# NFV Security: Emerging Technologies and Standards

# 2

Igor Faynberg and Steve Goeringer

## 2.1 Introduction

This chapter addresses the NFV security while reflecting on the work of the ETSI NFV Security Working Group (NFV SEC WG), which has indeed been the key forum in the industry work on the subject since 2011, gathering network operators, major vendors, governments' representatives (primarily regulators and law enforcement agencies), and researchers. Hence the consensus reached in the group *is* the industry view.

The authors feel it is important to communicate this view. At the same time, the authors also state, where appropriate, their opinions and present a vision on how certain technologies have to develop in the future, and when these opinions are stated, it is made clear that these are the opinions rather than standards or established views of the industry.

A few words on the history of the NFV SEC WG. This WG was championed by Don Clarke (then the Head of Research in BT), the man who had spearheaded the NFV Industry Specification Group creation in the first place. The security group started and functioned for the first 2 years as an *expert group*, whose charter was merely exploratory. The main task of the group was to outline the security problems that were *specific* to the NFV (as opposed to generic cloud, whose security problems had been already tackled by a number of organizations—the US National Institute of Standards [NIST] and Cloud Security Alliance [CSA] among them) so as to avoid duplication of effort and thus develop sharp focus on what is specific to the

I. Faynberg (✉) • S. Goeringer
Cable Labs, Louisville, CO 80027, USA
e-mail: i.faynberg@cablelabs.com; s.goeringer@cablelabs.com



S.Y. Zhu et al. (eds.), *Guide to Security in SDN and NFV*, Computer Communications and Networks, DOI 10.1007/978-3-319-64653-4_2

telecommunications industry.[1] In 2015, the NFV SEC WG has become a working group with the charter to develop industry standards.

As it turned out, a set of problems identified in the NFV Security Problem Statement [2] has been comprehensive in that all but one NFV SEC WG's work item has been accepted by the SEC WG to address problems identified in that set. One exception was the study of the OpenStack security [3], which was carried to document the state of the art in the open-source development. (Overall, by the nature of its work, the SEC WG stayed away from abstract models, concentrating instead on specific use cases, available technologies, and operators' requirements.)

The rest of this chapter is as follows:

- Section 2 outlines the main differences between the "generic" cloud and NFV and discusses the security threats as well as new benefits for security provided in the NFV environment;
- Section 3 discusses the problems in the NFV Security Problem Statement and explains how the NFV SEC work items map into these problems;
- Section 4 explains how *trust* is bootstrapped from hardware and established among the execution components, the discussion culminating in the treatment of the subject of remote attestation;
- Section 5 introduces the requirements and architecture for lawful interception in the NFV environment and reports on the results of the NFV Security WG work on the architecture and security controls for sensitive component execution;
- Section 6 is dedicated to security management and monitoring;
- Section 7 introduces the NFV Security WG work on the analysis of the *OpenStack* security;
- Section 8 is the conclusion;
- Acknowledgments; and
- List of references.

## 2.2 Threats and Opportunities

The first question that needs an answer here is what is specific to the NFV environment in comparison to that provided by the generic cloud computing (as, for example, described in a recent monograph [4]).

In short, the NFV is a "Telco cloud" established for and used by network operators, and here lies the answer: While the generic cloud provides computing services, the NFV is about providing telecommunications services. Of course, with the convergence, the differences between the two types of services are blurring, but there are some essential characteristics that make the telecommunications services distinct.

[1] For the history of the initial development of the NFV Security Working Group, see [1].

For one thing, network operators are regulated. This places special requirements on reliability. Even more stringent (and, as we will see, much more challenging to implement in the virtualized environment) are the requirements related to lawful interception. Examples of other stringent regulatory requirements—which differ from country to country—are those related to data retention, personally-identifiable information sharing, and movement of data that are considered private across national or regional borders.

From here, we can see that networking plays as defining role in the NFV as virtualization does. To this end, the development of the NFV is coupled with that of the software-defined network (SDN) technology standardized by the Open Networking Foundation (ONF).

SDN and NFV are independent (but related) technologies that network operators are using to create open distributed network architectures. The transition to this new model of networking is well underway, expanding and extending on lessons learned in data centers.

One limitation of the regulated environment is that network operators have much less control of interconnection options. The consequence is that the open distributed architectures they develop must support multiple-operator interconnectivity, while the solutions must support multiple tenants (often providing infrastructure to other operators as clients). These networks might span continents. The result is a nebulous network with soft perimeters. Providing a comprehensive layered security solution in this environment is quite challenging.

As far as security is concerned, opening up new services is often at cross-purpose with the objective of limiting the threat surface. The nature of network services is such that once a capability that has value is developed, that value is only achieved by opening up access to it. Opening up to address a market or company need inevitably means accepting a risk that another party may exploit this new capability in some way. Introducing security controls necessarily limits how a service or device might be used, which also necessarily decreases its value. In other words, the value of a network is inversely proportional to how secure it is!

Hence security engineers are always seeking balance between addressing a target market with compelling capability and limiting the use of this capability sufficiently so that only reasonable risks remain. The goals, therefore, are to make exploitation expensive (not to eliminate threats altogether) and employ evolvable or upgradeable security controls and methods.

SDN and NFV, in and of themselves, are only contributing technologies to how networks are evolving. Open distributed networks also integrate ideas from development operations (DevOps), software repository and distribution technologies, various virtual infrastructure implementations integrating virtual machines, hypervisors, physical and logical hosts, physical and logical interfaces, and much more. The resulting network manages complexity through abstraction. This abstraction can create security comfort through obfuscation; however, obfuscation is never a reasonable security approach. Moreover, abstraction itself presents significant security challenges. Do security operations have visibility to all the physical and logical elements that must be secured? Can security professionals see all the flows

that occur in their network and see far enough up through the layers of abstraction encoded in APIs and interfaces to have context? Is the architecture consistent enough to allow correlation of events and to chain dependencies so the security engineer can identify and isolate compromised devices?

Another complexity of SDN and NFV technologies is in the way they actually distribute network state. Ultimately the purpose of an SDN controller is to maintain network state by distributing the flow table entries across multiple network elements, enabling programmatic implementation of end-to-end connectivity. Similarly, NFV orchestrates the deployment of dynamic infrastructure, creating chains of service elements that run their own interdependent state machines to provide capabilities. Consequently, this creates a new, target rich environment for adversaries to do new types of denial of service attacks, different methods of pivoting to gain access and manipulate network behavior, and more.

Moreover, the consequence of failure can be much higher. Once an adversary gains access to a virtualized infrastructure, the adversary may have the opportunity to penetrate hundreds or thousands of other physical or virtual devices. Thus entire infrastructures are likely to be compromised, deeply and widely, and nearly simultaneously at that. The notions of security in-depth and threat management through kill-chain modeling are critical for open distributed networks.

Fundamentally, open distributed architectures introduce risks by two key factors. First, the new infrastructure transitions from a hardware-centric orientation to one focused on software, and so networks become vulnerable in ways traditionally associated with software-based solutions. Second, the decomposition of network elements that separate the data plane, control plane, and management plane dramatically increases the attack surface that adversaries can address. There are simply more interfaces and elements (physical and virtual) to exploit. Moreover, concurrent changes in other IT technologies (such as DevOps) introduce further emphasis on software as the actual infrastructure and really create a virtual supply chain for service delivery and deployment. Thus vulnerabilities in software development processes now become operational vulnerabilities in the nature of how networks are managed and maintained.

We refer a reader to the latest results in SDN development. The ONF summarizes threats to SDN in its Technical Report TR-530 [5]. It must be noted that the security mechanisms outlined by ONF are optimized for an environment specified in [6]. Developing specifications and concepts as outlined by the ONF view of the SDN evolution in TR-535 [7] introduce entirely new network control and flow management practices which remain to be assessed fully.

NFV has a similar, if not greater, impact on increasing the attack surface: NFV introduces a new model for management and orchestration with new interfaces, which an attacker may attempt to exploit. Most are implemented as software with inherent software vulnerabilities.[2]

[2]The fact that security problems are introduced by sloppy programming is well known, although it is often overlooked because it is rarely mentioned. As Dijkstra famously noted in [8], "The required techniques of effective reasoning are pretty formal, but as long as programming is done

With the newly abstracted nature of NFV elements, isolation of failures may be more difficult, affecting an often forgotten security factor—availability. Even aside from that, threat-correlating network security data so that compromises can be identified and isolated to specific physical or logical elements might be more complex.

NFV also includes the notion of service chaining—the ability for an orchestrator to provision multiple network functions in series or even in parallel to provide a composite service. This creates a level of cascading complexity which can dynamically increase the attack surface as services are dynamically and automatically created and provisioned.

A plethora of potential problems (and, as we will see soon, benefits) stems from the hypervisor administrator's capability for introspection—that is the full access to the memory of any virtual machine at run time. If the respective API falls into the wrong hands, no secrets can be kept. As a result, a virtual machine effectively escrows all cryptographic keys with the administrator as well as with any other entity that has access to the introspection API.

On the other hand, introspection is quite useful in that it allows, for example, to detect root kits and otherwise enable security monitoring services. To this end, NIST [9] encourages cloud operators to "Consider using introspection capabilities to monitor the security of activities occurring among guest O[perating] S[ystems]s." (For more information on the services that a hypervisor can provide and known attacks on the virtualization infrastructure, see [4].)

But as far as lawful interception (LI) is concerned, the hypervisor introspection presents a big problem. One critical LI requirement is that the very act of surveillance must remain undetected by the persons who don't have a need to know. The hypervisor administrator (a human or a software agent with the access to the hypervisor API) might not necessarily have such a need, but the administrator has full access to the infrastructure within an individual host. This is a major challenge in implementing reasonable support for LI on NFV infrastructure.

by people that don't master them, the software crisis will remain with us and will be considered an incurable disease. And you know what incurable diseases do: they invite the quacks and charlatans in, who in this case take the form of Software Engineering gurus." In the same page is a quote from an early 1984 EWD: "Machine capacities now give us room galore for making a mess of it. Opportunities unlimited for fouling things up! Developing the austere intellectual discipline of keeping things sufficiently simple is in this environment a formidable challenge, both technically and educationally." As unfortunate as it is, the "software crisis" must be a primary factor in security assessment.

The summary of the security challenges introduced by virtualization is thus as follows:

- Reliance on additional software (that is, hypervisors and modules for management and orchestration) and hence a longer chain of trust
- Reduced isolation of network functions
- Fate-sharing due to resource pooling and multi-tenancy
- Effective key escrow for hosted network functions
- Complexity of implementing LI

The good news is that there are mechanisms and tools to deal with these challenges. Furthermore, there are unique opportunities in NFV when it comes to security.

First, NFV helps streamline security operations. In a cloud environment, multi-tenancy drives the need for logical separation of virtual resources among tenants. Through orchestration, certain virtual network functions (VNF) can be deployed on separate compute nodes, and they can be further segregated by using separate networks. In addition, the use of security zones allows VNFs to be deployed on—or migrated to—hosts that satisfy security-pertinent criteria such as location and level of hardening. Centralized security management allows network functions to be configured and protected effectively according to a common policy as opposed to a collection of per-NF security procedures that may not always be consistent or up to date.

Second, NFV can ease the operational impact of deploying security updates. An upgraded instance of the VNF can be launched and tested while the previous instance remains active. Services and customers can then be migrated to the upgraded instance over a period of time (shorter or longer as dictated by operational needs). The older instance with the un-patched security flaw can be retired once this is complete.

Third, by using hypervisor introspection, root kits can be detected and, consequently, eliminated. Overall, the run-time memory analysis can improve the security posture of a VNF, a process that was very difficult on network appliances or standalone services used for legacy telecommunications infrastructure.

Fourth, NFV opens up new possibilities in incident response owing to the inherent flexibility it introduces. For example, automated incident response could include rapid and flexible reconfiguration of virtual resources. Another characteristic of network function virtualization that leads to improved incident response is the relative ease of decommissioning and recommissioning VNFs. If a VNF is suspected of having been compromised (for example, through unauthorized access via a backdoor), an uncompromised version can be instantiated to replace it, and the compromised version can be decommissioned and a copy of it made for forensic analysis.

