SCTE·ISBE CABLE-TEC EXPO 2017
DENVER, CO
OCTOBER 17-20

2017 Fall Technical Forum
SCTE·ISBE • NCTA • CABLELABS

Security of Open Distributed Architectures

Yet Another SDN and NFV Security Paper

A Technical Paper prepared for SCTE/ISBE by

Steve Goeringer
Principal Architect
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
s.goeringer@cablelabs.com

Dr. Indrajit Ray
Professor
Computer Science Department, Colorado State University
Fort Collins, CO 80528
Indrajit.Ray@colostate.edu

Table of Contents

List of Figures

List of Tables

Introduction

Our networks are continuously evolving and there are lots of neat ideas out there on how this evolution should be done. There are lots of ideas including software defined networking (SDN), network functions virtualization (NFV), development operations (devops), containers, virtual machines, just to name a few. All of these complex ideas are ultimately about automating the deployment and delivery of network services over open distributed architectures. Network operators are avidly reinventing themselves using open distributed architectures to reduce cost, accelerate service velocity, and enable new types of services. However, these technologies also present new security challenges — our traditional ways of addressing network security must evolve. This paper briefly reviews the unique security challenges and opportunities network operators face as they apply these technologies. After a brief introduction, the paper discusses some of the emerging challenges. These are organized into two parts: larger and obfuscated attack service; and new risks. This is followed by discussion of the opportunities operators can leverage to use these technologies to improve security. The paper concludes with a review of proactive actions currently available to operators.

Security of Open Distributed Architectures

1. Frame of Reference

Software defined networking, network function virtualization, virtual machines, continuous integration, containers, development operations… There are so many terms and buzzwords used to describe the ideas being applied to evolve networks today. In aggregate, these technologies create open and distributed network architectures that support services programmatically. The result is complete reinvention of our networks where now we architect network factories that churn out services at an unprecedented rate.

The technologies network operators use to achieve open distributed architectures are complimentary. First, we started with programmatically controlling the information flows of our networks by applying software defined networking (SDN) [ONF 1]. SDN used controllers as an interface to application logic to programmatically implement flows between switching elements. The most common SDN implementation uses OpenFlow® [ONF 2]. Then, we started to deploy and manage entire virtual resources programmatically using network functions virtualization (NFV). Through NFV, we programmatically orchestrate deployment of virtual network functions (VNFs) which comprise of one or more workloads distributed across multiple general purpose servers [ETSI NFV 1]. The common thread between SDN and NFV, synergistic technologies, is automation of network engineering and operations on an industrial scale, completely transforming the way network operators fulfill their customer's needs. An emerging technology, network slicing, promises to go one step further – to beginning orchestration of deployment and maintenance of entire virtual overlay networks [NGMN 5G].

It's important to recognize the IT revolution behind SDN, NFV, and slicing. Agile engineering, continuous integration, development operations, virtual machines, containers, and other evolutions in computer science disciplines and tools fuel the evolution of ***open distributed architectures***. In synthesis, this all combines to create entire, automated network supply chains, binding operators and their technologies to vendors and software companies closer than ever realized in the past. This paper will use the phrase open distributed architectures to reflect the whole technologies that are emerging to allow our networks to be more adaptive and agile in programmatic ways. The focus of this paper is discussion of gaps in securing developing open distributed architectures. Certainly, new and emerging architectures

must represent new risks – the fabric of developing networks is fundamentally different than that of our legacy "big iron switches and routers". So, it must follow they have different vulnerabilities.

2. What is security?

First, let's readdress why we need to address security at all and then reaffirm what security goals must be, at least from the perspective of a service provider. This may seem obvious, but perhaps there are important subtleties that are often overlooked.

