ETSI GS NFV-SEC 001 V1.1.1 (2014-10)

Network Functions Virtualisation (NFV); NFV Security; Problem Statement

Reference

DGS/NFV-SEC001

Keywords

NFV, security

Contents

Intellectual Property Rights 5

Foreword 5

Modal verbs terminology 5

1 Scope 6

2 References 6
2.1 Normative references 6
2.2 Informative references 6

3 Definitions and abbreviations 8
3.1 Definitions 8
3.2 Abbreviations 9

4 Industry Context 10

5 Security Reference Framework 11
5.1 Deployment Scenarios 11
5.2 Threat Surface 12
5.3 Attacker Profiles 13

6 Potential Areas of Concern 15
6.1 Topology Validation & Enforcement 15
6.1.1 Topology Validation Example 15
6.1.2 Validating the Topology of Virtualised Network Functions 17
6.1.3 Validating the Topology of the Infrastructure Network 18
6.1.3.1 SDN-specific issues 18
6.1.3.1.1 Hierarchies of control 19
6.1.3.1.2 Additional layers of hierarchy 20
6.1.3.1.3 Further issues 20
6.1.3.2 Issues specific to distributed (non-SDN) routing 20
6.1.3.3 Related issues 21
6.1.4 Topology Policy Validation & Enforcement 21
6.2 Availability of Management Support Infrastructure 21
6.3 Secured Boot 23
6.3.1 Background 23
6.3.2 Secure Boot and Trusted Boot Technology 24
6.3.3 Secured Boot Summary 25
6.4 Secure crash 25
6.5 Performance isolation 26
6.5.1 Network and I/O partitioning 27
6.5.2 Shared core partitioning 27
6.5.3 Acceleration hardware partitioning 27
6.5.4 Shared memory partitioning 28
6.5.5 Attacks on the resources of the virtualisation infrastructure 28
6.6 User/Tenant Authentication, Authorization and Accounting 29
6.7 Authenticated Time Service 30
6.8 Private Keys within Cloned Images 31
6.9 Back-Doors via Virtualised Test & Monitoring Functions 31
6.9.1 Operational Test, Monitoring and Fault Tracing 31
6.9.2 Developer's debug and test interfaces 32
6.9.2.1 Industry Experience Example 1 32
6.9.2.2 Industry Experience Example 2 33
6.9.2.3 Guidance for the NFV ISG working groups: 33
6.10 Multi-Administrator Isolation 33

7 General Concerns 33
7.1 Safety vs. Complexity 33

7.2 Altered Norms and Procedures 34
7.2.1 Broken Procedure 34
7.2.2 Software Vulnerabilities 34

Annex A (informative): Authors & contributors 36

History 37

Intellectual Property Rights

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**may not**", "**need**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document aims to:

- To identify potential security vulnerabilities of NFV and to determine whether they are new problems, or just existing problems in different guises.
- To provide a reference framework within which these vulnerabilities can be defined.

Out of scope: To list vulnerabilities that NFV suffers from that are no different from pre-existing vulnerabilities of networking and virtualisation technologies and are not altered by the virtualisation of network functions.

Intended audience: Security experts wanting to deploy NFV but needing to identify and solve potential security issues and then to attain security accreditation for systems.

Ultimate goal of the NFV Security Expert Group: Identify and propose solutions to any new vulnerabilities that result from the introduction of NFV. To enable checks for these vulnerabilities to be incorporated into processes for security accreditation of products based on NFV.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Homomorphic Encryption.

NOTE: http://en.wikipedia.org/wiki/Homomorphic_encryption.

[i.2] Trusted Computing Group.

NOTE: http://www.trustedcomputinggroup.org/.

[i.3] Unified Extensible Firmware Interface (UEFI) forum.

NOTE: http://www.uefi.org/home/.

[i.4] ISO/IEC 11889-1 (March 2009(en)): "Trusted Platform Module - Part 1: Overview".

[i.5] CERT Vulnerability Note VU#362332 (August 2010).

[i.6] NIST SP 800-147 (April 2011): "Basic Input/Output System (BIOS) Protection Guidelines".

[i.7] NIST SP 800-155 (December 2011): "DRAFT BIOS Integrity Measurement Guidelines".

[i.8] TPM Main Specification (March 2011).

NOTE: http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

[i.9] Virtualized Trusted Platform Architecture Specification (September 2011).

NOTE: http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

[i.10] NIST SP 800-147b (July 2012): "DRAFT BIOS Protection Guidelines for Servers".

[i.11] NFV White paper (October 2012): "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action. Issue 1".

NOTE: http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[i.12] ETSI GS NFV 002: "Network Functions Virtualisation (NFV); Architectural Framework".

[i.13] ETSI GS NFV 001: "Network Functions Virtualisation (NFV); Use Cases".

[i.14] ETSI GS NFV INF 001-1 (V0.3.8 April 2014): "Network Functions Virtualisation; Infrastructure Architecture; Sub-part 1: Overview", (work in progress).

[i.15] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.16] David Kleidermacher and Mike Kleidermacher: "Embedded Systems Security", Newnes, April 2012.

[i.17] Rowan Klöti, Vasileios Kotronis and Paul Smith: "OpenFlow: A Security Analysis". In Proc. Wkshp on Secure Network Protocols (NPSec). IEEE, October 2013.

[i.18] Diego Kreutz, Fernando M.V. Ramos and Paulo Verissimo: "Towards secure and dependable software-defined networks". In Proc. 2nd ACM SIGCOMM workshop on Hot topics in software defined networks, HotSDN '13, pages 55-60, New York, NY, USA, 2013. ACM.

[i.19] ONF Security Discussion Group.

NOTE: https://www.opennetworking.org/working-groups/discussion-groups.

[i.20] OpenFlow Switch Specification.

NOTE: Available via http://archive.openflow.org/wp/documents/.

[i.21] Recommendation ITU-T Y.3500 (July 2014) | International Standard ISO/IEC 17788 "Information technology - Cloud computing - Overview and Vocabulary".

[i.22] Thomas Ristenpart, Eran Tromer, Hovav Shacham and Stefan Savage: "Hey, You, Get Off of My Cloud! Exploring Information Leakage in Third-Party Compute Clouds". In Proc. Conference on Computer and Communications Security (CCS'09), pages 199--212. ACM, November 2009.