Fifth, one well-recognized benefit of the cloud environment is that it stimulates the use of analytics. This, of course, immediately applies to security in more than one way: analyzing the running code for viruses (and possible anomalies) as well as

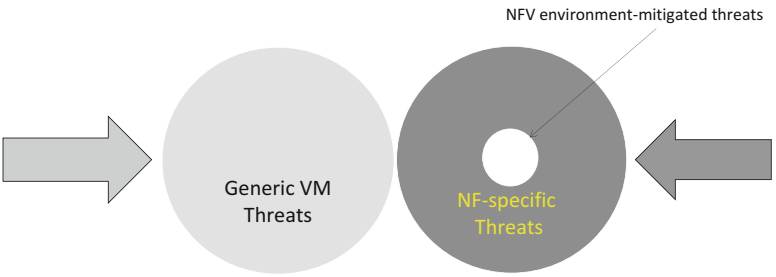


**Fig. 2.1** VNF threat classification

analyzing traffic both for early detection of distributed denial of service attacks and distribution of malware. Again, the relatively central nature of the NFV enforces systematic use of analytics to develop a "big picture" of the state of a data center and the whole operator's network.

Now, we are ready to classify the threats discussed so far and consider which of them are specific to the NFV environment. Figure 2.1 illustrates this point.

In the simplest case, a VNF is an instance of a network function running on a virtual machine (VM). The overall set of security threats to a given VNF can be, at the first approximation, viewed as a combination of all generic virtualization threats (a circle on the left) and those threats specific to the network function software (a circle on the right).

As we discussed earlier, the latter set has a subset comprising the threats that can mitigated by the new mechanisms—such as hypervisor introspection and centralized security management. For this reason, we "carve out" this subset, thus reducing the threat landscape.

Now, the Cartesian product of these sets (i.e., a set of pairs of virtualization threats and unmitigated network-function threats acting simultaneously) provides the full landscape of the NFV-specific threats. The potential problems that stem from the most pertinent threats in this space are the subject of the next section.

## 2.3 The Problems Identified in the ETSI NFV Security Problem Statement

To understand the actual risks of the threat landscape described above, it is essential to consider the deployment models envisioned in the NFV. This is exactly what [2] does.

The simplest is what [2] calls a *Monolithic Operator.* Effectively, this is an operator's private cloud. Only operator's own network functions are represented there, and thus, most security concerns that deal with hosting are absent here. (The reason we consider such deployment unlikely is that in its pure form, it excludes even hosting of content delivery servers.)

The next model is called *Operator Hosting Virtual Network Operators*. Here, the operator's cloud hosts VNFs that belong to other operators. Since, a *virtual machine escape* (i.e., a situation in which a rogue virtual machine can get control of a hypervisor) is not unheard of and also because of potential "noisy neighbor" problems, the expectation is that an operator in such deployments will isolate the VNFs of a hosted operator on a separate hardware platform. With that, each hosted operator will be provided a separate hardware platform.

More extreme is the *Hosted Network Operator* model in which "An IT services organization operates the *compute* hardware, infrastructure network, and hypervisors on which a separate network operator runs virtualized network functions. The premises including cable chambers, patch panels, etc. are physically secured by the IT services organization." In this model, of course, the security of such operator's practice depends entirely on that of the IT service organization.

The *Hosted Communications Provider* model is a hybrid of the two previous models. Here, the IT service organization hosts either more than one communication services provider (CSP) or even a more than one wholesale network operator. In the latter case, the IT service organization sells the rights to each network that provides to run VNFs for the wholesaler. The wholesaler then resells these rights to the retailers. (We can note the necessity of a well-developed identity management framework for this case.)

The *Hosted Communications and Application Providers* model takes the next step by permitting the IT service organization to offer full-blown public cloud services, while the same facilities that are used in that offer are supporting the network operators and communication service providers.

In the *Managed Network Service on Customer Premises* model, a network operator runs VNFs on its own hardware located on a customer's premises and physically secured by the customer.[3] This model can be deployed for an enterprise or even a home network.

When the hardware belongs to and is operated by the customer, the above model becomes that of *Managed Network Service on Customer Premises Equipment*. For instance, the customer may allocate a host to the network operator where all the network operator's VNFs are to run. This specific deployment model excludes sharing the host (and hypervisor) between the network operator and a customer, although the model in which this is done is valid, too.

To determine the security implications of a deployment scenario, one needs to consider all parties at the level each of them operates (e.g., host hardware, hypervisor, or guest VNF). Then a decision has to be made as to which use rights each party may have over its resources. The fundamental security engineering design factor here is to provide a basis for trust suitable for all parties. See [2] for more discussion of how this can be effected.

Next, the NFV Security Problem Statement considers the potential attackers, traditionally classified by their respective means, motives, and opportunities. The

[3]Over the years, the NFV ISG has considered a number of such use cases.

introduction of NFV does alter the means and opportunity to exploit a vulnerability. How far this can go depends on the technical and contractual position of an organization in relation to others in the supply chain of NFV. To this end, the following hierarchy is considered:

- End-customers of retail network operators
- Retail network operators
- Wholesale network operators
- Hypervisor operators
- Infrastructure (i.e., hosts, storage, and infrastructure network) operators
- Facilities managers (who are responsible for the physical security of buildings and equipment)

A hosted service implies that a party at a given level contracts with (and thus places a degree of trust in) the parties operating lower levels.

The attacks are likely to occur from either a higher level, or at the same level (as in the case where a hosted network operator might spy on is competitor sharing the same facilities), or from inside (by disgruntled or unfaithful employees).

A hosting operator might mount willingly an attack on a guest (such as stealing confidential information that can be sold) as long as the attack does not degrade performance or otherwise affect the operator's reputation. Among existing threats are those related to intellectual property (i.e., proprietary algorithms, configuration files). Reverse engineering and side-channel attacks are specifically mentioned in [2] as the ones that need to be mitigated to protect the intellectual property of vendors from (1) one another, if they are running on the same platform and (2) from the platform operator. This can be achieved with the technologies for execution of sensitive components, discussed in Sect. 6. Of course, the full protection here is limited as it is infeasible for all of a guest's computing functions to be concealed from the host. One alternative technology applicable here, noted in [2] is homomorphic cryptography, which is becoming practical for certain very specific functions without too much overhead.

The rest of this section describes specific problems that the ETSI NFV Security Group has identified in [2]. These problems are:

1. Topology validation and enforcement
2. Availability of management support infrastructure
3. Secured Boot[4]

[4]This term has been subsequently changed to "Trustworthy Boot," defined in [11] as the means to encompass "the technologies and methods for validation and assurance of boot integrity." This subject will be addressed in the next section, but, in a nutshell, the same result can be accomplished with different alternative technologies based on different standards. Since the NFV Security Problem Statement has not changed, we keep the old term throughout this section. (As pedantic as it may sound, the term "Secured Boot" was created for a similar reason: to refer to a generic set of mechanisms vs. the UEFI *Secure Boot.*)

4. Secure crash
5. Performance isolation
6. User/Tenant Authentication, Authorization, and Accounting
7. Authenticated Time Service
8. Private Keys within Cloned Images
9. Backdoors via Virtualized Test and Monitoring Functions
10. Multi-Administrator Isolation

### 2.3.1 Topology Validation and Enforcement

An essential requirement for a network provider is that customers' networks and the provider's own network are partitioned so as to be isolated from one another.[5] This creates separate trust domains. In any pre-NFV environment, this is effected by a set of firewalls (containing network address translators), which are properly provisioned by the operations and management systems according to a provider's policy.

Virtualization changes the demarcation between customers' and providers' trust domains. As Fig. 2.2 illustrates, a virtualized forwarding function may interconnect partitioned networks even in the simplest (i.e., virtual LAN-based) cloud environment. Overall, while the inter-host paths can be controlled in the pre-NFV ways, the intra-host paths fall under control of virtualized forwarding functions and, ultimately, hypervisors. Thus, this is a classic example of the generic case mentioned earlier in which the environment is exposed to a pair of a threats, one inherent to physical networking and another threat introduced by virtualization.

Therefore a network operator needs to be able to ensure that the connectivity of the whole network meets the security policy. Furthermore, it is necessary to prevent the establishment of an unauthorized connection.

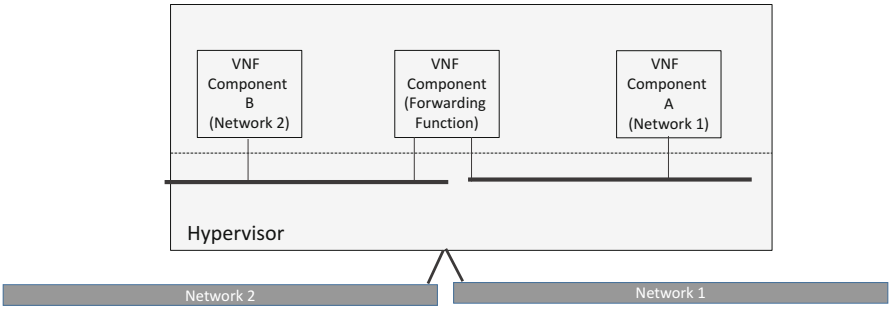


**Fig. 2.2** Interconnection of partitioned network by a virtualized forwarding function

[5] A storage network also needs to be isolated, as does the operations-and-management network. This issue will be addressed later in this section.

Various examples of establishing and validating service chains are presented in [2], which also recommends approaching the problem at different connection levels: the physical cabling, ports of each forwarding function, internal configuration of the forwarding function (as it relates to the assigned place in the service chain, etc.).

This is a challenging problem, which is further complicated by the possibility of introducing loops into service chains, which can be exploited to amplify a denial-of-service attack traffic. Potential mitigation steps here involve loop detection during the topology validation stage.

As may be expected when discussing networking, the subject of SDN is brought up in [2] because "SDN is considered highly complementary to NFV in certain scenarios (e.g., data centers)."[6] SDN connectivity can be defined programmatically, resulting in dynamic and flexible network configurations. Consequently, validating and constraining topologies is more difficult than in the "traditional" case. To address this complexity, [2] suggests an approach in which a network is partitioned into *security zones*, each zone defined by a distinct set of security policies. It is important to list here several reasons for such partitioning, as these reasons are specific to the managed networks and therefore are defining as far as NFV is concerned. These reasons include legislative or jurisdictional control, customer-type (e.g., government, enterprise, or residential), transferred content (as it may require rights protection or confidentiality), and multi-tenant controls, where network functions of competing network operators are hosted.

The SDN topologies are likely to be hierarchical—arranged in several layers. The simplest case is presented by one-layer, single-controller network, in which the controller will push rule sets into the switches. In more complex schemes, lower-level components request routing decisions from higher-layer components. In all cases, there is a need for a mutual authentication for every pair of interlocutors to prevent injection of malicious commands (by an entity masquerading an upper-layer component) or, divulging the network topology (by switches). Different layers may be implemented by different network operators, so trust management and network partitioning again become critical design considerations.

Network performance and security are often at cross-purposes when one tries to find the right place in the hierarchy for making forwarding decisions. Consider two extreme polar cases. Making forwarding decisions for every packet by the lowest controller in the hierarchy may provide the best security in terms of correlating and isolating distributed denial of service (DDoS) attack traffic. However, this may be impractical because of scaling and performance concerns. In contrast, making all the forwarding decisions in a switch may provide excellent performance but may also result in never detecting problems that would have been obvious had there been a possibility of correlating the traffic visible only to the higher-level entities.

[6]It is noted, however, that the SDN is still an emerging technology, in which (as of 2014) the full set of controls has not been standardized.

One possible solution here is for a switch to monitor the traffic and then send periodic updates to the SDN controller hierarchy. This has to be designed carefully to avoid making a controller a possible target of a denial-of-service attack, which would likely destabilize the whole network.