Network operators and their supply chain partners work very hard to engineer very specific experiences for their users – experiences that are repeatable and provide unique value to their users. The infrastructure and software used to provide great experiences has inherent value. Usually, the components upon which services are built are general purpose and can be used to support many kinds of service functions. And, of course, they must be interconnected to other components and ultimately to the end users. Therefore, it is axiomatic that network resources have value and, to at least some degree, those resources must be exposed to be usable. Since these resources have value and must be exposed, others – our adversaries – seek to harness the value of network resources to provide services other than intended.

This is a fundamental subtlety that seems largely overlooked. When hackers or other cyber criminals attack and compromise a network resource, they change the experience of end users. Often, this may be done in a way that end users may not even realize. Sometimes, the impacts are quite apparent – the services may not work, users' confidentiality or privacy are compromised, or worse. On a macro scale, network hacking really hijacks, or at least subverts, network operators' supply chain – it redirects the entire value for which a service was intended to benefit another part.

What then should the goal of the security engineer be? Certainly, it's not to implement perfect security. Perfect security – making it impossible to misuse any resource by any party that is not authorized – is impractical (and probably impossible). To have value, networked resources must be exposed. The less exposed they are, the harder they tend to be utilized. Rather, the goal should be to make using a networked resource expensive to misuse, while being inexpensive to use by authorized users for the intended purposes. This orientation recasts the focus of security on managing the total life cycle cost of delivering a targeted experience.

3. Security challenges of open distributed architectures

The security of open distributed architectures is challenging on several fronts. There are many, many stakeholders – customers, service providers, vendors, and in some cases regulators and other government agencies (such as law enforcement who require support for lawful intercept). Even within a service provider, multiple organizations and even business units may be engaged. And, of course, there is lots of complexity which is easily seen in papers from the Open Networking Foundation (ONF) [ONF Primer] and the European Technical Standards Institute (ETSI) [ETSI NFV 1]. Multiple stakeholders responsible for lots of complexity create an environment with a much larger attack surface than legacy networks possessed.

However, the programmatic nature of evolving networks must also be considered. The attack surface can be obfuscated because SDN and NFV networks enable a great deal of abstraction. Abstraction makes complex things appear less complicated. One example of abstraction is the use of application programming interfaces (APIs). When the application programming interfaces (APIs) used to access

layers of abstraction are well implemented, the APIs can significantly improve security. However, they can also hide security problems and make monitoring the security of running processes hard to measure. Moreover, these network tools can be applied recursively. In SDN, for example, you may have an SDN controller of multiple SDN controllers which may in turn control other SDN controllers. In NFV, VNFs can nest VNFs which in turn may be distributed across multiple containers or virtual machines. The result is that security vulnerabilities may be recursively exposed.

The programmatic nature of these network technologies is driven by various forms of data models. How this works varies per implementation, but there may be model driven service abstractions, templates, scripts, configuration files, etc. Consider that now our networks can be hacked simply by hacking the models used to create them.

A major differentiator between data center oriented and network virtualization is how service state is distributed across so many elements. At a minimum, service may involve a cable modem, a set top box (which may have a cable modem integrated into it for non-linear video requiring broadband functions), an access point (particularly for shared or public WiFi), the cable modem termination systems (CMTS), a variety of service and management servers (for configuration, software download, address grants, etc…), and more at the core level. Often, the nature of service state and management across such a complex access architecture is forgotten. Transitioning this architecture to a virtualized model and ensuring predictable and manageable service will be challenging.

SDN poses specific challenges [ONF 3]. At a minimum, the attack surface is increased by the introduction of the SDN controller. The server and application software implementing the controller present new vulnerabilities. For example, the Southbound OpenFlow® and NETCONF [RFC 7803] interfaces and the Northbound RESTful APIs are required to enable meaningful services. Moreover, the model driven service adaptation level (MD-SAL) introduces new types of data elements (files and entries in files) [ODL]. In addition, legacy management and operations interfaces must still be supported, along with any proprietary interfaces. Control plane and data plane separation are changed, and partial or full data plane packets may be sent to the SDN controller for analysis and processing to create new flow table entries.