[i.23] Dawn Song, David Wagner and Adrian Perrig: "Search on Encrypted Data". In Proc. IEEE Symposium on Security and Privacy, May 2000.

[i.24] IETF draft-mrw-sdnsec-openflow-analysis-02 (April 2013): "Security Analysis of the Open Networking Foundation (ONF) OpenFlow Switch Specification", Margaret Wasserman and Sam Hartman, (work in progress).

[i.25] IEEE 802.1ah: "Provider Backbone Bridges".

[i.26] IEEE 802.1ad: "Provider Bridges".

[i.27] IETF draft-mahalingam-dutt-dcops-vxlan-09 (April 2014): "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", M. Mahalingam and others, (work in progress).

[i.28] IETF draft-davie-stt-06 (April 2014): "A Stateless Transport Tunneling Protocol for Network Virtualization (STT)", Bruce Davie and Jesse Gross, (work in progress).

[i.29] IETF draft-sridharan-virtualization-nvgre-04 (February 2014): "NVGRE: Network Virtualization using Generic Routing Encapsulation", Murari Sridharan and others, (work in progress).

[i.30] IETF RFC 3031 (January 2001): "Multiprotocol Label Switching Architecture", Eric Rosen, Arun Viswanathan and Ross Callon.

[i.31] ETSI/TC LI#35 LI(14)P35037r2 "NFV LI Considerations".

[i.32] NFV White paper (October 2013): "Network Functions Virtualisation, Network Operator Perspectives on Industry Progress".

NOTE: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ETSI GS NFV 003 [i.15] and the following apply:

Forwarding Function: provides forwarding connectivity between multiple (two or more) network interfaces

NOTE 1: This distinguishes a Forwarding Function from just any networked application. A Forwarding Function receives data at any of the interfaces, processes it, then outputs some transformation of the original to other network interface(s).

EXAMPLE: Message routing or the filtering provided by a firewall. Some examples of other network functions are shown in Figure 1.

NOTE 2: 'Some transformation' is necessarily vague. It is meant to preclude server applications that might take in requests on one interface and send out responses on another. But it is meant to include, say, deep packet inspection or a packet filter that takes in data packets on one interface and forwards most to an output interface, but discards or re-orders some packets in the process. It also includes switches or network address translators that largely forward the data unchanged, but make a few focused changes to addresses. It even includes meters that take a packet flow as input and output a stream of measurement data that summarises the characteristics of the input flows.

NOTE 3: Although a Forwarding Function is defined by its multiple data interfaces, the definition does not preclude other interfaces for control (e.g. routing messages) and management - and these are certainly within scope of security problem analysis.

NOTE 4: All Forwarding Functions are Network Functions, but some Network Functions do not involve forwarding. For instance, most directory, control or management functions are Network Functions but not Forwarding Functions.

hypervisor: computer software, firmware or hardware running on a host computer that creates, runs and monitors guest virtual machines.

NOTE: A hypervisor enables multiple instances of a variety of guest operating systems to share the virtualised hardware resources of the host.

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GS NFV 003 [i.15] and the following apply:

AAA	Authentication, Authorization and Accounting
API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
BGP	Border Gateway Protocol (IETF)
BIOS	Basic Input/Output System
CA	Certification Authorities
CPU	Central Processing Unit
DMA	Direct Memory Access
ECDSA	Elliptic Curve Digital Signature Algorithm
FB	Functional Block
FTP	File Transfer Protocol (IETF)
GRE	Generic Routing Encapsulation (IETF)
GS	Group Specification (ETSI)
I/O	Input/Output
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IOMMU	I/O Memory Management Unit
ISG	Industry Specification Group (ETSI)
IS-IS	Intermediate System to Intermediate System (IETF)
ISO	International Organisation for Standardization
IT	Information Technology
JTAG	Joint Test Action Group
LAN	Local Area Network
LI	Lawful Interception (ETSI TC)
LOM	Lights-Out Management
MAC	Medium/Media Access Control
MANO	Management & Orchestration
MMU	Memory Management Unit
MPLS	Multi-Protocol Label Switching (IETF)
NAT	Network Address Translator
NFV	Network Functions Virtualisation
NFVIaaS	NFV Infrastructure as a Service
NIST	National (US) Institute of Standards and Technology
NOC	Network Operations Centre
NV-GRE	Network Virtualisation using GRE
ONF	Open Networking Foundation
OS	Operating System
OSPF	Open Shortest Path First (IETF)
PCI	Peripheral Component Interconnect
QoS	Quality of Service
RSA	Rivest-Shamir-Adleman
SDN	Software Defined Network
SMMU	System MMU (ARM)
SR-IOV	Single Root I/O Virtualisation (a PCI special interest group standard)
STT	Stateless Transport Tunnelling
TC	Technical Committee
TCG	Trusted Computing Group
TLS	Transport Layer Security (IETF)
TPM	Trusted Platform Module (TCG)
UEFI	The Unified Extensible Firmware Interface forum
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtualised Network Function
VNFaaS	VNF as a Service
VNPaaS	Virtualised Network Platform as a Service
VPN	Virtual Private Network
VT-c	Virtualisation Technology for connectivity (Intel)

VT-d	Virtualisation Technology for directed I/O (Intel's IOMMU)
VXLAN	Virtual eXtensible LAN

4 Industry Context

Traditionally, Network Functions have been bundled into bespoke hardware appliances. In contrast, network functions virtualisation (or NFV) is the deployment of these services as software modules that run on common off-the-shelf generic hardware [i.11] over a hypervisor or container that controls access to the hardware devices.

Network function virtualisation has become economic because the sheer scale of the data centre market has drawn investment and skills towards generic server technology; 9 million IT servers are bought globally each year, but only 180 thousand edge routers. It is safe to predict that a network equipment facility will become physically the same as a data centre. Virtualised network functions will also be managed in common with IT management processes - as orchestrated remote software installations deployed independently to hardware upgrades.

The transition towards network functions virtualisation will be incremental. As each bespoke appliance reaches the end of its life it will be replaced by a software equivalent. Independently, more server blades, storage or network interface cards will be plugged in to existing racks to provide the necessary hardware resources. For further information, see [i.11] and for an update on industry activity on NFV see [i.32].