Another complication standing in the way of consistent topology validation process is that operators[7] can program the behavior of the switches via the operations and management interfaces independently of the controller. To avoid inconsistency, a capability to report any such change to controller must be built into the protocol. (Note that the NFV Security Problem Statement mentions *attestation* (discussed further in this chapter) as the mitigation means to ensure that the configurations and other essential operational data have not been changed since the last time they were modified legally.)

A more complex feature interaction problem may occur because the virtualized forwarding function may change the routing of packets in application-specific ways. With that some functions may take their instructions from the SDN control hierarchy (via *OpenFlow*™ interfaces), but, as [2] explains, it is one of the purposes of NFV "to enable deployment of application-specific Forwarding Functions, that will not, in general, be amenable to description by a deliberately constrained protocol such as *OpenFlow*."

Having touched on the SDN-related matters, we refer a reader to [2] for the discussion of the much better understood topology validation issues specific to the use of the traditional distributed routing protocols.

Finally, [2] stresses the necessity of keeping the overall "out-of-band" management system always alive. This can be helped by ensuring that the management ports of processing blades, switches, and storage controllers have both the physically independent connectivity to the management and orchestration system and locally accessible caching mechanisms for storing configuration state and logging events.

### 2.3.2 Availability of Management Support Infrastructure

The single most important requirement here is that the management infrastructure be available even when the infrastructure that it manages is out. In a way, that requirement has been spelled out already when we discussed the SDN. To quote [2]: "Ideally the management ports of processing blades, switches and storage controllers ought to have physically independent connectivity to their configuration state in the management and orchestration system, as well as locally accessible storage/caching for configuration state and the necessary access controls to these rudimentary but critical resources." The goal, or necessary practice here, is to make the operations network inaccessible from customers' networks.

[7] And thus an inside attacker.

The problem with fulfilling this requirement is the costs associated with providing a separate (physical) network for operations and management. It is quite possible to do so in a data center (and *OpenStack* supports that as demonstrated in [4]), but for an operator's network that spans multiple data centers spread over a sizable geographic area, the solution has been to dedicate a virtual private network for these purposes.

There are several aspects to this arrangement. First, the management network must be robust. To ensure availability, [2] recommends path diversity (including cellular network backup) whenever it is economically feasible. Second, access control to the management network also needs to be more stringent than access to the supported networks. An example of a specific challenge introduced by the NFV is booting of a hypervisor. This procedure may require network access to obtain its own configuration, software licenses, cryptographic keys, etc. For that, a hypervisor does need a "purely physical" access to the management network, which must be physically isolated from others. A similar problem arises on the start-up of a virtualized forwarding function. It may rely on accessing the network through another forwarding function in the chain only as the latter does not rely (circularly) on the very function that is being started up. One solution proposed by [2] is never to allow the management network to use a virtualized function on any forwarding path. Perhaps over time, the industry could develop a provable recursive solution though.

### 2.3.3 Secured Boot

Here we address the fundamental problem of establishing the chain of *trust*, on which we further expand in the next section. In a nutshell, an application's users trust both the application software and the operating system on which the software runs. An operating system, in turn, trusts the hardware on which it executes. In the cloud, a hypervisor is largely replacing the hardware as a trusted entity (by operating systems), but a hypervisor still has to trust the hardware. Finally, the cloud operator must have a basis to trust the hypervisor, the hardware, and the various software installed on it.

Overall in the NFV environment, a hosted network operator has to trust its hosting provider's virtualization platform sufficiently to run virtual network functions on it; conversely, the NFV operator must trust each VNF (which means being able to ascertain that each VNF comes intact from an accepted vendor, performs to its specifications, and is not being modified in the process). Each VNF can, in turn, be composed of multiple workloads (VMs in traditional virtualized infrastructure), and so trust chaining can become quite complex.

To bootstrap the process of building trust, we provide mechanisms and processes to base trust in the hardware (in that it has no malicious modules and otherwise acts according to its specifications). The next step is to ascertain that the booted software belongs to the trusted vendor. This is precisely the problem addressed here.

Secured boot encompasses the technologies and methods for validation and assurance of *boot integrity* validation. The secured boot process actually can do a bit more than just checking software—in addition to checking the hypervisor and OS image, it can also validate add-on hardware modules (such as acceleration hardware) and firmware.

Furthermore, the established trust base is further used to ensure that the software loaded into the VNF execution environment is authentic and has not been tampered with. This is achieved by checking cryptographic signatures of the respective modules. (Unless specifically stated otherwise, we always assume asymmetric cryptography.)

In the NFV environment, there is a need to incorporate software from multiple software vendors. As [2] notes, "to minimize certificate management complexity in such cases it may be desirable to have a single certification authority for VNFs."[8]

The relevant secured boot technology (often under different names—such as "secure boot" or "trusted boot"—and in somewhat different contexts) has been addressed in various fora. For example, the architecture and mechanisms for verifying signed firmware and software images are specified by the *Unified Extensible Firmware Interface* (UEFI) Forum (www.uefi.org). UEFI enables one to ascertain that host is booted into a known configuration based on hardware-rooted trust. Although supported on servers, the technology is not yet in use widely.

The *UEFI secure boot* involves checking the signatures of all UEFI modules as they are being loaded. If the signature check fails, the boot stops. This process leverages a public key infrastructure, in which the public keys of vendors are stored in a database, augmented with the revocation list. There is also an option for an administrator to approve a boot signature manually at the console.

Another, more general technology to achieve this—and wider—purpose is called *trusted computing* and standardized by the Trusted Computing Group (TCG) (www.trustedcomputinggroup.org). The new and essential implement here is the *Trusted Platform Module* (*TPM*), a tamper-resistant hardware "box,"[9] which stores the private endorsement key and also performs a variety of computing operations. Neither the host CPU nor, for that matter, any other hardware module may look inside the TPM arbitrarily. The TPM communicates with the outside world via a well-defined interface. The ultimate goal is to establish the *chain of trust* that encompasses all pieces of firmware and software.

Recognizing that the TPM technology can be implemented using various hardware standards, the ETSI NFV Security Group came up with a general term, *hardware-based root of trust (HBRT)* (defined in [17]), to refer to the anchoring function presented in a hardware-based TPM. There have been claims that a similar

[8]Indeed, there has been a long-standing work item in the NFV Security Working Group on this subject.

[9]Implemented as a dedicated ASIC or a subcomponent of another processor. The chip would provide external mechanisms to prevent or make difficult tampering or inspection and also provide mechanisms to destroy stored secrets if tampering is detected.

function can be developed by a *hardware security module (HSM)* (see [4] for a review of the HSM technology).

The trust chain is maintained through the execution of secure transactions, which (1) isolate memory (for example, for storing derived keys), (2) bind storage to specific configurations of hardware and software, and (3) provide *remote attestation* (alarming a specified party to all environment changes).

Among other functions that TPM provides are those to generate the cryptographic keys (bound to the endorsement key) and to store the *measurement* of the respective boot components. TPM has been implemented in hardware, but there has been an effort to virtualize it [10].

Given the definitions provided above, a clarification of terminology is necessary. Booting with TPM is called *trusted boot* to differentiate it from the UEFI *secure boot.*[10]

The outstanding question that [2] poses is whether these technologies have proven to be feasible to operate at network operator scale. We return to this in the next section.

### 2.3.4 Secure Crash

It is a common place that programs must not crash. About any crash leaves the program memory and other resources in an unknown state, and this results, among other problems, in a significant potential vulnerability. With that, a crash of an application is different in its consequence from a crash of an operating system because the latter naturally exposes the resources of all its applications. Ultimately, a crash of the host's hypervisor exposes the resources of all virtual machines; however, [2] concludes that the "Cloud technology already has a strong track-record of robust design against crash-related vulnerabilities. Therefore it would seem that NFV adds no new concerns here." The NFV-specific problem is that NFV magnifies the consequence of a successful attack.

Within the NFV framework, the key components that are at risk in this context are the hypervisors and virtual network function component instances. In the latter case, the role of the hypervisor is to ensure that all file references, hardware pass-through devices, and memory are safe from being accessed by unauthorized entities.

But not all problems are confined to the host itself. An example of a "remote" problem is the references to a crashed virtual network function component stored on remote devices. As [2] notes, the devices often use such references as the means to "authenticate" a machine. There is a need to purge those references from the devices, but this can, of course, be achieved only when they are known. An easier objective to achieve is to ensure that it is not possible for a newly executing VNF component instance to adopt identifiers (e.g., MAC or IP addresses) that were recently used by a crashed instance, lest this instance impersonate the crashed one.

[10] As we will see later, the industry has introduced a new, generic term, "trustworthy boot."

A similar problem is related to storage (both local and remote) resources attached to the crashed virtual network function component instances. Since the hypervisor cannot know what storage resources need to be wiped in the event of a crash, it is likely to be the job of the VNF manager to wipe them.

Naturally a crash of a virtual network function component affects the availability of a service. In this case, [2] suggests that the VNF manager needs to identify the likely cause of the problem and work with the NFV infrastructure (via the virtualization infrastructure manager) to work around it. The remedy may be the creation of a new component instance (or set of instances), rerouting of packets passing through the crashed component, or the creation of new routes among the dependent entities. This places requirements on the virtual network function descriptor to store the information to be used by the VNF manager in the case of crash.

### 2.3.5 Performance Isolation

The generic problem here is that a virtual machine may (more often than not because of one or another software fault in a hypervisor) affect performance of other virtual machines on the same host. In an extreme case, a machine can "escape," that is take control of the hypervisor thus control all other virtual machines. Even when done passively, this amounts to learning all cryptographic secrets of other machines and unlimited monitoring of all communications. In milder cases, without "escaping," a misbehaving machine may consume more resources (such as memory, CPU cycles, or bandwidth) than it is supposed to do, thus degrading the performance of other machines. We refer a reader to [4], which describes this problem—and some ways of dealing with it—at length.

As with the previous problem, the consequences of isolation failure in the NFV environment may be catastrophic (especially in view of the lawful interception requirements). Hence [2] is considering a range of isolation approaches, of which the most effective is static hardware segregation (hard partitioning of resources such that memory and storage at not shared at run time).

Others include ensuring proper configuration of the hypervisor so as to constrain the ability of a VNF component to acquire memory, processing cores, CPU quanta, and so on. These techniques, however, may prove inefficient when the granularity at which the resource can be allocated is too coarse or when it is impossible to predict the correct usage of resources by a given VNF. Moreover, some of these techniques significantly reduce the potential cost benefits of NFV that drives operator investment in virtualization. In fact, [2] warns that "network and I/O partitioning of … [one guest] is hard to isolate from that of other guests, because it can range widely over different distributed network resources and it can be highly variable at any point, making any partitioning very inefficient."

Another factor that stands in the way of performance isolation is the recurrent need to optimize the performance of a hypervisor. To increase I/O throughput, for example, hypervisors may allow direct pass-through, thus allowing guests access to

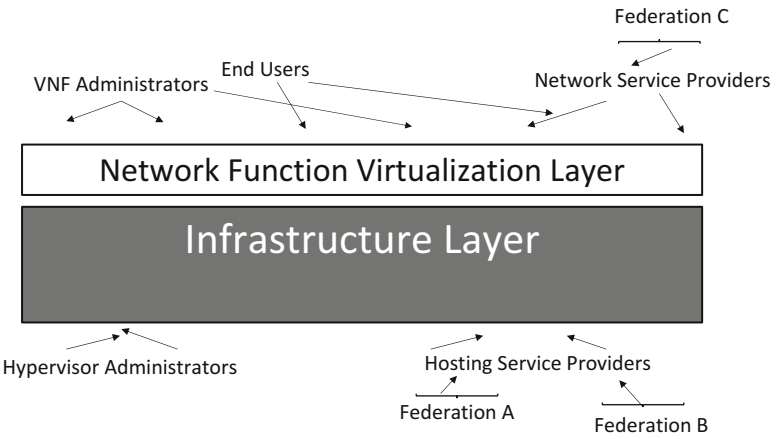


**Fig. 2.3** Identities in NFV

the common physical memory. To counter this, [2] recommends using I/O memory management units.