NFV also presents specific challenges [ETSI NFV 2]. At the most fundamental level, NFV transitions functions that used to be achieved using dedicated hardware to software running on general purpose computers (at least, that's the theory). This means that our service infrastructure is now software based. Consequently, software vulnerabilities become infrastructure vulnerabilities. Also, NVF is very complicated – it introduces lots of new functions and approaches to the way we implement network services. The attack surface, like with SDN, becomes larger relative to legacy network implementations as we introduce additional controllers, orchestrators, hypervisors, virtual machines or containers, application stores (and more). Management and control of this complicated architecture requires lots of layers of abstraction, which can obfuscate security vulnerabilities. Moreover, these abstracted vulnerabilities can cascade through the entire architecture as, for example, a compromised template for virtualized network function (VNF) can create vulnerabilities in forwarding graphs which in turn may introduce vulnerabilities to additional VNFs that are part of service chains [Lee].

Unfortunately, focusing solely on SDN and NFV really misses the point. The point of network evolution seems to be focused in automation. So, open distributed architectures also incorporate IT technologies for agile service development and deployment including continuous integration and deployment, live builds, development operations, and more. The result is a new network infrastructure which allows operators to rapidly integrate services as a highly automated supply chain. This supply chain bridges both

development and production networks. This means now we have to secure software development and integration components as part of our production network for service delivery. This includes our tools sets for bug and feature tracking, software build environments, code integration, package management, configuration, testing, and more.

4. Some critical needs

Given the evolutionary path we are on, what must we achieve in terms of security? Our goal is simply to ensure all things done with future networks are done securely. This sounds trite, of course, but that is the goal. The challenge is the scope is audacious and the depth is daunting. In terms of scope, we want to ensure all activities related to developing and integrating our virtual network components and the infrastructure on which those components will be deployed is secure. The images, models, and data repositories used to store configurations must be secure and the loading of those resources into runtime must also be secure. Then the run time of virtual network functions themselves must also be secure. Again, the scope is audacious (see Figure 1).