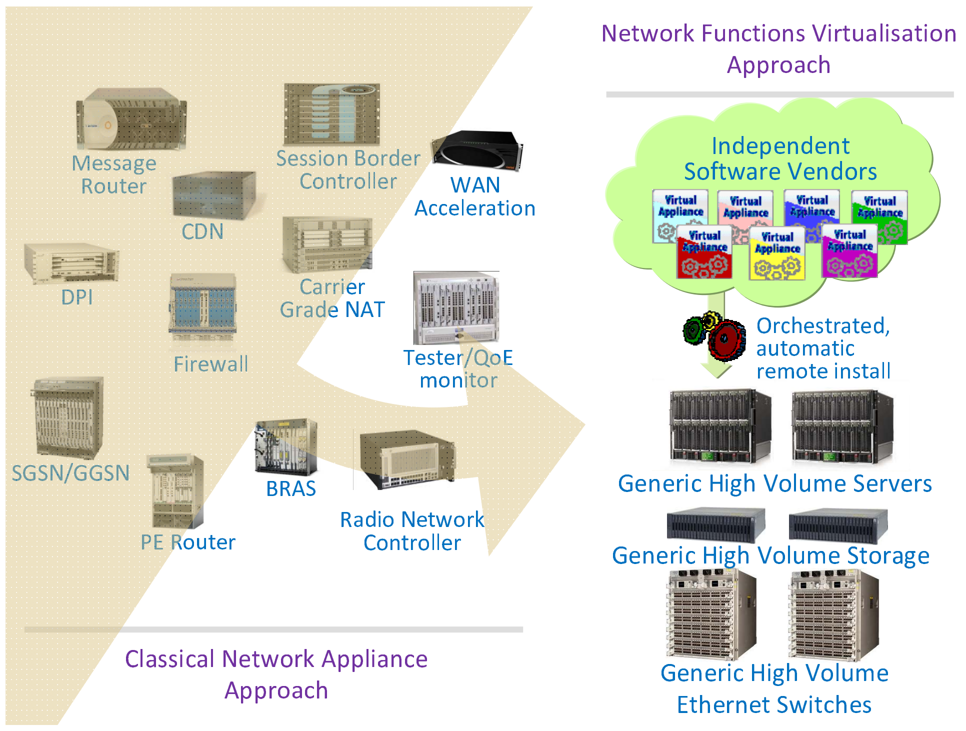

Network Functions Virtualisation Approach
Message Router
Session Border Controller
WAN Acceleration
CDN
DPI
Carrier Grade NAT
Firewall
Tester/QoE monitor
SGSN/GGSN
BRAS
PE Router
Radio Network Controller
Classical Network Appliance Approach
Independent Software Vendors
Virtual Appliance
Orchestrated, automatic remote install
Generic High Volume Servers
Generic High Volume Storage
Generic High Volume Ethernet Switches

Figure 1: Examples of Network Functions showing the Incremental Process of Virtualisation

NOTE 1: Network functions virtualisation should not be confused with virtual networks like virtual local area networks (VLANs) or virtual private networks (VPNs) - the two concepts are orthogonal. Another term for virtual networks is overlay networks, and again NFV is orthogonal to overlays. A virtual network is a logical partition carved out of a physical network by ensuring its logical connectivity is isolated from other virtual networks. In contrast, NFV only concerns whether the functions of the nodes of a network are implemented as software on generic hardware and hypervisor technology, rather than on bespoke hardware.

NOTE 2: The scope of this problem statement does not include cases where there is no hypervisor or container. In other words, network functions running as software over a monolithic operating system, even if on industry-standard hardware are out of scope of the present document, given that bare metal introduces no security changes relative to the baseline (which is bare metal).

5 Security Reference Framework

5.1 Deployment Scenarios

Figure 2 illustrates the elements with potentially separate security responsibility in different deployment scenarios: the building, the host compute hardware, the hypervisors and the guest virtual network functions within their virtual machines.

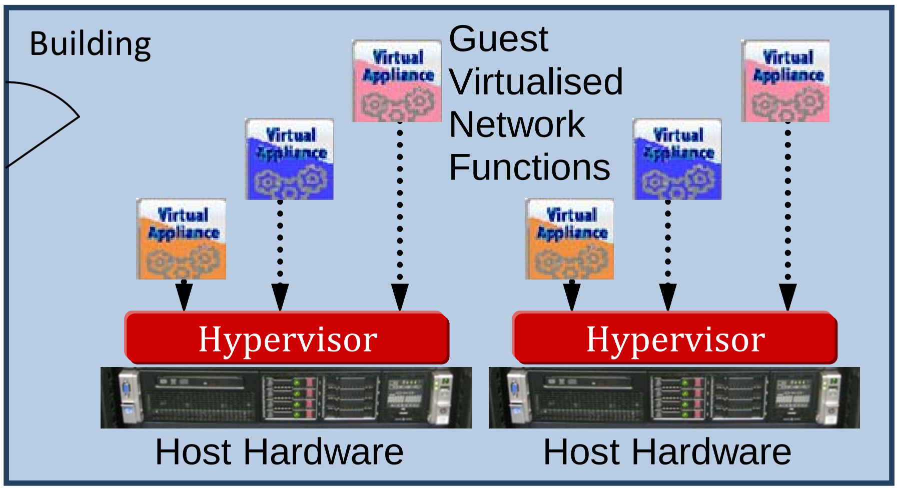


Figure 2: Deployment scenario elements

Below some deployment scenarios are described that are likely in realistic contractual arrangements.

For convenience they are summarised in Table 1. The right-most column also identifies which NFV deployment scenarios are similar to the common deployment models used in cloud computing, as identified by the ITU-T [i.21]. It can be seen that the cloud computing industry uses deployment models that sometimes, but not always, relate to those expected for NFV. The ITU-T document [i.21] also defines Cloud Service Models (Infrastructure, Platform and Software as a Service). It is possible that virtualisation of network functions might eventually become commoditised into a set of similar service models, but the NFV industry needs to be allowed to mature before jumping to conclusions on popular service models.