And yet another group of attacks are those on the resources of the virtualization infrastructure. As [2] notes, "Even when isolation is in place, whether for storage I/O, network, memory or CPU, there is a class of attacks on the resources used by the hypervisor platform itself, which may vary in ease of execution and efficacy depending on the failure modes of the underlying hypervisor and the hardware architecture."

One essential security capability recommended by [2] is proactive monitoring, which can enable mitigation. Monitoring is prescribed at two levels: the infrastructure level and, independently, at each VNF. Detecting anomalous traffic behavior, degraded performance, unusual spikes in I/O processing, and other irregularities and then leveraging the management and orchestration system to bring these data into a central place where they can be analyzed so as to find a proper response is an essential technique recommended for the NFV.

### 2.3.6 User/Tenant Authentication, Authorization, and Accounting (AAA)

In the NFV, the identities of various actors are used in (at least) two layers: at the network and virtualization infrastructure and at the network function layer, as depicted in Fig. 2.3.

Federations of actors result in compound identities, and so identity sets develop both horizontally and vertically. What the figure does not show, but what has been implied, is the law enforcement actors who may have access to some identities but not to others (and whose very presence must remain a secret from most actors). This

point in its more general form is reflected by [2] thus: "Authentication procedures can imply privacy breaches associated to the disclosure of user information at layers that are not intended to consume certain identity attributes." Consequently, addressing privacy issues in authentication needs to be validated in this multilayered environment.

Similarly, the accounting in the NFV environment may also impact privacy and so must be taken into account. For example, traffic packet acquisition and classification as well as the policy enforcement blocks on a per-actor basis should be kept private between customers, and operator use on such information may be regulated.

### 2.3.7 Authenticated Time Service

The correct function of many cryptographic protocols depends on knowing the correct time of the day, which is, for example, used in timestamps or to check certificate expiration. Tampering with time is an attack that can interfere cryptographic and security protocols as transport-level security (TLS), Kerberos, DNS security (DNSSEC), and time-limited access controls. Moreover, time accurate event logging and reporting time can be critical for performance and fault isolation procedures and event identification and management for identifying security compromise. Beyond just interfering with the security protocols, tempering with time poses a plethora of additional security problems—especially in network functions—because operations on various communications caches (such as that used in DNS) depend on correct time as does operation of routing protocols such as Border Gateway Protocol (BGP).

While there are authenticated time servers that render a variety of man-in-the-middle attacks on the Network Time Protocol (NTP) difficult, in the virtualized environment, the hypervisor is a trusted man-in-the-middle, and so a compromised hypervisor can easily tamper with timing queries.

### 2.3.8 Private Keys within Cloned Images

The potential problem is that images from which VNFs are booted may contain private keys or other sensitive data. The recommendation in [2] is that such keys have to be supplied at boot time.

The use of Trusted Platform Modules or hardware security modules can reduce the need for key provisioning, and the work in the NFV Security Group on the architecture for sensitive component execution has addressed this.

### 2.3.9 Backdoors via Virtualized Test and Monitoring Functions

This problem deals with the current dubious practice of certain vendors in which they develop "hidden" (unofficial) interfaces for run-time access to their code for

debugging purposes. As such, the problem is not exactly NFV-specific except for the hope expressed in [2] that virtualization technology could be used to create a more structured approach for authorizing whether testing and monitoring can be conducted, which diagnostic functions are allowed, and who is allowed to run them. A good practice would be to require all test and monitoring functions to be cryptographically authenticated just as for any management access to infrastructure or virtual components.

### 2.3.10 Multi-administrator Isolation

The defining use case here was dictated by needs of lawful interception as communicated by the members of the ETSI Technical Committee on Lawful Interception (TC LI). The problem here (already mentioned in the discussion of multilayered administration environment of the NFV) is that administrators of the virtualization infrastructure naturally have higher privileges than those of administrators of the virtualized functions executing on the system. For instance, a host administrator already has access to all virtual machines on the host through introspection capabilities, while an administrator of an orchestrator has access to all infrastructure controlled by the orchestrator.

This gets in the way of lawful interception—inasmuch as it occurs at the virtualization layer—because the infrastructure administrators do not necessarily have the need to know even that the lawful interception occurs, let alone be able to learn every detail of it.

In fact, the problem here is more general than that of lawful interception. Hosted operator environments are just as vulnerable to potential confidentiality violations—and exactly for the same reason. Hence the work undertaken by the NFV Security Group has been centered on solving the larger, more general problem. In effect, this solution must eventually be an evolution of role-based access control which assures administrators are able to see and do only the activities and data they should.

In conclusion, Table 2.1 demonstrates how certain work items undertaken in the group relate to the above problem set. (It should be noted that not all work items were driven by the problem statement. Some work items, such as *Report on Security Aspects and Regulatory Concerns* or *Report on Retained Data problem statement and requirements*, are of more general nature, while others—notably *Security Specification for MANO Components and Reference points*—are specific to the detail of the NFV architecture.)

## 2.4 Establishing and Maintaining Trust

Before we start with the formal approach, let us consider an intuitive one. We can envision the "bootstrapping" of security of three planes of the NFV as depicted in Fig. 2.4. We start at the lowest plane—the physical infrastructure. Assuming that we can trust the hardware, we can use it to boot all hypervisors securely, using the

**Table 2.1** Relation of certain ETSI NFV Security WG work items to the problems in the security problem statement

| | Topology validation & enforcement | Availability of management support infrastructure | Secured boot | Performance isolation | User/tenant AAA | Private keys within cloned images | Back-doors via virtualized test & monitoring functions | Multi-administrator isolation |
|---|---|---|---|---|---|---|---|---|
| Cataloguing security features in management software | | * | | | * | * | | * |
| Report on lawful interception implications | | | | * | * | | | * |
| Report on certificate management | | | | | * | * | | |
| Report on attestation technologies and practices for secure deployments | * | | * | | | | | * |
| Report on use cases and technical approaches for multi-layer host administration | | | * | * | | | | * |
| Security report on NFV LI architecture | * | | * | * | | | | * |
| System architecture specification for execution of sensitive NFV components | | | * | * | | | | * |
| Security management and monitoring specification | * | | | | | | * | * |

* The asterisk indicates that the problem in the given column is addressed in the specification in the given row.

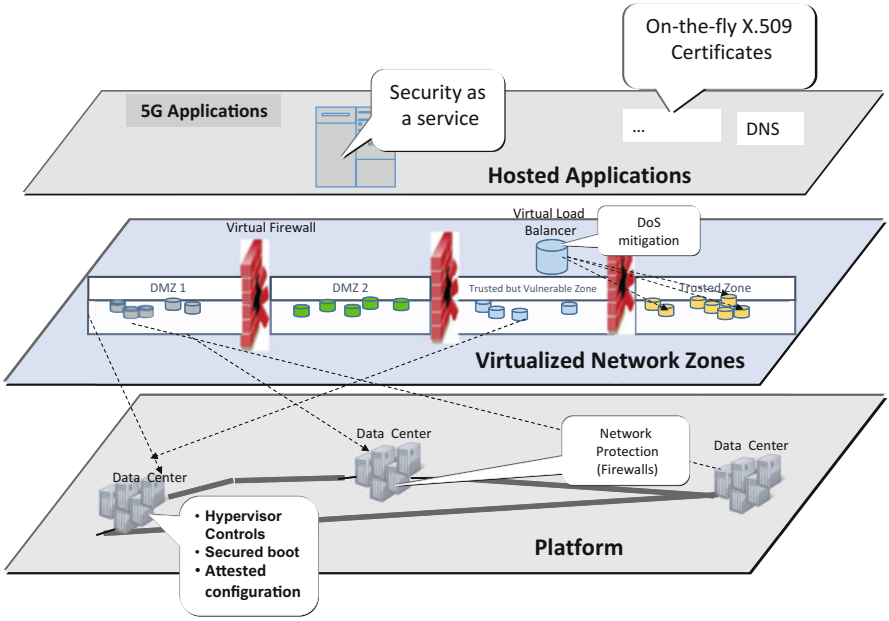


**Fig. 2.4** Bootstrapping trust

HBRT. Once booted, we can maintain the same level of security by applying all software patches and otherwise following the best industry practices for security hardening.

At the same time, we have to ensure that the physical network inside the data centers is secure and that access to it is adequately protected. The word "adequately" implies adherence to the operator's security policy.

As the hypervisors start building their own local area networks, we must ensure that the respective configurations adhere to the appropriate security policies and, once deployed, remain unchanged (except for controlled changes sanctioned and performed by the operator). This can be achieved by employing remote attestation.

At this point, we can extend the trust chain to the next plane, in which we place virtual network appliances—firewalls, SDN controllers, load balancers, and so on—and develop trust zoning in the virtualized environment. This also includes hosted environments, and so various networks can coexist now founded on the trust within the platform.

Subsequently, this chain will extend toward the applications (such as fifth-generation mobile applications) of the upper plane. Incidentally, the security services deployed at the upper plane (for example, identity management services) can be used now further to strengthen the security of the physical plane—this recursive nature of developing and chaining trust should be fully leveraged.

Having developed the intuitive view, we can take a look at the standards work in this area. The first document [11] gives a high level but systematic review of

establishing trust and the security controls in the life cycle management of the virtual network function component instantiation (i.e., a virtual machine that implements a part of a network function).

In fact, the very start of the life cycle—the creation of such a machine—can take different forms; a machine can be instantiated from a pre-built image or from a cloned image of another machine (in which case it may carry into its new a life the old baggage of security problems). Consequently, virtual asset tracking and audit records as well the networking-related data, security credentials, and software licensing information—just to list a few examples—need to be verified and, in some cases, updated.

Similarly, removal of a machine follows the same steps, but here, additional actions may be required for secure wipe and verified destruction of data.[11] Furthermore, removal has to be verified across backed-up images and cloned images. As the private keys are destroyed, so should the respective certificates be revoked. Needless to say, all these steps must be properly logged.

Of course, it is not only the "beginning of life" and "death" processes that require such scrutiny; the lifetime maintenance is actually much more involved with ensuring consistent (across hosts and data centers) patching and configuration changes. An implementation of an ingenious virtualization feature—live migration—must address memory reuse, feature parity, configuration compatibility, and service availability.

But what is *trust* after all? It is defined in [11] as "confidence in the integrity of an entity for reliance on that entity to fulfil specific responsibilities." Typically, trust is expressed through an *assurance level* based on specific measures, but it may be expressed merely through a relation (as is in *A* trusts *B* more than *C*).

With that, trust is temporary. (For instance, once booted, a hypervisor may be trusted for no longer than it is running; the trust has to be reestablished at the next boot.) The other constraining characteristic of trust is the context. *A* may trust *B* to know a parameter's value but not to change it. The trust may also be delegated.[12]

Some examples of parameters for measuring trust in NFV presented in [11] are software integrity, geographical location, hardware capabilities, and time elapsed since last audit.

Among well-known examples of trust relation is that established by a party with a certification authority (CA) in public key infrastructure. From that, a *chain of trust* is formed to the entities that are issued certificates by this CA. Specific to NFV, as we saw in the Security Problem Statement, is a matter of provisioning and storing the private keys. The techniques mentioned in [11] include the *injection* of

[11] An important point to remember is that certain data may need to be retained, for regulatory reasons (such as lawful interception). For detail, see [12].

[12] Trust delegation is typically established for the purposes of authorization. An example: when a person wants to use a printing service to print photos available on a social site, this person delegates the authority to do so to the printing service. We will see a detailed example when reviewing the *OpenStack* security below.

the private key by a hypervisor as well as the reliance on the HBRT.[13] In view of the "private key in images" problem discussed earlier, the hypervisor injection is a solution to it. Incidentally, the hypervisor trust is implicit in virtualization, and therefore, validating the hypervisor is the first and most essential step in the grand scheme of the NFV trust establishment.