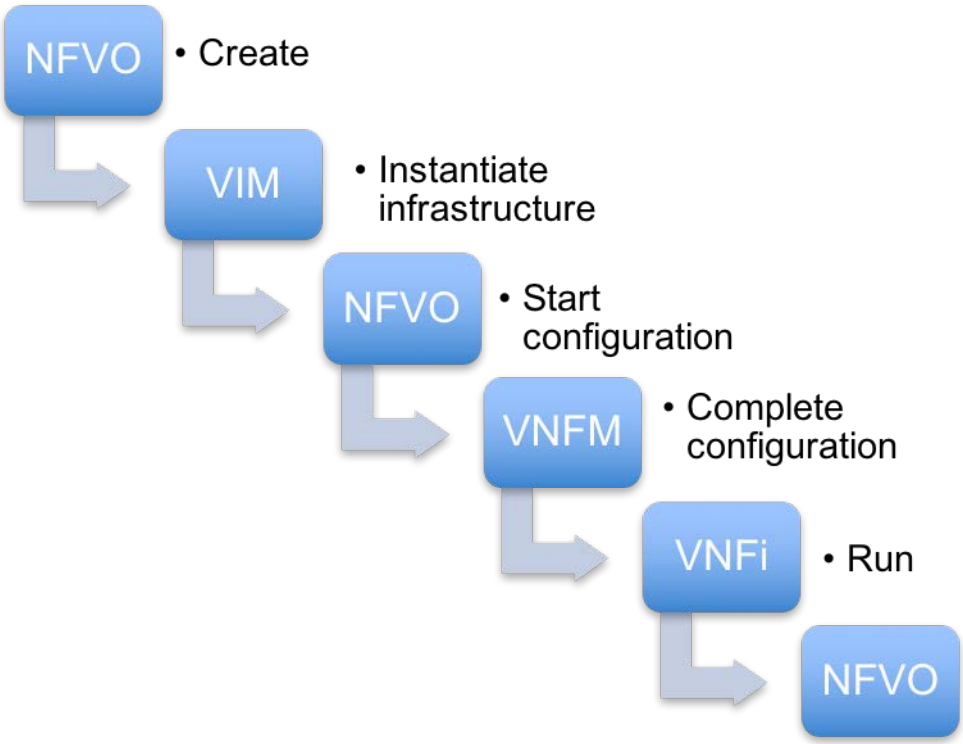

Figure 1 - VNF development and deployment process

Of course, “everything” involves significant detail. Hardware must be discovered and identified, as must software elements. The hardware must be jumpstarted and made available as infrastructure components, whether as part of a pod, cluster, cloud, or whatever architectural concept is used. The VNFs themselves must be developed and once well integrated and understood, stored in reference implementations (gold configuration) or images in catalogues. When customer demand justifies a new network element, onboarding a VNF must occur, bootstrapping the new VNF from the catalogue or image repository, configuring it for the specific need at hand, and then integrating into the actual routing and switching

flows of the current running network. Once the VNF is up and accepted, life cycle management processes such as maintenance, migration, restarting, and recovering must be supported. Once the VNF is no longer needed, it must be terminated (turned down). And all of this may need to have corresponding customer facing activities through operational and business support systems. And it must all be done in a secure way, including strong attestation (which ultimately means the run time result is provably secure).

In efforts to pursue creation of an end-to-end proof-of-concept of this entire vision have been challenging. When considering OpenStack, OPNFV, ONAP, and FD.io as possible solutions, much of their security focus has been on vulnerability management of the code base themselves. This is not to undervalue the good work their various security working groups have labored to achieve. They have worked hard to provide a good basis for security. However, the result is largely flows from the Linux legacy on which these solutions are built. They're all Linux based. Consequently, security settings and implementation options are distributed as flags and fields in lots of files supporting dozens – sometimes hundreds – of processes that implement any given VNF. Encryption and authentication keys are often stored unencrypted in these files in addition to various access control indicators and other security controls. While OpenStack™ and OPNFV do have excellent security guides, they are insufficient as hardening guides. They advise what security features exist, but don't advise how to configure security against specific threats. Moreover, no configuration and security validation tools are optimized for validating security of SDN or NFV infrastructure. Never-the-less, a diligent and persistent security engineer probably can reasonably lock down SDN and NFV infrastructure and applications – it's just going to take a great deal of effort and incremental validation.

There are three critical areas where open distributed architectures are falling critically short. When SDN, NFV, and slicing are combined with continuous integration and development operations, the result is a dynamic supply chain. Supply chains are networks in and of themselves. We tend to think of them in terms of trucks and warehouses, but in the information age, they use telecommunications networks and servers to develop and deliver digital goods. These digital supply chains reach into both our development and production networks. Consider MANO, OpenDaylight and the model driven service delivery it considers, our devops tools like Puppet and Juju and dozens of others, SDN and NFV application stores, and VNF catalogues. These resources reach into our networks, but few security engineers have fully considered how cyber supply chain risk management must be approached differently than traditional network security risk management. How do we patch this kind of architecture? How do we assert a consistent security policy across this environment? Code validation and automated network security management must be integrated. Also, open distributed architectures should apply a zero trust model for all network functions.

Consider the ETSI NFV reference model illustrated in Figure 2. The model shows how virtualized network functions will overlay an infrastructure layer and how all the various NFV elements must use a wide range of reference point connections for inter function communications. This includes connectivity to OSS/BSS support and Management and Orchestration functions. Three types reference points are shown – execution, other, and main NFV reference points. Only the NFV reference points are specified, with execution and other reference points waiting to be specified or judged out of scope. Even, so, the result is extremely complicated (see Table). This does not even include other interfaces to support connectivity SDN controllers, the VNF catalogue, instrumentation and telemetry services, container stores, devops application, and more. How can these multiple connections be sufficiently specified and architected to support a zero trust architecture? Simplification.