Table 1: Some realistic deployment scenarios

Deployment Scenario	Building	Host Hard-ware	Hyper-visor	Guest VNF	cf. ITU-T Cloud Vocabulary
Monolithic Operator	N	N	N	N	Private Cloud
Network Operator Hosting Virtual Network Operators	N	N	N	N, N1, N2	Hybrid Cloud
Hosted Network Operator	H	H	H	N	
Hosted Communications Providers	H	H	H	N1, N2, N3	Community Cloud
Hosted Communications and Application Providers	H	H	H	N1, N2, N3, P	Public Cloud
Managed Network Service on Customer Premises	C	N	N	N	
Managed Network Service on Customer Equipment	C	C	N	N	
NOTE: The different letters represent different companies or organisations, and are chosen to represent different roles, H = hosting provider, N = network operator, P = public, C = customer.					

Monolithic Operator: The same organisation that operates the virtualised network functions deploys and controls the hardware and hypervisors they run on and physically secures the premises in which they are located.

Network Operator Hosting Virtual Network Operators: Based on the 'Monolithic Operator' model, except as well as hosting itself, the network operator hosts other virtual network operators within the same facility. It would probably isolate each virtual operator on separate hardware. However, in theory, the virtual machines of different virtual network operators could run alongside each other over the same hypervisor.

Hosted Network Operator: An IT services organisation operates the compute hardware, infrastructure network and hypervisors on which a separate network operator runs virtualised network functions. The premises including cable chambers, patch panels, etc. are physically secured by the IT services organisation.

Hosted Communications Providers: Similar to 'Hosted Network Operator', except the IT services organisation hosts multiple communications providers. Alternatively, the IT services organisation may host one (or many) wholesale network operators, which in turn host multiple virtual retail communications providers. In this latter case the IT services organisation would give controlled rights to run virtualised network functions to the wholesaler, which could in turn delegate rights to the virtualised retailers.

Hosted Communications and Application Providers: Similar to 'Hosted Communications Providers' except servers in a data centre facility are offered to the public for deploying virtualised applications (Cloud) while in the same physical facility network operators and communications providers deploy virtualised network functions on the same type of generic hardware platforms (probably not sharing the same server hardware, but in blades and racks alongside and sharing the same data centre network).

Managed Network Service on Customer Premises: A network operator runs virtualised network functions on its own generic server hardware located on a customer's premises and physically secured by the customer.
This model could be in a residential or enterprise scenario, for example, respectively, a remotely managed home gateway or remotely managed VPN gateways, firewalls, etc.

Managed Network Service on Customer Premises Equipment: Similar to 'Managed Network Service on Customer Premises', except the compute hardware is supplied and operated by the customer not the network operator. The customer allocates a blade to the network operator, which runs a hypervisor on this blade, and in turn runs all the necessary network functions within virtual machines over this hypervisor.
This model does not involve the customer running virtual machines on the same hypervisor and hardware as the network operator, although this would be another valid scenario (similar to 'Hosted Network Operator' except the customer both hosts the virtual network functions of the network operator and runs guest applications in virtual machines alongside them).

In general, in order to determine the security implications of a deployment scenario, the two main factors are:

1) The different parties that operate each of the levels (building, host hardware, hypervisor, guest VNF).

2) Whether the party at any one layer has exclusive or non-exclusive use of the resources of the lower layers and, if non-exclusive, are the resources available to all other parties or only to a restricted set (e.g. only allied operators? competing operators? the general public?)

5.2 Threat Surface

This problem statement focuses solely on threats that are specific to NFV (as compared to the known generic virtualisation threats or known generic networking threats). With proper securing of the infrastructure, NFV could actually offer security benefits by improving the inherent security properties of network functions.

Since a VNF is but a network function running on a virtual machine, the set of all the security threats to a network comprising VNFs, at the first approximation, is a union of:

1) all the generic virtualisation threats (e.g. memory leakage, interrupt isolation);

2) the threats specific to the system of physical network functions prior to virtualisation (e.g. flooding attacks, routing security);

3) new threats due to combining virtualisation technology with networking. These new threats are shown as the intersection of the two sets in Figure 3, which is the focus of this problem statement.

This observation can be corrected though when taking into account a major security benefit of virtualisation: the hypervisor can eliminate some and mitigate other threats inherent to non-virtualised network functions through the use of hypervisor introspection and other techniques. In other words, virtualisation offers the real possibility of virtualisation improving the security of a given network compared to its physical counterpart, as represented by the hole in the intersection of the sets in Figure 3.

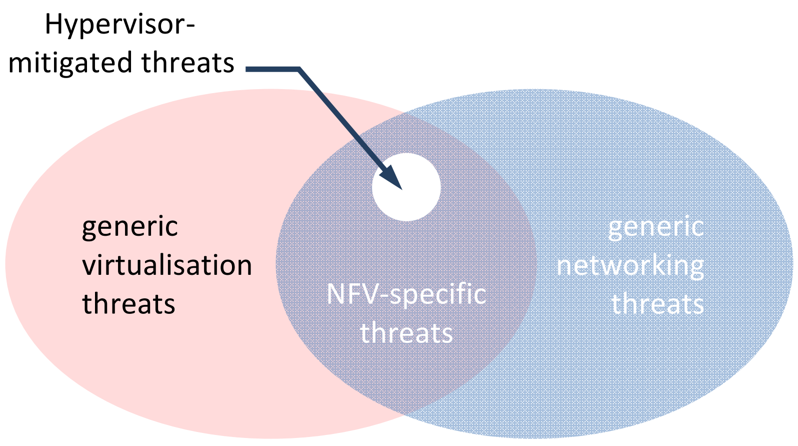

Hypervisor-mitigated threats
generic virtualisation threats
NFV-specific threats
generic networking threats

Figure 3: Visualisation of the NFV threat surface

Hence the duality of NFV security-the introduction of new threats due to virtualisation also gives rise to new opportunities.

It follows that the approach to improving security of a network of VNFs will be two-pronged so as to combine 1) "shrinking" as much as possible the intersection between the sets by securing the NFV infrastructure and 2) "carving" as large a "hole" as possible out of the intersection by applying appropriate hypervisor-based security techniques.

The problems identified in this NFV Security Problem Statement outline threats identified so far at the intersection of virtualisation and networking, which, when contained,should bring the NFV infrastructure to a sufficiently secure level to be deployable.

5.3 Attacker Profiles

It is traditional to profile attackers by means, motive and opportunity. For instance, attackers could be profiled by their degree of co-ordination and motivation:

- Disaffected or criminal members of the general public.
- Co-ordinated small groups, such as criminal gangs or terrorist organisations.
- Co-ordinated large organisations with a political, religious or commercial agenda, e.g. government agencies, corporations.