Developing of a trust chain starts with the *trustworthy boot*, which, according to [11], "encompasses the technologies and methods for validation and assurance of boot integrity."

This term was defined to differentiate from earlier industry terms: the *secure boot* and *measured boot*. With secure boot, the integrity checks are based on the known hardware-based *roots of trust*. The booting process stops when integrity check fails. With measured boot, the integrity state is merely recorded without affecting the boot process. This state is checked by a *verifier* after the boot is complete, and it is up to the verifier to validate it and assign the appropriate level of trust.

The trustworthy boot process can use any of the existing boot types either alone or in combination. One overarching requirement here is that the virtual network function manager is assured that the boot process of the VNF component instance has completed.

The interpretation of the results of the process is not simply "black or white," as in the case of secure boot. Booting can still be allowed, but with reduced privileges and restricted access to certain hardware. Handling of failed integrity checks is subject to respective policies.

As a hypervisor is aided by the on-the-chip TPM, which cannot be directly used by the operating systems of the virtual machines, virtual TPMs can be created—under control of the hypervisor. There is a certain amount of controversy in the industry whether a virtual TPM can be trusted, and [11] neither prescribes nor proscribes its use.

Now we can delve into what should constitute the trust measurements in the NFV environment and how the remote attestation of this environment is performed. Here we report on the research results rather than a standard as, at the moment of this writing, the respective work in the NFV SEC Working Group is still in a progress, and so the resulting specification [13] is still in its draft form.

Attestation is formally defined in [13] as "the process through which a remote challenger can retrieve verifiable information regarding a platform's integrity state [TCG PCSISCB]." The term *remote attestation* is often used in the industry to point out that the attestation process is to be observed by a geographically remote party (a *challenger*)—not only by someone at the console of a host. Thus, even though there is no suggested central use of attestation at the moment, the notion lends itself naturally to an operations and management environment in which the whole of a provider infrastructure can be measured and attested to.

The platform's integrity information is delivered to a challenger in the form of a measurements log. One immediate difficulty here is that such a measurement log is generated by the software that is being measured. Since the challenger is trying

[13]The detail of this is not elaborated on and "left for further study."

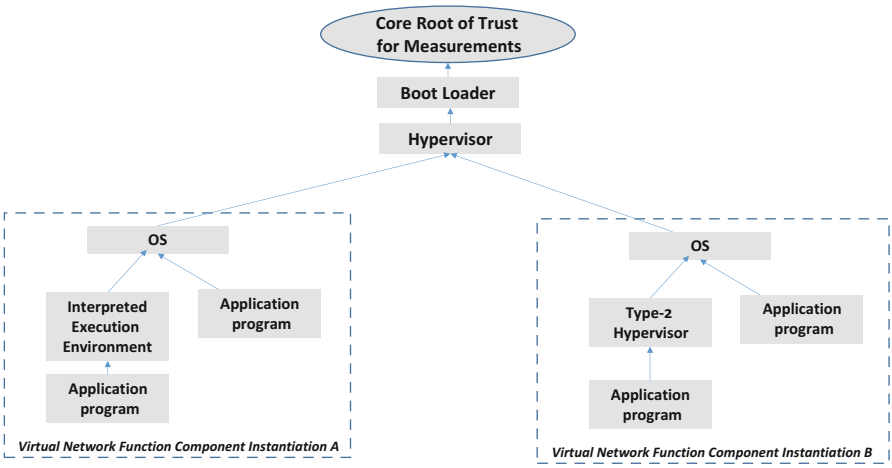


**Fig. 2.5** Establishing chains of trust

to ascertain whether this software can be trusted, it is follows that it is necessary to establish a chain of trust first and then maintain the evidence that the measurement log has not been tampered with.

The chain of trust is developed recursively, as shown in Fig. 2.5.

The process starts with establishing the *Root of Trust for Measurement (RTM)*. The boot loader is measured using the *Core Root of Trust for Measurement (CRTM)* whom everyone and everything trusts. When the boot loader is executed (after having been measured and approved for execution), it inherits from the CRTM transitive trust and thus becomes the first node in a trust tree. Every path in this tree—traversed from a leaf to the root—forms a trust chain. At this point a hypervisor or an operating system (in the case of non-virtualized environments where containers are run instead of virtual machines) is similarly measured and then booted, joining the trust chain. Similarly, once the operating system runs, an interpreted execution environment (such as *Java execution* environment) or a type-2 hypervisor can in turn be measured and approved for becoming a link in the trust chain, thus being able to measure the application, which could in turn measure its software components.

As we have already mentioned, the hardware and network configuration of the platform must be measured and verified to provide the holistic view of the platform security.

At the moment, the industry proposes six *levels of assurance* (LOAs) for the NFV [13]. In the first five LOAs, each subsequent level contains all the checks performed for the previous levels and then the additional ones that go deeper in checking the corresponding link in the chain. The sixth level checks the infrastructure network.

In relation to the last point, [13] provides an example of how a TPM can be used to verify SDN and otherwise extend the network function's attestation features to report on the current SDN configuration.

To effect that, the SDN verifier retrieves from the SDN controller the configuration of the attested network element, measures it, and then compares it with the attestation result.

The last example is a special type of *run-time attestation*, which is the attestation performed on a running program. It is fairly easy because what is measured here is a specific data segment of a program, which is not supposed to be modified. With the general programs, the problem is much harder and remains a topic of active research. (See [14] for the problem description, bibliography, and a description of a prototype implementing a partial solution.)

## 2.5 Lawful Interception and the Environment for the Execution of Sensitive Components

Lawful interception (LI) concerns two aspects of communications: the *intercept-related information* (IRI) (which can be anything but the actual content—that is signaling, call information, log record information, etc.) and the actual *content of communication* (CC) in the form of streaming traffic.

The related data are acquired through the *point of interception* (POI) in the operator's network, whose precise location must be handed by the network operator along with the above data.

An operator is expected to support three interfaces called HI1, HI2, and HI3, which are, respectively, used for administration, IRI, and CC.

LI can take place only when requested by an authorized law enforcement agency (LEA). With that the POI must be physically present in the jurisdiction in which the law enforcement has authority, and it is a requirement that the network operator must ensure this. This requirement has an implication for the NFV—specifically for the NFV orchestration and management system—in that the function component that implements POI must always be deployed on the hardware located within the appropriate jurisdiction.[14]

The next major LI requirement that constrains NFV is that of LI being undetectable. All the LI data—and the very fact that the LI takes place—must be contained within the jurisdictional borders and protected from exposure to anyone except those authorized to have access to it by both the law enforcement agency and the network operator.

To summarize, the high-level LI requirements are as follows:

- The LI service capability must always be available.
- The LI service must be activated upon issuing a valid interception order from law enforcement.

[14] We can see now how remote attestation of geographic attributes can be useful in meeting this requirement.

- The LI service must be deactivated when the interception warrant expires (or earlier, if requested).
- The LI service must be invoked on any communication authorized for interception from or to the target visible to the network.
- Interrogation (in the form of operations and management queries) can be admitted only by an LI interface administrator authorized by both the network operator and the law enforcement agency.
- An authorized user for the purposes of interrogation is one who is allowed and authorized by both LEA and the CSP to administer the LI interface.
- LI must not visibly interact with other services (in order to ensure that it is only visible to authorized entities).

For the detail of handling encryption, identities of potential and actual interlocutors, triggers for sending the IRI, and parameters to be enclosed, see [15].

As we noted earlier, the "interrogation" requirement poses a problem in the virtualized environment because of the administrative access to the hypervisor introspection capabilities. As [15] states: "It is very unlikely that the administrator of a conventional hypervisor or orchestrator will be authorized as an interrogator who should be allowed to know that the LI function is activated, and against whom, as information that has to be strictly controlled." The problem is further amplified by the potential capabilities of the analytics software to infer the presence of LI.

The actual LI architecture for virtualized environment is being specified in [16], and the current consensus[15] of the NFV Security Working Group is summarized in Fig. 2.6.

The *LI virtual machine* (LI VM) is to be placed at the optimal POI in the CSP infrastructure to intercept the target traffic. The LI VM then passes the intercepted traffic to the *LI Mediation Function* (MF)/*Delivery Function* (DF), which, in turn, frames the traffic in the standard format and then forwards it to the *Law Enforcement Monitoring Facility (LEMF)* in LEA. There is a range of present implementations of the MF/DFs: from a single MF/DF per POI to an MF/DF being a large concentration point serving multiple POIs.

This is as much as we can say here about handling the LI data. But what about management and control?

The first element here is the *Administrative Function* (ADMF), imported from the legacy environment, which is responsible for administering target warrants and instructing the POI and MF/DFs to take the actions necessary to capture communications of a given target. To perform this function, the ADMF must keep the database of all POIs and MF/DFs under its control. This database is effectively built by the *LI Controller* (LI CTRL), another entity exported from legacy environment, which is responsible for the activation, configuration, and audit of the POIs, as well as for notifying the ADMF that a POI is ready for interception. In the NFV environment, the ADMF has to adapt dynamically to the

[15] As of December 2016.

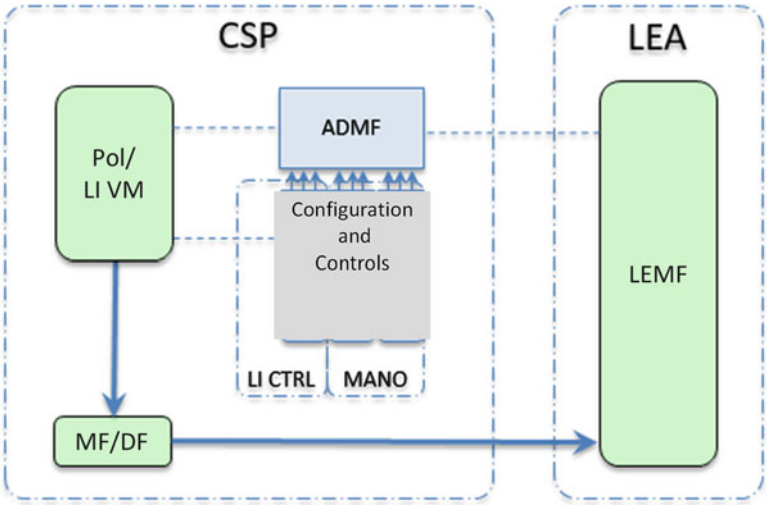


**Fig. 2.6** (Draft) LI architecture in virtualized environment (a simplified version of Fig. 6-2-2 of [16])

newly instantiated (or migrated) POIs and MFs. This can be achieved only through some form of cooperation with the NFV management and orchestration.[16]

In the NFV environment, as [16] observes, "it may be desirable for security reasons to place the MFs outside of the NFV platform in which the LI POIs are implemented. However as the LI POIs move and change in scale, this may make the routing complexity required to backhaul traffic from the LI POIs to the MF/DFs unacceptable. It would potentially be difficult to adequately hide the routing/traffic flows in an SDN connectivity environment." This backhaul problem (known as "trombone effect") is essential for understanding the complexity of the SDN and NFV interactions. This is an open problem. Even though it has first manifested itself during the LI case study, it is likely to arise in other use cases.

In terms of the infrastructure development, [16] suggests that either LEMF be moved into the CSP or a trusted third-party proxy be used to represent the LEMF in the CSP. One problem with implementing this is that national security requirements, which differ across the LEAs, might make this difficult to achieve.

The consensus on the nature of the ADMF is clear. As the ADMF is the root of trust and central point of control, [16] recommends that it be implemented "as standalone hardware which is fully separated from the NFV platform hosting the VNF POIs."

Finally, as far as the respective interfaces (or, *reference points*, in the standards parlance) are concerned, Fig. 2.7 should give a reader a good idea of the current consensus on the subject.

[16] Several such scenarios are discussed in [16].