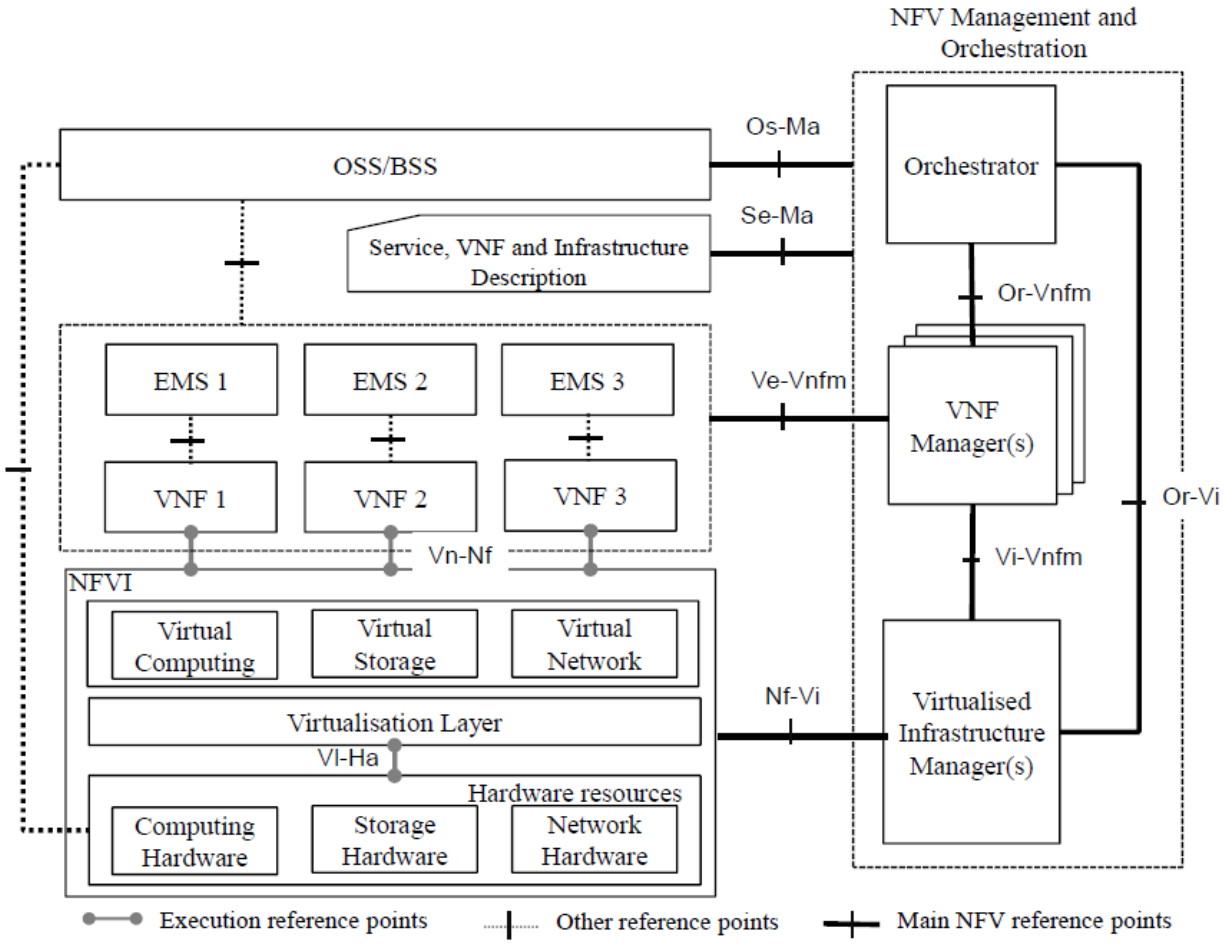

Figure 2 - ETSI NFV reference architectural framework

Three types reference points are shown in the ETSI NFV reference architecture framework – execution, other, and main NFV reference points. Only the NFV reference points are specified, with execution and other reference points waiting to be specified or judged out of scope. Even, so, the result is extremely complicated (see

Table **1**). This does not even include other interfaces to support connectivity SDN controllers, the VNF catalogue, instrumentation and telemetry services, container stores, devops application, and more. Ensuring secure implementation of all these interfaces can be difficult.