Introducing NFV makes little difference to such pre-existing motives, but it does alter the means and opportunity to exploit a vulnerability, to a degree that depends on the technical and contractual position of an organisation in relation to others in the supply of NFV. Therefore, we define a contractual hierarchy of types of organisations or individuals in the supply of NFV technology:

- End-customers of retail network operators.
- Retail network operators (sometimes called virtual network operators, but we will avoid that term here because it is confusing in an NFV context where both real and virtual network operators use virtualised machines).
- Wholesale network operators (in some markets there may be no wholesale/retail split, in other markets there may be multiple levels of wholesaling).

- Hypervisor operator - usually the same party as either the infrastructure operator or the wholesale network operator.
- Infrastructure operators - the party that operates the compute, storage and infrastructure network.
- Facilities manager - the party that secures the physical building, racking, power, cabling, etc.

The last three parties listed above map directly to the lowest three levels of figure 2, while there is sufficient flexibility within the guest Network Functions in figure 2 to allow the higher levels of the hierarchy to be created in all manner of different arrangements to suit market conditions. A party at one level voluntarily contracts with the parties operating lower levels to provide hosted service. This implies that the hosted party places at least a degree of trust in its underlying hosts (e.g. those operating the security of the building facilities, those with operational access to the hardware).

Therefore, attacks of initial interest are likely to be confined to those:

1) from a higher level, e.g. an end-customer attacking her network operator or another operator;
2) across from the same level, e.g. a network operator extracting information about a competitor sharing the same facilities, or one end-customer perverting their network service in order to attack another end-customer;
3) from inside, e.g. a disgruntled employee of a network operator.

Nonetheless, we cannot preclude 'upward' attacks on a guest by its own hosting provider. A hosting company could mount nearly any attack on its own guests, but it would be unlikely to attempt any attack that degraded its own service, for obvious reasons of reputation. A hosting provider might however be tempted to mount an attack that its guest would not be aware of, such as exploiting or selling confidential information like customer records. Alternatively, an insider within the hosting company might degrade a guest network operator's service in more subtle ways, perhaps in order to enable a secondary attack. For instance, an insider within the hosting company might open a hole in a guest's firewall, or suppress certain intrusion detection messages.

Intellectual Property related threats also exists in the NFV environment. Various providers of NFV functions will run proprietary algorithms, images, static and runtime configuration files and signatures, that they would want to be protected. Reverse engineering and side channel attacks need to be mitigated to protect the intellectual property of various vendors running on the same platform from each other and from the platform operator.

Technology is readily available (discussed further in clause 6.3) that enables a guest to store keys that it trusts and to do limited processing within a tamper-resistant module on the host computer. This can provide a technological basis to underpin the contractual trust that a guest places in its hosting service. However, it is generally infeasible for all of a guest's computing functions to be concealed from the host within such a secure element, given its limited memory, processing and I/O resources. An alternative is partial homomorphic cryptography, which is becoming practical for certain very specific functions without too much overhead (e.g. a guest can ask a hosting service to search through encrypted data for an encrypted search term and return an encrypted result, so the hosting service never holds the data in the clear even within its processors [i.23]). Extending applicability from single specific functions to any function would require fully homomorphic encryption, which has been implemented as an early proof of concept but the outputs degrade as the number of processing steps increases. Although advances are being made rapidly, it is unknown how soon (if ever) practical implementations will become feasible [i.1].

6 Potential Areas of Concern

The key issues submitted by participants in the first ETSI NFV industry specification group (ISG) meeting have been analysed to identify those related to security. They and additional issues identified within the NFV Security Expert Group are consolidated into the present document.

Table 2 lists these issues and the identified subsequent clauses describe each issue.

Table 2: Summary of Potential Areas of Concern

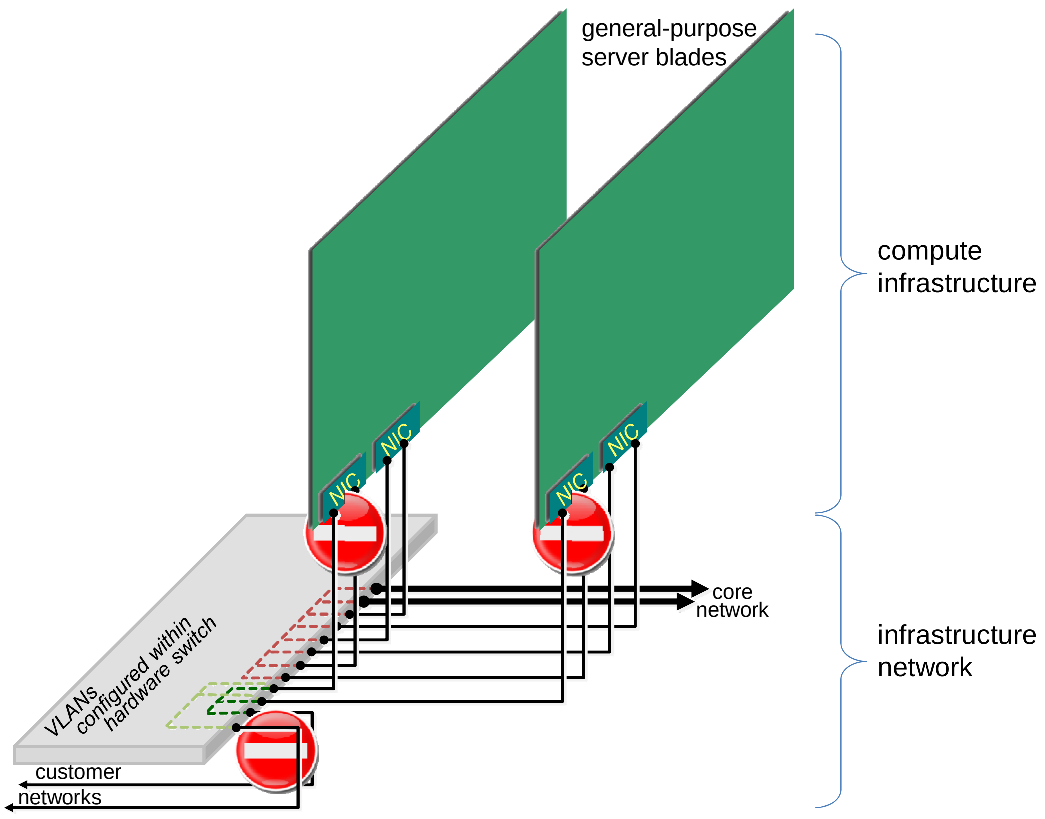
Clause	Key Issue
6.1	Topology Validation & Enforcement
6.2	Availability of Management Support Infrastructure
6.3	Secured Boot
6.4	Secure crash
6.5	Performance isolation
6.6	User/Tenant Authentication, Authorization and Accounting
6.7	Authenticated Time Service
6.8	Private Keys within Cloned Images
6.9	Back-Doors via Virtualised Test & Monitoring Functions
6.10	Multi-Administrator Isolation

