructure corrupts an ongoing migration. On a technological point of view, the implementation of these sub-modules are software components inside the network hypervisor.

Verification: When counter-measures are deployed, it is necessary to evaluate their impact. The verification is implemented

by having a monitoring solution combined to a theorem prover. Verification is either at run-time or as a post incident analyzer. Given the trace of the network events during the incident, the analyzer is able to correlate the different elements and detect where and when the breach happened.

C. Threat model

The migration of a virtual topology (VT) creates nodes, virtual links, and establishes the links between the nodes. We will describe the different components of considered threats using the following syntax : Attacker - Action - Target.

1) Attacker: We consider the different locations from where an attacker could attack the migration of the VT: the insider, the collocated user, the provider and the outsider. The insider is an attacker located inside the VT that is going to be migrated. The collocated user is another user of the virtualization platform. The provider is the owner of the physical infrastructure. The outsider is a malicious user outside the SDN environment.

2) Action: There are 4 major actions that can be performed by an attacker to affect the VTs and their environment. The first two are related to the infrastructure and the others are related to the data carried across the network.

- Unauthorized access to a network element.
- Disruption of service on a network element.
- Information disclosure without authorization.
- Information modification or destruction without authorization.

3) Target: We can consider two types of targets: nodes/links or data. The nodes/links are either the physical switches/links in the infrastructure or the virtual nodes/links of a VN. There are three types of data: the user data carried by the virtual elements, the management data and the configuration data carried by the infrastructure. The user data is the traffic generated by the user and going through the VN. The management data is made of the packets sent to operate the SDN equipments properly. The configuration data are the different flow rules deployed on the physical switches when instantiating a VN.

V. Preliminary results and future work

In [3], we have presented our first results on the formalization and verification components. We proposed a formal definition for the *confidentiality* property inside a networking environment. We integrated this formal definition with SNARK[15], a theorem prover capable of pinpointing the events responsible for the security breach. SNARK expresses conditions as first-order logical predicates, and embeds a temporal reasoning facility, which we use to represent the ordering of the network events and deduce their consequences. As a future work, we will work on the preservation component. Section III detailed several aspects and properties of a live migration that are worth considering. For instance, the different solutions presented for the live VM migration are interesting to apply in a live VN migration context. Indeed, the works on co-residency have shown that it is possible for an attacker to infiltrate an infrastructure and realize both passive and active network monitoring. This leaves space for side-channel attacks inside a virtual network.

References

[1] Ahmed Atya, Azeem Aqil, Karim Khalil, Srikanth V Krishnamurthy, and Thomas F La Porta. Stalling Live Migrations on the Cloud. (Llc).

[2] Ahmed Atya, Zhiyun Qian, Srikanth V. Krishnamurthy, Thomas La Porta, Patrick McDaniel, and Lisa Marvel. Malicious co-residency on the cloud: Attacks and defense. *Proceedings - IEEE INFOCOM*, 2017.

[3] Fabien Charmet, Richard Waldinger, Gregory Blanc, Christophe Kiennert, and Khalifa Toumi. Preserving Confidentiality during the Migration of Virtual SDN Topologies: A Formal Approach. In *IEEE International Symposium on Network Computing and Applications*, 2017.

[4] Roberto Doriguzzi Corin, Matteo Gerola, Roberto Riggio, Francesco De Pellegrini, and Elio Salvadori. VeRTIGO: Network virtualization and beyond. *Proceedings - European Workshop on Software Defined Networks, EWSDN 2012*, pages 24–29, 2012.

[5] Victor T Costa and Henrique M K Costa. Vulnerability Study of FlowVisor-based Virtualized Network Environments. *2nd Workshop Network Virtualization Intelligence Future Internet, Rio de Janeiro, Brazil*, 6994:6994, 2013.

[6] Victor T Costa and Lus Henrique M K Costa. Vulnerabilities and solutions for isolation in FlowVisor-based virtual network environments. *Journal of Internet Services and Applications*, 6(18):9, 2015.

[7] S Ghorbani and P B Godfrey. COCONUT: Seamless scale-out of network elements. *EuroSys 2017*, pages 32–47, 2017.

[8] S Ghorbani, C Schlesinger, M Monaco, E Keller, M Caesar, J Rexford, and D Walker. Transparent, live migration of a software-defined network. *Proceedings of the 5th ACM Symposium on Cloud Computing, SOCC 2014*, 2014.

[9] Soudeh Ghorbani and Brighten Godfrey. Towards correct network virtualization. *ACM SIGCOMM Computer Communication Review*, 44(4):99, 2014.

[10] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69, 2008.

[11] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *National Institute of Standards and Technology, Information Technology Laboratory*, 145:7, 2011.

[12] Soo-jin Moon and Michael K Reiter. Nomad : Mitigating Arbitrary Cloud Side Channels via Provider-Assisted Migration. *Ccs '15*, (1):1595–1606, 2015.

[13] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Empirical exploitation of live virtual machine migration. *Proc. of BlackHat DC ...*, (Vmm), 2008.

[14] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. FlowVisor: A Network Virtualization Layer. *OpenFlow Switch Consortium, Tech. Rep*, pages 1–13, 2009.

[15] ME Stickel, RJ Waldinger, and VK Chaudhri. A guide to SNARK. *SRI International, Menlo Park, ...*, (March), 2000.