Table 1 - ETSI reference points

Reference point classification	**Reference point**	**Terminating entities**	
Main NFV reference points	Os-Ma	MANO	OSS/BSS
	Ve-Vnfm	VMF-Manager	EM or VNF
	Nf-Vi	VIM	NFVI
	Or-Vi	VIM	NFV Orchestrator
	Vi-Vnfm	VIM	VNF-Manager
	Or-Vnfm	VNF-Manager	NFV Orchestrator
Execution reference points	Vi-Ha	Hardware resources	Virtualisation layer
	Vn-Nf	VNF	Network Function
Other reference points	Not specified	EM	VNF
	Not specified	OSS/BSS	EM/VNF
	Not specified	OSS/BSS	HW resources

Open distributed architectures need to focus on establishing security associations between functions and devices rather than simply interfaces (as reflected by a standard reference point). Such a security association must be:

- Based on strong identity: Identity is the basis for any meaningful trust system. Identity should be based on a secret paired with a unique identifier. The identity must be attested by a certificate or equivalent by signing or equivalent cryptographic operation. The certificate may contain other information (but not any information that should be changed such as software versions).

- Authenticated: Each security association must be verified when the association is requested using a cryptographic challenge.
- Authorized: Once entities have validated their mutual identities, their resource or activity accesses must still be authorized. Authorization should be based on a system or service wide policy system. The policy system should assume a least privilege orientation and assure separation of duty and function. Implementation may use a policy lookup or token grant approach.
- Isolated: Isolation of network, storage, and compute resources used for specific workloads must be assured. There are a wide range of obvious security risks that are managed this way, however, it is equally important from a performance perspective. Specifically, workloads or process should not impact other workloads or processes unless allowed by the operator. Isolation may be achieved by network segmentation (through secure addressing or encapsulation) and various virtualization tools for ensuring workload isolation in memory, CPU, and storage.
- Confidentiality: Data and communications should be kept private. The isolation functions discussed above may achieve sufficient confidentiality. However, encryption will ensure even stronger confidentiality, assuming adequate protection of encryption keys.
- Attested: Finally, all the security controls that implement a security association and protect it must be provably untampered. This is traditionally done using accounting and logging mechanisms. There are improvements in trusted computing systems that allow secure boot and run time monitoring to improve on legacy approaches. Whatever specific strategies are used, the goal must be to verify that the infrastructure and the security associations implemented to interconnect both hardware and software components are, indeed, what they are expected to be.

These ideas are illustrated in Figure 3. When implemented as a whole, they provide a secure communication channel (security association) between elements. The security association itself is attestable and can be managed according to network wide policies. Every interface in an open distributed architecture – whether link layer (physical) or network layer (logical) should be implemented as a security association.