6.1 Topology Validation & Enforcement

6.1.1 Topology Validation Example

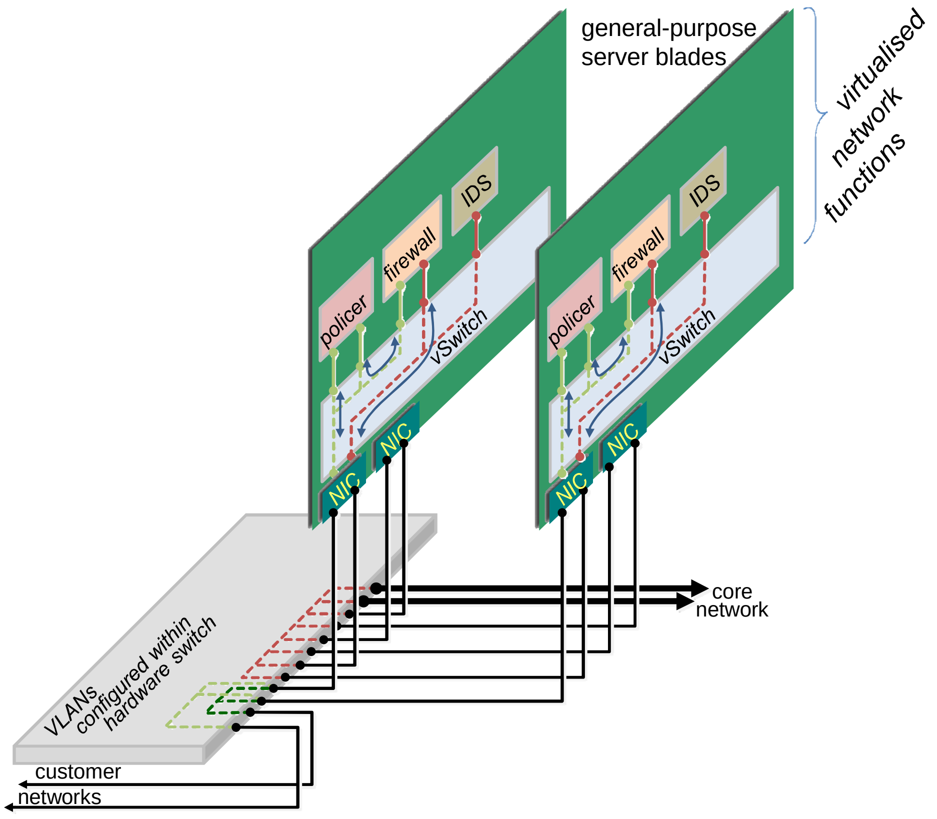
Figure 4 presents part of a network operator's infrastructure that we will use as an example to explain the topology validation problem. Figure 5 shows the same infrastructure, but with a representation of some virtualised Forwarding Functions running on each server. A Forwarding Function is defined in clause 3.1. It can be seen that these particular virtualised Forwarding Functions do not use all the physical interfaces configured in the hardware. Nonetheless, if demand required, the unused network interfaces could be powered up and rapidly connected within the vSwitch.

In Figure 4, the hardware switch has been configured to group subsets of the interfaces into different VLANs, each represented by dashed connecting lines of different colours. The 'no-entry' signs emphasise that the infrastructure has been configured to provide no connectivity between the customer networks, and also no connectivity between the front interface on each server and the rear three interfaces (when viewed as a 3-dimensional picture). Therefore, without any virtualised Forwarding Functions running, each customer network and the core network are all partitioned from each other. This is an example of the requirement for ***disconnectedness*** between many parts of an infrastructure network (see clause 7.3 of ETSI GS NFV INF 001 [i.14]).



NOTE: The 'no entry' signs represent 'default-off' routing between partitioned infrastructure networks.

Figure 4: Example compute and network infrastructure



NOTE: For clarity hypervisors are not shown.

Figure 5: Example virtualised network functions running on the compute infrastructure of Figure 4

Nonetheless, once all the virtualised Forwarding Functions are running (Figure 5) they connect together all the otherwise partitioned networks. However, the connectivity is conditional on meeting the criteria imposed by each virtualised Forwarding Function - in this case each vSwitch is configured to require all communication to pass through a firewall, each firewall will only allow communication if it complies with its own rule-set and if it has not been flagged as a suspected attack by the IDS. Further, even if a communication passes these tests, its traffic rate has to fit within the limits imposed by the policer.

To be concrete, the example infrastructure above uses Ethernet VLANs, but it is not important which virtual network technology is used, and the specific virtualised network functions chosen for the example are also unimportant. Of course, this toy example represents only a small part of a larger network. For simplicity it only shows data plane Forwarding Functions, although in reality, the topology of at least 2 types of network would need to be validated:

- data plane (the network over which the packets that provide the user services pass), including:
  - intra-host paths (between VNF components and/or VNFs on the same hypervisor)
  - inter-host paths (possibly between different hypervisors)
  - paths between hosts and bespoke equipment providing physical network functions
- control and management planes (see ETSI GS NFV 002 [i.12]), including:
  - paths within the management and orchestration (MANO) system
  - paths between the MANO system and the virtualisation infrastructure
  - paths between the MANO system and the hardware infrastructure
  - paths between the MANO system and the VNFs that it manages
  - paths between network controller(s) and the network functions that they control (e.g. in networks that use SDN)

The topology of these two networks might be validated separately. However, they might share some, all or none of their physical and virtual routing, so it will often make sense to validate their topologies together. Indeed, later (in clause 6.2) this toy example data network is extended with a toy example control and management network, some parts of which uses separate physical cabling from the data network, while other parts are only logically separated from the data network, but they share the same physical cabling.