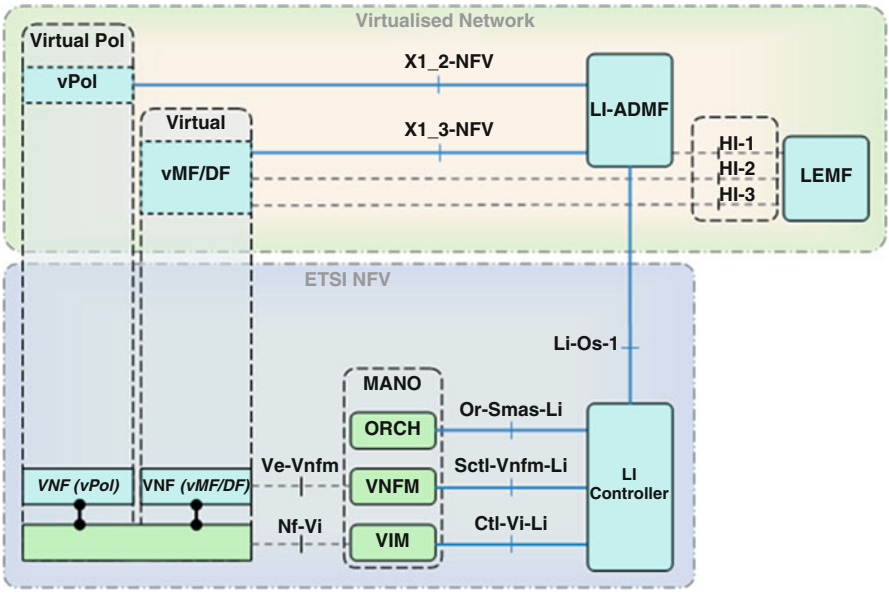


**Fig. 2.7** LI reference points (Fig. 6.5-1 of [16], Draft)

Unfortunately, the space limit of this chapter does not allow us to go into detail of various deployment scenarios (and their respective vulnerabilities and controls). We refer a reader to [16].

We note one particular scenario—the one called the "POI VNF Embedded," in which the POI is part of a VNF. This, of course, is the ultimate NFV-based use case. [16] notes that this scenario, when implemented in conjunction with the security mechanisms for the execution of sensitive components, "most closely provides an equivalent level of LI capability and security to that of an 'on-switch' legacy hardware implementation . . . [which] should address most national security requirements."

This naturally brings us to the subject of the implementation. As should be obvious to a reader now, in order to meet the most basic requirements of LI, the platform must provide both specialized hardware and the capabilities for implementing special security controls.

These have been addressed in [17] in terms of the overall platform hardware and software requirements as well as the life cycle maintenance requirements, system-hardening mechanisms, and identity management controls.

The major requirement is the presence of the HBRT, which is tamper-resistant and tamper-evident and whose interfaces to other hardware components are protected—to a level established by a certification process—from eavesdropping, manipulation, and replay attacks.

With that, a control must be present to restrict (e.g., halt) booting "if assistance from the HBRT is not available or the HBRT currently does not contain valid cryptographic material."

As the HBRT is first and foremost serves as the identification of the platform, it is essential that it be an irremovable part of the host hardware. Any attempt to tamper with the HRBT itself or separate it from the host, must be detected and reported.

An essential task of the HBRT module is key management, which includes creation and deletion of cryptographic keys. HBRT must store the cryptographic material in a "shielded" (i.e., physically protected from an unauthorized access) location. Based on these capabilities, the overall key management system, which also includes access right management, is developed. An essential requirement in terms of services based on HBRT is that the "host system shall provide cryptographically separated secure environments to different applications."

The core software requirements presented in [17] are based on the premise that the HBRT, with the valid cryptographic material be present and its services available. Otherwise, the booting procedure must prevent running of workloads.

With HBRT firmly present, the hardware and software of the system can be molded into the foundation called the *trusted computing base (TCB).* A number of requirements in [17] concern the life cycle of the operation in the presence of the TCB. One of these requirements is that the host system strictly authorize the use of potentially dangerous capabilities (such as memory sharing among virtual machines), with the established default that none such capabilities be available.

The run-time techniques are prescribed to ascertain the level of integrity such of running machines and their respective file systems. This is performed by specialized agents, but those can be also compromised, and so the external behavioral monitoring is also recommended. To run software in a stealth mode, [17] suggests the use of hardware-mediated execution enclaves.[17]

As far as cryptographic algorithms are concerned, [17] has both prescribed and proscribed a number of them, referencing the ISO/IEC standards and the NIST specifications. For communications security, the latest stable versions of the application and network protocols are prescribed.

Further to life cycle-related requirements, [17] refers to a set of prescribed system-hardening and logging techniques, including the operating system-level access and confinement controls as well as physical controls and alarms. The attribute-based access control defined by NIST is declared mandatory. In addition, logging controls are recommended. At the end of the workload life cycle, secure wipe of the relevant storage should be performed. Making provisions for a rainy day, [17] specifies a set of requirements for dealing with the failure conditions.

[17]This term is rather loosely defined in the NFV, but the authors have ascertained two firm implementation examples: (1) that of the Intel's Software Guard Extensions (SGX) (https://software.intel.com/en-us/sgx) and (2) a joint proprietary implementation developed by ARM and Apple (https://www.quora.com/What-is-Apple%E2%80%99s-new-Secure-Enclave-and-why-is-it-important).

While the requirements for the execution of sensitive components naturally apply to the hosts, there is a natural dependency on the operations and management systems (e.g., Management and Orchestration [MANO], attestation authority, certificate authority, or logging systems) to act in concert supporting and enforcing the NFV provider-wide compliance with the requirements.

## 2.6 Security Management and Monitoring

The draft specification [18] is still under development at the time of this writing. A number of factors have been shaping it, and the authors feel that explaining these factors (and also providing some history) will help with understanding the resulting standard.

The work on the subject, or rather the monitoring part of it, started in 2014[18]. The initial objective was to define a security monitoring framework that would provide sufficient material to which analytics could be applied to detect attacks.

The first plan was concrete: to consider specific use cases (such as the *IP Multimedia Subsystem (IMS)* and the *Evolved Packet Core (EPC)* environment) that were of immediate concern to network operators, develop the monitoring solutions for those use cases, and then derive the generic architecture that would support security monitoring for all of these use cases.

But in the beginning of 2015, a proposal for the work on developing an active security management (as opposed to just monitoring) framework came. There was unanimous agreement that the work should proceed, but specifying active controls separately from the mechanisms that trigger them did not make much sense and so the scope of the monitoring work item. As a result, the new work item[19] was created, resulting in a numbering gap, but this was the least controversy.

Over the 2 years of the development of this work (which is expected to be completed in 2017), significant questions were raised as to what should be visible to the monitoring software and what actions it may take. As a reader has probably inferred from the previous section, lawful interception requirements pose a major problem. For one thing, the copied stream would be perceived as anomaly (perhaps even an attack) by a monitoring system, and so all the attributes of an operation that is supposed to be secret would be divulged. Even worse, any action to stop this "attack" would interfere with the lawful interception traffic.

Hence the overarching principle that security management should be confined within a *trust domain.* The latter has been defined in [18] as "a collection of entities that share a set of security policies." Actually, lawful interception is not the only case where trust domain separation—and the confinement of security management to its own a trust domain—is required. Another such case is when a provider of

[18] As part of now extinct work item 8 (https://portal.etsi.org/webapp/workProgram/Report_WorkItem.asp?wki_id=45992).

[19] Work item 12.

the NFV infrastructure hosts a network operator. Naturally, the NFV infrastructure provider's concern is the security of the infrastructure, and so the job of the security management software is to enforce *these* policies rather than secure the operations of the hosted operator. The hosted domain operates under a different set of policies. It may require its own security management operation to enforce those. Alternatively, the infrastructure provider may deliver security management as a service, but in this case, its operation will be distinct from that of the security management of the infrastructure.

There is a detailed discussion in [18] of the use case in which the IMS is deployed on the infrastructure that belongs to a single operator but consists of multiple trust domains.

The life cycle of security management, according to [18], is recursive in that it employs three processes (called phases), which run concurrently and influence one another. The operation starts with the *security planning phase*, in which the security policies are specified for the respective trusted domain. Then, in the *security enforcement* phase, the policies are deployed, at which point the *security monitoring* phase kicks off. The latter observes whether the policies are followed and reports violations to the security enforcement phase, which sends back the updates. Security monitoring may also pass to the security planning phase the request for changes in policies (as, for example, may be required in order to optimize security operations).

We introduce the security management framework, the following discussion accompanied by Fig. 2.8, with the warning to a reader that this is still a work in progress. Some nuances, which will point out in due time, remain to be worked out before the standard is published.

Following the MANO model, [18] defines the VNF layer security function (VSF) for security management of a specific function, a (subordinate) VNF instance security function (ISF) and—to take into account the remaining un-virtualized physical network functions in legacy operations—the physical security function (PSF).

Lest these definitions sound too abstract, [18] provides examples of the VSF, of which we list two: (1) a firewall and (2) a tap for monitoring. The two examples of the ISF are an appliance provided directly by a hypervisor and a hardware box (an HSM, or TPM, or a crypto accelerator).

As far as the management is concerned, there are three blocks. First is the block of traditional security element managers, which enable the NFV security management functional block (NSM-FB)—depicted at the top of the figure. The NSM-FB is in charge of the overall security management. The three phases described earlier are exactly the processes it manages. The NFVI security management functional block (ISM-FB), depicted as part of the MANO virtual infrastructure manager,[20] is responsible for the horizontal management of the virtualization layer. There is a set of requirements specified in [18] that govern the operation of these entities.

[20]This depiction is likely to change as some participants in the NFV Security Working Group share the opinion that security management should not be performed by MANO.

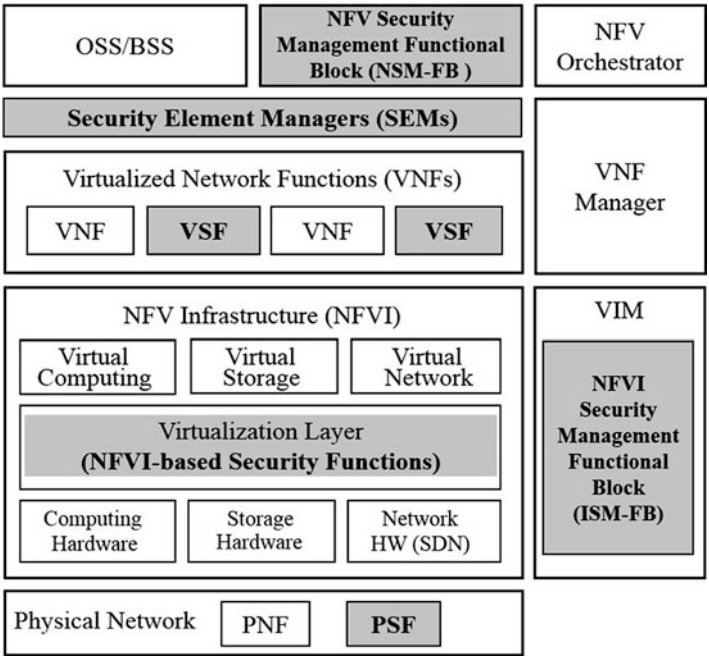


**Fig. 2.8** (After Fig. 6.3-1 of [18]): Security management framework

Without going into the detail, we conclude this section with the note that [18] also defines a separate functional architecture for monitoring. Within this architecture a set of services and a protocol are specified for bootstrapping the trust for the whole infrastructure assuming the existence of the trust chain extending to virtual network functions.

## 2.7 Analysis of the *OpenStack* Security

The work on analyzing the *OpenStack* security was set up at the ETSI NFV Security Group at a very early stage, as the second work item after the NFV Security Problem Statement.[21] The work resulted in the publication of [19]; its findings communicated to *OpenStack* whose contributors were actively involved in writing this document.