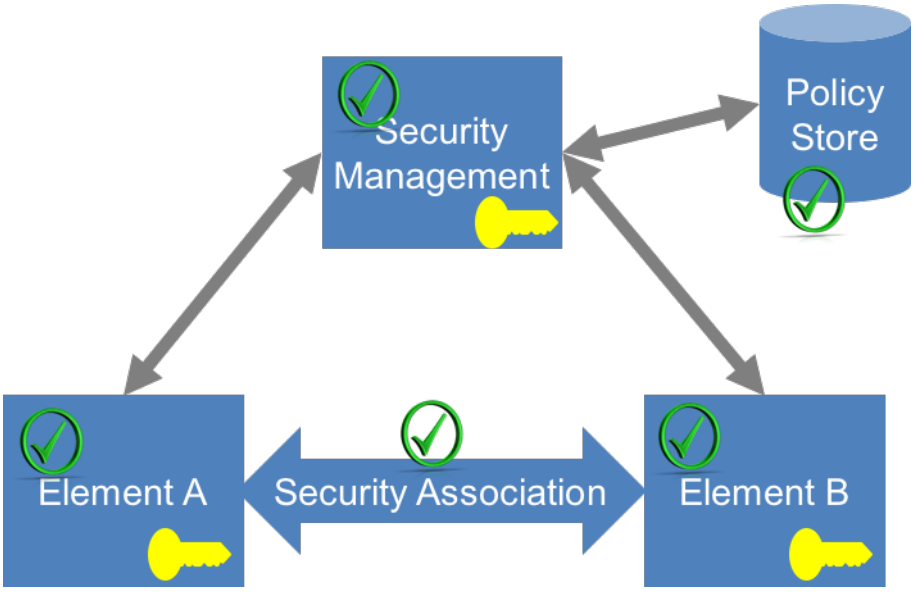

Figure 3 - Security Association

The characteristics outlined above are left intentionally non-specific. Details will depend on whether the trust system uses PKI, symmetric keys, or tokens. Further details may depend on how secrets are issued and stored (such as through use of a Trusted Platform Module or Hardware Security Module).

Applying identity to evolving technologies is proving challenging. Security professionals working on IoT, connected healthcare, and virtualized network technologies are all finding that establishing a strong, evolvable and scalable identification mechanism suitable as a basis for ecosystem wide trust is challenging. A secure identity is comprised by at least three elements. First is a secret known to the entity against which the identity is associated. This is must often a private key. This secret is the basis for trust in the ecosystem in which the entity participates. In asymmetric cryptographic trust systems, the private key will have an associated public key and the private key should not be known to any other element in the system. In symmetric cryptography trust systems, the key must be known by at least one other element (such as an authentication center). This secret must also be immutably bound to a unique identifier (within the scope of the ecosystem). This immutable binding may be achieved by digitally signing the identifier and other attributes and information using a public key certificate [X509][RFC 4158] or an external validator such as an authentication center.

One of the reasons identity is challenging is that to be most useful identities must be persistent. Persistence, however, comes at a price. It usually means identity must be issued by some central authority, which can create friction in the supply chain that supports distribution of the devices or software elements being identified. The central authority can also represent a risk factor in cyber security and supply chain terms. Moreover, there must be some way of revoking persistent identities, which requires processes and resources that ultimately are overhead. Multiple schemes are available to fluidly and dynamically assign identity, but these are problematic as they increase the attack surface of the ecosystem the identity supports. Dynamic identities also are hard to associate to security policies. Token

based systems attempt to mitigate this deficiency. However, while many token systems are reasonable authorization schemes, they are insufficient for authentication. Tokens must be validated as authentic before they are used to authorize actions or use of resources. Otherwise, tokens can be misused by third parties. A superior identity system suitable for ecosystem trust might require a combination of public key (asymmetric), symmetric (such as specified by 3GPP/GSMA™), and token based technologies. The challenge is implementing such a hybrid with minimal complexity – otherwise, the hybrid will likely increase the attack surface of the ecosystem without sufficient gain in security functionality.

5. Opportunity

The scope and depth of new threats and attack vectors related to open distributed architectures should be neither exaggerated nor understated. There is both risk and opportunity. There is the opportunity to improve security relative to legacy infrastructure, mostly as a consequence of the programmatic nature of emerging networks. The improved potential of automation provides the opportunity for more consistent execution of security processes and controls. Moreover, automation and more consistent execution of interfaces and management protocols may make it easier to upgrade and patch the network as security threats evolve and develop.