In addition, it is considered good practice to separate storage traffic from other network traffic, at least logically, given the state of the art in network performance isolation is immature (see clause 6.5).

6.1.2 Validating the Topology of Virtualised Network Functions

A network operator will need to be able to validate that the connectivity of its whole network, including all its virtualised functions meets its security policy. For instance, security policy might require that all connectivity between each customer network and the core is monitored by an intrusion detection system and traverse a firewall and a policer. Therefore operators need to be able to validate that the instantiated network satisfies this policy.

It is also necessary to be able to check for any connectivity that should not be present. For instance, it is no use if three connections between a customer network and the core comply with the above security policy but there is also a fourth connection that directly connects the same customer network to the core without passing through a firewall. But even that is not enough - it also needs to be possible to prevent unauthorised connectivity being added, and to prove that it cannot be added by an unauthorised party.

Given the complexity of the topology validation problem in a network containing virtualised Forwarding Functions, it makes sense to divide it down into a number of levels:

1) the physical (cabling) and logical (e.g. VLAN) topology of the underlying infrastructure network would have to be checked;

2) the various ports of each virtual Forwarding Function need to be connected to the correct virtual network (represented by the graph of VLANs within the vSwitch in Figure 5);

3) each virtualised Forwarding Function needs to be internally configured to correctly address output data to the next function in the chain (represented by the double ended arrows within the vSwitch in Figure 5);

4) further it is ideally necessary to be able to validate that a particular virtualised Forwarding Function is indeed performing the required function: e.g. if it is meant to be a firewall, that it is not just forwarding everything without filtering;

5) the per-flow-type topology also needs to be verified. A system will have numerous simultaneous flows with their specific security policy at different layers of the network, including application layer. It may be possible that the devices are verifiably chained at each level, however, the chain for certain types of flows may still not be consistent with the network design (intra-level inconsistency).

NFV allows for arbitrary chaining of Virtualised Network Functions. Loops might be introduced accidentally or maliciously, but even accidental loops might be exploited to amplify attack traffic. Therefore, at each relevant stage above an essential part of topology validation will include detecting and preventing loops. Simple loops might occur where one VNF loops its output back to its input. However, much larger loops might arise, which are harder to prevent. Possible mitigations might be loop detection during topology validation, but might also involve VNF-loop detection at the message forwarding stage.

The infrastructure network (level 1) will usually consist of fairly generic forwarding elements connected in a graph that might be governed by a logically centralised controller, e.g. a software defined network (SDN) controlled via the FORCES or Open Flow protocols; or by a traditional distributed routing protocol, e.g. the spanning tree protocol or OSPF. The two main subclauses of clause 6.1.3, address each of these cases in turn.

In contrast, a network of virtualised Forwarding Functions (levels 2 & 3) consists of fairly arbitrary (non-generic) opaque software modules that process and forward arriving traffic in application-specific ways. In the NFV architecture, ETSI GS NFV 002 [i.12], the orchestrator holds the topology information that connects virtualised Forwarding Functions together (level 2), and the VNF Managers hold the configuration of each virtualised Forwarding Function (level 3). Finally, to validate that each virtualised Forwarding Function is indeed functioning as expected (level 4) would strictly require source code attestation (see clause 6.3). However, at minimum, it would be useful to check that the module running has the right name and was installed and run by properly authorised identities.

For systems of non-trivial scale, it will be infeasible to do all these topology checks manually. It is believed that there are no products or standards covering such system-wide topology validation, so topology validation could be a genuine gap in the market and in standards coverage.

6.1.3 Validating the Topology of the Infrastructure Network

6.1.3.1 SDN-specific issues

NFV deployments could use Software-Defined Networks (SDN), either for the management and control planes or for the data plane. Although any security problems with SDN are not specific to NFV, they are outlined here because SDN is considered highly complementary to NFV in certain scenarios (e.g. data centres).

There are various - and conflicting - definitions of SDN, but core aspects of a software-defined network are a combination of one or more of:

- Data plane and control traffic is typically separated - though this separation might be virtual, not physical.
- Control of routing tables in physical and/or virtual switches.
- The ability to change routing tables dynamically.
- The remote control of routing tables: this might be through:
  - a single, logically centralised entity (a single "SDN controller")
  - a set of federated entities
  - a hierarchy of entities

Also, a software defined network will invariably exhibit the following characteristics, although they are not defining characteristics of SDN:

- Abstraction of routing topologies from the physical components of the network to a virtual network
- Tunnels between various nodes on a network, which may or may not be encrypted
- Overlays, such as L2 over L3, that stretch across multiple networks, terminating at nodes or at gateways between physical networks, using e.g. VXLAN [i.27], STT [i.28], NV-GRE [i.29], MPLS [i.30], IEEE 802.1ah [i.25] (MAC in MAC), IEEE 802.1ad [i.26] (Q in Q), etc.

SDN is an emerging technology, and is therefore immature: even simple command and control systems, including topology validation, are currently (2014) few and far between. Since software-defined networks can be defined on-the-fly, validating topologies and/or constraining the SDN topologies to "validated" instances becomes significantly more complex than before and a real challenge (particularly in terms of security).

It seems vital that some techniques are employed for simplification of this complexity. One such approach is that of security zones, where a set of network components are contained within a "zone" (whether virtual or physical) where they are all subject to the same security policies. This might be for a variety of reasons, including:

- legislative or jurisdictional control;
- customer-type (government vs. enterprise vs. residential, for instance);
- content/traffic (storage and transfer of content such as video or audio requiring rights-protection);
- "multi-tenant" controls, where components from different entities (e.g. competing network operators) are hosted on networks that are physically or virtually connected.

6.1.3.1.1 Hierarchies of control

Almost all SDN topologies are likely to contain at least some level of hierarchy, with the lowest cardinality being a single controller (or controller layer) controlling switches (virtual or physical) via a control network. Typically, these switches will have a rule-set pushed to them by the controller (either once, or with updates and/or refreshes as required), and they are then able to make decisions about how to route packets (including applying "catch-all" default drop rules, for example). However, the opportunity exists in at least some SDN schemes (such as OpenFlow-based networks) for the lower-layer components to request routing decisions from high-layer components.