Before introducing the findings of [19] (which assumes familiarity with the *OpenStack* architecture), the authors feel that an introduction to the architecture

[21] In fact, for the first 2 year of its existence, the group was an *expert* group (rather than a working group—the status achieved in 2015). As an expert group, the security group was not expected to produce its own documents except for the Problem Statement. Yet, the founders felt that a bottom-up study was necessary both to develop a sound standard and to influence *OpenStack*.

is in order. The *OpenStack* documentation, available at http://www.openstack.org/, is somewhat overwhelming as a first reading because of the sheer amount of detail. This is distilled to a more basic form in [4], to which we refer a reader. Here, we briefly list the most essential facts.

The foundation software components of the OpenStack deal with compute (i.e., host administration), networking, and storage. These are governed by the management functions, which include those of orchestration and identity and access management (to be addressed in the last section of this chapter).

The part of a component that implements an HTTP server (and is thus accessed via a API) is referred to by the OpenStack documentation as a *service*.

Each component is associated with a separate project in charge of its software development. The names of components and their associated projects are used interchangeably in the OpenStack documentation.

The compute component (developed in the project called *Nova*) contains functions that govern the life cycles of all virtual machines. Within the compute, the controller processes—the cloud controller, volume controller, and network controller—take care of the compute resources, block-level storage resources, and network resources, respectively.

The networking component (developed in the *Neutron* project) is concerned with enabling network connectivity for all other components. The services provided by this component support network connectivity and addressing. The native Neutron software presently supports configuring the TLS support for all API and implements Load-Balancer-as-a-Service (LBaaS) and Firewall-as-a-Service (FWaaS).

Neutron also allows to create routers, which are gateways for virtual machines deployed on the nodes that run the Neutron L3 agent software. Among other things, the routers perform NAT translation for the floating IP address—the public IP address that belongs to the cloud provider. It is a unique feature of the Neutron design that this address is not assigned through Dynamic Host Configuration Protocol or set statically. In fact, the guest operating system is unaware of it as the packet delivery to the floating IP address is handled exclusively by the Neutron L3 agent. That arrangement provides much flexibility as the floating (public) and private IP addresses can be used at the same time on any network interface.

To deal with detailed network management, Neutron supports plug-ins—among them that for SDN software. The plug-ins run in the back end. The front-end REST API allows, among other things, to create and update tenants' networks as well as specific virtual routers.

As far as storage is concerned, there are two projects in the OpenStack: Swift and Cinder. The former deals with unstructured data objects, while the latter provides access to the persistent block storage (here again, there is room for plugging in other block storage software).

Also related to storage—of a rather specialized type—is the service component (developed in the Glance project). True to its name, the service deals with storing and retrieving the registry of the virtual machine images. The state of the image database is maintained in Glance Registry, while the services are invoked through Glance API.

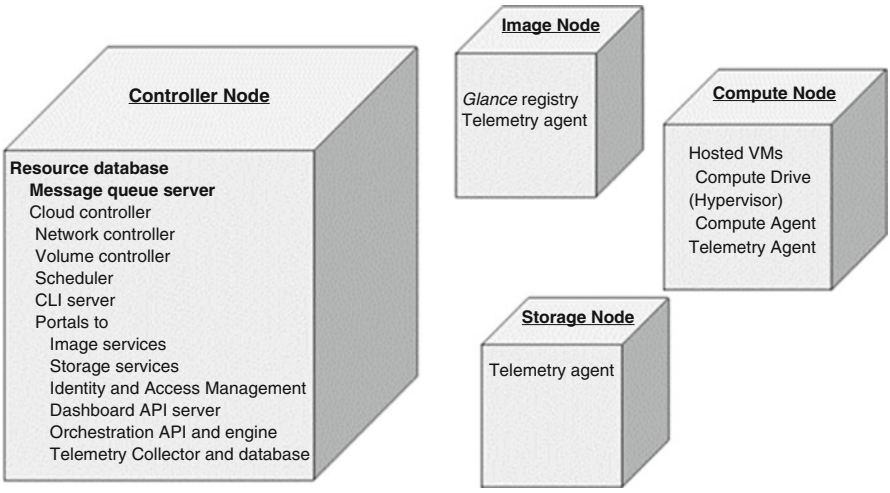


**Fig. 2.9** Deployment example

The authentication and access authorization component is worked in the OpenStack *Keystone* project, which governs the identity and access management. Needless to say, this function was a centerpiece of the security-related study.

Finally, there are three management and orchestration components. The user interface is available both in the "old" CLI form and through the web-based portal, the OpenStack Dashboard, developed as part of the OpenStack Horizon project. Two other components are (1) telemetry, developed in the OpenStack *Ceilometer* project, which is in charge of metering (achieved through monitoring) and (2) service orchestration, developed in the OpenStack *Heat* project.

To give a reader the feel for how these components may be deployed on physical architecture, Fig. 2.9 introduces a four-node deployment example.

The *compute node* is the workhorse of a data center—this is where the virtual workload is deployed. A compute node also runs various applications that belong to the management infrastructure. Some of these applications—called *agents*—initiate interactions with other components (and so act as *clients*); others respond to communications initiated elsewhere (and so act as *servers*). An agent can also be both a client and a server.

The *compute agent* creates and deploys virtual machines. It acts as a server to the *scheduler* (located at the *controller node*), but it acts as a client when dealing with the central resource database, *image node* and *storage node*, which, respectively, maintain the *Glance* image registry and either type (block or object) of storage.

The t*elemetry agents*, present in all three nodes, collect the performance data used in orchestration.

Finally, the *controller node* is in charge of cloud management. To begin with, it contains the global resource *database*. This database is replicated in all practical

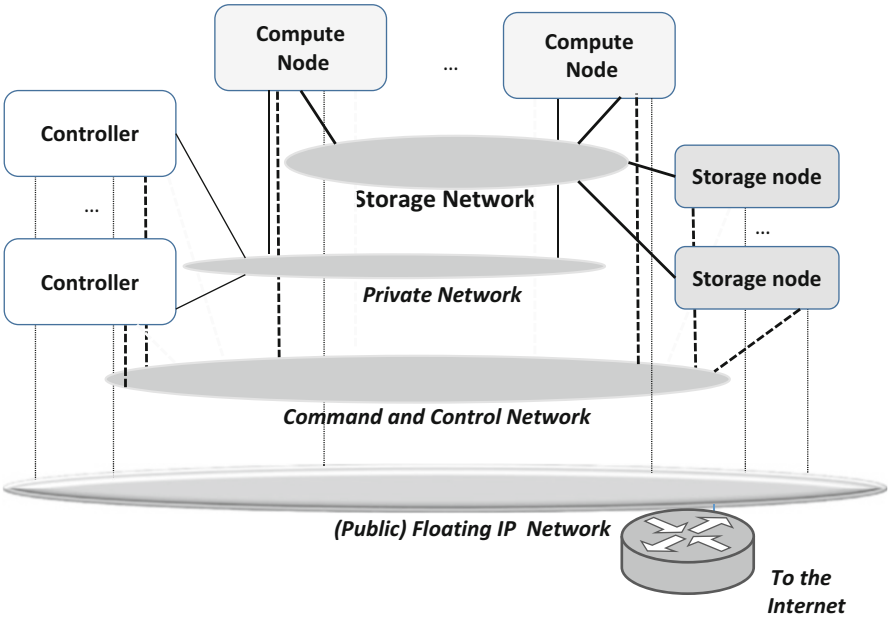


**Fig. 2.10** Physical network isolation in *OpenStack*

deployments—for scaling reasons, and thus it needs a front-end (called *Nova conductor*), which handles the *compute agent* interface.

The *scheduler* is in charge of the placement function. It makes the decision on where (i.e., on which compute node) a new virtual machine is to be created and on which storage node a new block storage volume is to be allocated. For scheduling, the *Nova scheduler* is employed and for storage, the *Cinder scheduler.*

The *Message Queue Server* is the communications center for the messaging among the *OpenStack* API servers representing its components.

The practical deployments follow the principles of isolation outlined in the NFV Security Problem Statement. A typical deployment in a cloud data center is depicted in Fig. 2.10.

There, four networks are completely separate from one another:

- The storage network, which is intended only for accessing storage (and thus interconnects only the compute nodes and storage nodes)
- The private network, which exists only for communications among the hosted virtual machines
- The command-and-control network, which supports orchestration and management
- The public network, which allows connection to the Internet and which, for this reason, employs floating IP addresses

Keeping these networks separate, in addition to aiding security, also help to differentiate the network capacity among the components as their respective bandwidth demands are different.

As we mentioned earlier, [19] addresses all but one problem of the Security Problem Statement. The space of this chapter does not allow us to go into any detail here, except for the most important part—the identity management.

The rest of this section follows [19] in describing Keystone, which, again, is the component that provides centralized authentication and authorization services. As such, it controls access to all API consumed by the rest of OpenStack components.

Keystone works as follows. A user is first authenticated by Keystone.[22] If authentication passes, the user is given a temporary token, which is to be included in all subsequent service requests. The authorization decision is made based on the user's role.

Keystone is organized as a library of internal calls (HTTP requests), which comprise the identity service, token service, and catalog service.

The identity service handles user authentication and user-data validation. Among the constructs used here are the user, project,[23] and user-group identities and the role, whose value is the set of resource access rights. The identity service supports basic operations (e.g., *create*, *read*, *update*, and *delete*). It allows plug-ins for authentication and authorization via a back end module (such as Lightweight Directory Access Protocol (LDAP) servers or an SQL database server, the latter being the default).

The token service supports token management and validation. It relies on a database to store tokens and the token-management data, such as token revocation lists, token lifespan, and token scope—the set of projects and roles associated with the user. Initially, at the authentication time, the token is *unscoped* as no scope is yet defined. The scope of a token is determined by a combination of projects and roles associated with the user. An unscoped token may be issued during the initial authentication of the user, which can then use the token to discover accessible projects and then exchange it for a scoped token.

The token service ensures that tokens be protected from unauthorized access or alteration. Several types of tokens are supported, including public key infrastructure (PKI) (that assume the existence of PKI infrastructure) and the universally unique identifier (UUID), the latter type—defined by the IETF in [20]—being the default. The PKI-type tokens are verified based on the RSA signatures; the UUID tokens are merely random strings. We will discuss both types in more detail in a moment. It is important to note right away that both types of tokens are *bearer tokens*; in other words, a token is a magic wand—whoever possesses it has all the rights associated with it. It follows that it is essential to safeguard a token, for which OpenStack makes special provisions.

[22] Keystone provides the flexibility of employing an external authentication system.

[23] A *project* is defined as a specific set of OpenStack resources.

The catalog service manages the registry of all OpenStack services, supporting the service discovery—including the discovery of addresses of the respective servers. The region is where a server is located, and the characteristic of a server (i.e., public, internal, or administrative) is an attribute that can be defined here, and it is also possible to specify tenant-specific endpoints. As a reader may recall, this feature is essential for meeting the separation requirements for multi-administrative domains.

An important feature of OpenStack is that access permissions can be delegated. See [4] for the explanation and use cases. The construct for delegation is called a *trust.* The trust is implemented as an augmented token, where the delegation-specific information is added. It is created by the delegating party, called a *trustor*, and issued to the *trustee.* The trustor can revoke a trust that it had created.

The scope of the trust is limited to the set of rights that are being delegated. Once created, a trust cannot be changed. Unlike the tokens, the trusts may have unlimited lifetime. This feature is important since it is often unknown when a delegated operation needs to take place. If the lifetime is specified as *infinite*, the trust is valid until it is revoked. The original trustor can allow re-delegation, in which case the trustee may, in turn, become a trustor and delegate the rights it acquired as a trustee to another trustee.

Let us illustrate the use of the UUID and PKI tokens with a (simplified) workflow for provisioning a virtual machine.