Both SDN and NFV also promise specific advantages. As NFV is based largely on general purpose hardware, it may be much easier to enable cryptographic functions. Pervasive support for encryption may be possible much more cost effectively than in the past. Also, using general purpose hardware may provide improved support for consistent and well implemented authentication. Finally, both SDN and NFV promise to support very granular control of packet monitoring and again more cost effectively than in the past.

Conclusion

This paper has reviewed some issues to realizing secure virtual infrastructure and software based services. While there are challenges, there are also opportunities that promise a future where our networks are easier to secure. However, we need to address at least three gaps. We need a consistent approach to identity management, starting with a method of asserting identity in virtualized infrastructure. We need a similarly consistent way to secure the myriad of interfaces our future networks require. This paper advocates applying the notion of security associations in a consistent way to all interfaces. Finally, we need to consider how emerging distributed architectures allow service providers to implement supply chains that turn out networks as a service. Our operational security practices need to extend into development and integration and even to the open source organizations and vendors supply both hardware and software used in our infrastructure.

Abbreviations

API	Application programming interface
CMTS	Cable modem termination systems
devops	Development operations
ETSI	European Technical Standards Institute
HSM	Hardware security module
IoT	Internet of things
MD-SAL	Model driven service adaptation layer

NFV	Network function virtualization
NFVO	Network function virtualization orchestrator
ONF	Open Network Foundation
PKI	Public key infrastructure
SDN	Software defined network
TPM	Trusted platform module
VFNM	Virtual network function manager
VIM	Virtual infrastructure manager
VM	Virtual machine
VNF	Virtual network function
VNFi	Virtual network function infrastructure

Bibliography & References

Software-Defined Networking (SDN) Definition. Open Networking Foundation. Online. Downloaded July 11, 2017. https://www.opennetworking.org/sdn-resources/sdn-definition.

OpenFlow. Open Networking Foundation. Online. Online. Downloaded July 11, 2017. https://www.opennetworking.org/sdn-resources/openflow.

[ETSI NFV 1] ETSI GS NFV 002 V1.1.1 (2013-10): Network Functions Virtualization (NFV); Architectural Framework. European Telecommunications Standards Institute. 2013. Online. Downloaded July 11, 2017. http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf.

[NGMN 5G] NGMN 5G White Paper. NGMN Alliance. 2015. Online. Downloaded July 11, 2017. https://www.ngmn.org/uploads/media/NGMN_5G_White_Paper_V1_0.pdf.

[ONF Primer] SDN Architecture – A Primer. Open Networking Foundation. Online. Downloaded July 11, 2017. https://www.opennetworking.org/images/stories/downloads/sdn-resources/7-26%20SDN%20Arch%20Glossy.pdf

[RFC 7803] RFC 7803: Network Configuration Protocol (NETCONF). Internet Engineering Task Force. 2011. Online. Downloaded July 11, 2017. https://tools.ietf.org/html/rfc6241.

[ODL] OpenDaylight Controller: MD-SAL. OpenDaylight Wiki. A Linux Foundation Project. Online. Downloaded July 11, 2017. https://wiki.opendaylight.org/view/OpenDaylight_Controller:MD-SAL.

[ONF 3] ONF TR-530. Threat Analysis for the SDN Architecture. Open Networking Foundation. Version 1.0. July 2016. Online. Downloaded July 11, 2017. https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Threat_Analysis_for_the_SDN_Architecture.pdf.

[ETSI NFV 2] ETSI GS NFV-SEC 001 V1.1.1 (2014-10): Network Functions Virtualization (NFV); NFV Security; Problem Statement. European Telecommunications Standards Institute. 2013. Online. Downloaded July 11, 2017. http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/001/01.01.01_60/gs_NFV-SEC001v010101p.pdf.

[Lee] Resource Management in Service Chaining. IETF Internet-Draft. draft-irtf-nfvrg-resource-management-service-chain-0. July 2015. Online. Downloaded July 11, 2017. https://tools.ietf.org/id/draft-irtf-nfvrg-resource-management-service-chain-01.html.