We consider the UUID case fist. The workflow starts with the user agent, say *Horizon*, being authenticated by Keystone and, as a result, issued a token in the form of a unique string. (Keystone, which is the only entity that can validate the token, keeps a database, in which the string is associated with the user information.) To create a virtual machine, Horizon sends a request to Nova, enclosing its token. To understand whether the request is valid, Nova has to send it back to Keystone (enclosing its own token so as to allow Keystone to authenticate the transaction). Now Keystone has to look up the date associated with both tokens, first to ensure that the validation request actually came from Nova and, second, to validate that the token passed to it indeed belongs to Horizon and that Horizon has the right to create a virtual machine of the requested type. If all is well, Keystone will respond to Nova positively. We can see that for this transaction, Keystone had to perform two database look-ups. In reality (see [4] for the actual example of what is involved in the actual process of creating a virtual machine), Keystone needs also to talk to Glance and Swift. A reader can see that always going through Keystone make create a performance bottleneck. Again, this is because, UUID tokens can be validated only by Keystone.

In contrast, a PKI token is self-contained. Its structure is depicted in Fig. 2.11.

In this structure, the roles in the domain *SuperTel* are specified as well as the authentication method. The token is protected by the Keystone signature, which can be verified using its certificate. Thus the token can be validated by the receiver without going to Keystone, which eliminates the potential bottleneck and fixing the problem caused by the UUID tokens.

```
            "expires_at": "2017-07-27T22:52:58.852167Z",
            "issued_at": "2016-11-27T21:52:58.852167Z",
            "methods": ["password"],
            "domain": {
                    "id": "3b7650cecd974bf08041328b53a62458",
                    "name": "SuperTelNFV"
                  },
            "roles": [{
                    "id": "7ae2ff9ee4384b1894a90878d3e92bab",
                    "name": "admin"
                      }
                     ],
            "user": {
                "domain": {
                    "id": "3b7650cecd974bf08041328b53a62458",
                    "name": "SuperTelNFV"
                           },
                "id": "3ec3164f750146be97f21559ee4d9c51",
                "name": "EntitledUser"
            }
        }
    }
```


**Fig. 2.11** A PKI token structure

Unfortunately, nothing is simple. The problem is that the size of a PKI token can grow beyond the limit allowed in the HTTP header. This constrains the use of PKI tokens, and OpenStack, after temporarily making this type of a token a default, reverting the default back to the UUID format.

## 2.8 Conclusion

This chapter addresses the network function virtualization (NFV) security while reflecting on the work of the ETSI NFV Security Working Group (NFV SEC WG), and the industry view it has formulated in the past 4 years. The chapter has explained the differences between the "generic" cloud and NFV and discusses the security threats as well as new benefits for security provided in the NFV environment. The chapter further explained how *trust* is bootstrapped from hardware and established among the execution components and introduced the current work on the remote attestation. The requirements and architecture for lawful interception (LI) in the NFV environment, as well as the security monitoring and management in the NFV environment, are treated in much detail. Finally, a separate section is dedicated to the analysis of the *OpenStack* security. There is substantial bibliography offered to a reader who wishes to understand the background and minute detail of the subject.

## 2.9 Review Questions

1. Explain how the NFV environment differs from the generic cloud environment and list as many NFV security challenges and benefits as you can.
2. Explain how the NFV and SDN rely on each other's features in delivering network services and explain the security problems related to service chaining.
3. Explain why hypervisor introspection presents a problem for LI. What is being done to deal with this problem (name specific hardware components)? How can the proposed solution be applied to solving other (non-LI-related) problems?
4. Explain the differences between the TPM and HSM, and give one example for a typical use of each of these two modules.
5. Explain why remote attestation is needed and outline its steps.
6. Outline the architecture for the delivery of security management and monitoring services and explain its interfaces.
7. Explain how *OpenStack Keystone* uses *trusts* for tokens and outline potential security attacks when *bearer tokens* are used. How can tokens be changed to eliminate the security threats you described?

**Acknowledgments** The authors thank all participants of the ETSI NFV Security Working Group for the continuous discussion and development of the very subject of this chapter. In particular, thanks go to the Rapporteurs—who have been leading the work on the group's work items as follows:

*Problem Statement* (work item 1)—Bob Briscoe;
*Cataloguing security features in management software* (work item 2)—Hui-Lan Lu;
*Security and Trust Guidance* (work item 3)—Mike Bursell, Kurt Roemer, and Mihai Serb;
*Report on Lawful Interception Implications (work item 4)*—Scott Cadzow
*Report on Certificate Management* (work item 5)—Markus Wong;
*Report on Security Aspects and Regulatory Concerns* (work item 6)—Scott Cadzow;
*Report on Attestation Technologies and Practices for Secure Deployments* (work item 7)—Diego Lopez and Mihai Serb;
*Report on use cases and technical approaches for multi-layer host administration* (work item 9[24])—Mike Bursell and Anne-Marie Praden;
*Report on Retained Data problem statement and requirements* (work item 10)—Mark Shepherd;
*Security Report on NFV LI Architecture* (work item 11)—Alex Leadbeater;
*Security Management and Monitoring specification* (work item 12)—Ashutosh Dutta, Wei Lu, and Kapil Sood;
*Security Specification for MANO Components and Reference points* (work item 13) and *Security Specification for other MANO reference points* (work item 14)—Pradheepkumar Singaravelu.

We are very grateful to Michael Bilca whose LI architecture figures we re-used. Special thanks go to Don Clarke, whose leadership in the NFV ISG ensured that the work on security got the attention, support, and resources needed to produce the results partially described here.

[24]There is the gap in numbering because of renaming a former work item 8, as explained in Sect. 6.

## References


1. Amzallag David (2014) "NFV Insights: The making of NFV Security—from Vision to Reality. Published by TNT at http://blog.tmcnet.com/next-generation-communications/2014/06/nfv-insights-the-making-of-nfv-security---from-vision-to-reality.html. Retrieved on December 14, 2016
2. ETSI Group Specification (2015) ETSI GS NFV-SEC 001 V1.1.1 (2014–10): Network Function Virtualization; NFV Security; Problem Statement
3. ETSI Group Specification (2015) ETSI GS NFV-SEC 002 V1.1.1 (2015-08): Network Functions Virtualization; NFV Security; Cataloguing security features in management software. Sophia Antipolis, France
4. Faynberg I, Lu H, Skuler D (2016) Cloud computing: business trends and technologies. Wiley, LTF, Chichester
5. Open Network Foundation (2016) TR-530, Threat analysis for the SDN Architecture Version 1.0″ (https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Threat_Analysis_for_the_SDN_Architecture.pdf. Retrieved on December 19, 2016
6. Open Network Foundation (2014) OpenFlow switch specification version 1.3.4 (Protocol version 0x04). https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.4.pdf. Retrieved on December 19, 2016
7. Open Network Foundation (2016) TR-535 "ONF SDN Evolution Version 1.0 ONF." Retrieved on December 19, 2016
8. Edsger W. Dijkstra, EWD 1305. https://www.cs.utexas.edu/ EWD/transcriptions/EWD13xx/EWD1305.html. Retrieved on December 19, 2016
9. Scarfone K, Souppaya M, Hoffman P (2011) Guide to security for full virtualization technologies. Special Publication 800-125. National Institute of Standards and Technology. US Department of Commerce
10. Trusted Computing Group (2011) Virtualized trusted platform architecture specification. http://www.trustedcomputinggroup.org/virtualized-trusted-platform-architecture-specification/. Retrieved in December 2016
11. ETSI Group Specification (2016) GS NFV-SEC 003 V1.2.1. Network Functions Virtualization (NFV); NFV Security; Security and Trust Guidance
12. ETSI Group Specification (2016) GS NFV-SEC 010 V1.1.1. Network Functions Virtualization (NFV); NFV Security; Report on Retained Data problem statement and requirements
13. Draft ETSI Group Specification ETSI GS NFV SEC 007 Network Functions Virtualization (NFV); NFV Security; Trust; Report on Attestation Technologies and Practices for Secure Deployments. Work in progress.https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=44578. Retrieved on December 5, 2016
14. Simpson AK, Schear N, Moyer T (2016) Runtime integrity measurement and enforcement with automated whitelist generation. In: Proceedings of annual computer security applications conference (ACSAC). Available at https://homes.cs.washington.edu/ aksimpso/publications/ACSAC2014Abstract.pdf. Retrieved on December 5, 2016
15. ETSI Group Specification NFV-SEC 004 V1.1.1 (2015) Network Functions Virtualisation (NFV); NFV Security; Privacy and Regulation; Report on Lawful Interception Implications
16. Draft ETSI Group Specification NFV SEC 11 V0.0.6 (2016–05) Network Functions Virtualization (NFV); NFV Security; Trust; Report on Report on NFV LI Architecture. Work in progress. https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=47603. Retrieved on December 12, 2016
17. Draft ETSI Group Specification NFV-SEC 012 V0.0.13 (2016) Network Functions Virtualisation (NFV); Security; System architecture specification for execution of sensitive NFV components. Work in progress. https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=47619. Retrieved on December 15, 2016

18. Draft ETSI Group Specification NFV-SEC 013 V0.0.6 (2016) Network Functions Virtualisation (NFV); Security Report; Security Management and Monitoring for NFV [Release 2]. Work in progress. https://portal.etsi.org//tb.aspx?tbid=799&SubTB=799. Retrieved on December 21, 2016
19. ETSI Group Specification NFV-SEC 002 V1.1.1 (2015–08) Network Functions Virtualisation (NFV); NFV Security; Cataloguing security features in management software
20. Leach P, Mealling M, Salz R (2005) RFC 1422, A Universally Unique IDentifier (UUID) URN Namespace. (https://tools.ietf.org/html/rfc4122)

**Igor Faynberg** a 2011 Bell Labs Fellow, is an industry consultant and an Adjunct Professor of Computer Science in Stevens Institute of Technology. He represents Cable Television Laboratories in the ETSI NFV ISG, where he has been chairing the Security Working Group for the past 4 years.

Prior to founding the Stargazers Consulting LLC in 2015, Dr. Faynberg had had various staff and management positions in Bell Labs and Alcatel-Lucent business units where he had contributed to a range R&D projects, starting from the development of variants of Karmarkar algorithm for supercomputers, Intelligent Network, and its interworking with the Internet to cloud computing and network functions virtualization. Most recently, he directed a group that researched solutions for security and identity management problems and led their standardization in the ATIS, IETF, ITU-T, ISO/IEC, ETSI, and INCITS Cyber Security Committee.

Prior to joining Bell Labs in 1986, Dr. Faynberg had contributed to design and development of operating systems and a hypervisor as well as a network management suite for the Sperry Distributed Communications Architecture and designed the Local Area Networking architecture and protocols for the Burroughs Network Architecture.

Dr. Faynberg holds over 50 US and international patents for inventions relevant to converged services, data communications, and security, and he has over 30 refereed publications in the area of application of computer science to communications and network security. He has co-authored three books entitled, respectively, Intelligent Network Standards, Their Applications to Services (McGraw-Hill, 1997), Converged Networks and Services: Internetworking IP With PSTN (John Wiley & Sons, 2000), and Cloud Computing—Business, Trends, and Technologies (John Wiley & Sons, 2016).

He holds an M.A. in mathematics from Kharkov University, Ukraine, and M.S. and Ph.D. degrees in Computer and Information Science from the University of Pennsylvania, Philadelphia.

**Steve Goeringer** is a principal security architect at CableLabs working on emerging technologies and innovation projects. He has recently worked on security of network functions virtualization (NFV), software-defined networking (SDN), medical devices, and cable modems architecture. He has also been investigating innovations in integrating cryptography into cameras, block chain solutions for the cable industry, and new approaches to securing home networks. He is currently supporting the Center for Medical Interoperability as the chairperson of the security working group.

Prior to working at CableLabs, Steve had worked as a consultant for Polar Star Consulting, LLC, providing technology leadership to government agencies. In this role, he researched WAN acceleration solutions, investigated Ethernet security, and performed engineering and technical selection of nationwide optical networks. Before that, he fulfilled several engineering roles at Qwest, including Technical Director of the Access and Transport Networks team.

Steve spent 12 years at the National Security Agency where he was a Master Intelligence Analyst. He started his career in the US Army as a Communications Station Technical Controller. Steve has a Bachelor of Science degree in computer and information science.