It is important, of course, for there to be mutual authentication between the controller and switching/routing entities. Should the controller be spoofed, the switching fabric is at risk of being taken over and misused. On the other hand, should the switches be spoofed, there are equally concerning issues:

- the intended topology of the virtual network might be revealed to an attack, yielding useful mapping and attack data;
- the controller, which should act as a trusted holder of knowledge of the state of the network, ceases to hold this role.

In terms of where forwarding decisions are made, there is a trade-off between performance and security. The theoretically "most secure" option would be to allow the controller entities to make decisions about every single packet passed to a switching or routing entity, so that the controller would be able to correlate signs of attacks from multiple elements and be in a position to immediately block attacks. However this would clearly be impractical in most cases (hence the use of rule sets). However, a situation where a switch can *never* pass decisions up the layers to controlling entities is also restrictive, and allows for possible security issues to be overlooked. There is a spectrum of options:

- All decisions made by controller: significant performance impact.
- Subset of decisions made by controller: likely based on exceptions to rules, or possibly on time degradation of rule sets.
- All decisions made by switch: less information about unexpected traffic available to controller, less ability to change topology to react to change.

To soften this dilemma, switches can be programmed to send regular traffic monitoring updates to the controller. Then, at least the controller can correlate attack information from multiple sources. Then, even though each switch is making frame-by-frame decisions, the controller can still intervene against attacks over longer timescales.

If a switch loses contact with its controller, it needs to decide whether it should fail-open or fail-safe, and how long it should wait for a reply before applying such a rule. It seems important, therefore, to ensure that the control plane is kept live and the controller responsive in order to avoid such rules having to be applied. It is likely that administrators would wish to be able to tune and configure the parameters that control such behaviour in order to understand and mitigate particular failures and attacks, but the difference between the two may be difficult to identify in real time.

An additional complication is that when some or all decisions are made by the controller, it becomes a possible target for a Denial of Service (DoS) attack, leaving the wider SDN infrastructure vulnerable given components rely on the controller to make decisions.

6.1.3.1.2 Additional layers of hierarchy

There are at least two additional layers of hierarchy which impact on SDN topology validation:

1) **A lowest layer:** Operators might log in to equipment - whether hardware or software - to make changes to the behaviour of the router/switch independently of the controller (either unaware of correct procedure, or to manually override an inaccessible controller, or perhaps to mount an insider attack). The consequent inconsistency between the switch and its controller might be mitigated by a combination of physical and procedural controls and required "report-back" functionality of the router/switch which would inform the controller of changes.

2) **A higher layer:** Virtualised Forwarding Function can change the routing of packets, in application-specific ways. Certain virtualised Forwarding Functions will take their instructions from the SDN control hierarchy (e.g. via OpenFlow™ interfaces). However, one of the purposes of NFV is 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™. Therefore, an SDN hierarchy always needs to allow the possibility of being subordinate to the orchestration system of a network comprising both (virtualised) Forwarding Functions and SDN controlled elements (see clause 6.1.2).

6.1.3.1.3 Further issues

For a comprehensive discussion of SDN-specific security issues, see [i.18], [i.17], [i.24] and [i.19].

6.1.3.2 Issues specific to distributed (non-SDN) routing

In many circumstances, the infrastructure network might use traditional distributed routing algorithms to build the routing and forwarding tables at each node, e.g. open shortest path first (OSPF), the Intermediate System to Intermediate System (IS-IS) protocol or the spanning tree protocol widely used for Ethernet. This offers the benefits of NFV such as more flexible deployment and execution with initial deployment simplicity, in that the infrastructure network can be unchanged and VNFs can replace existing network functions on a like-for-like basis.

In such deployments, the topology validation problem is largely no different from that with non-virtualised network functions. Indeed, distributed routing deployments might well consist of a mix of virtualised and non-virtualised implementations, given incremental deployment is one of the main benefits of NFV.

As in the SDN discussion above, routing graphs are often built hierarchically and, as with traditional non-virtualised implementations, a higher layer in the hierarchy depends on the existence of a working lower layer to carry routing messages between nodes.

The main difference is that initially a non-virtualised implementation has to be physically installed, whereas a virtualised implementation depends on the physical existence of compute infrastructure (servers, storage, etc.), and sufficient network infrastructure over which to deploy the executable image. The security implications of this difference are covered in clause 6.2.

6.1.3.3 Related issues

Topology validation and enforcement allows operators to address several possible families of attacks on the NFV. Full knowledge of vulnerabilities may not lead to the ability - or will - to address these attacks with 100 % effectiveness, but mitigation or other measures may be appropriate. Attack families include:

1) sniffing of packets: an invasion of privacy attack;

2) various man in the middle attacks: typically for replay attacks, though real-time substitution might be an option in certain cases;

3) spoofing attacks;

4) denial of service attacks;

5) Fuzzing Attacks on the network: Attackers trying various packets to see if some can circumvent the intended chain of elements.

Although correct topology knowledge is certainly a pre-requisite for solutions to some of the attacks above, the above problems do not seem to be altered by NFV.

6.1.4 Topology Policy Validation & Enforcement

Before validating that the instantiated topology matches the topology design, it ought to also be possible to check (offline) that the design itself satisfies the desired security goals. For instance, a policy might require that there is no path between networks N and M that does not pass through at least one firewall and exactly one NAT, in that order.

It is not enough to check that the design contains no out-of-policy connectivity; it also needs to be possible to know that the design would prevent it being added.

Nonetheless, given policy validation would be executed independent of the run-time environment (either offline, or within an SDN controller hierarchy), it is unlikely NFV will introduce any new concerns at this stage.

6.2 Availability of Management Support Infrastructure

Irrespective of whether NFV is used, out-of-band management infrastructure (aka. lights-out management or LOM) is required to remotely manage any large compute or network infrastructure deployment. This ensures that management access is still available even when the compute or network infrastructure has crashed. Otherwise the management system offers no way to remotely re-start failed services, including repairing a corrupt operating system or loading certification keys, operating system licence keys and other essential configuration.

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 same is true for the communication channels to log routine data and problems to management systems, and for remote administrators to manually diagnose and fix problems, again irrespective of whether NFV is used. Otherwise configuration errors could lock out a remote administrator, and alarms notifying crashes might not be conveyed to the management system.

However, although a physically separate management network would be ideal, its cost can be prohibitive. Instead, vendors nearly always provide administrator access as well as connectivity to management systems via a network port or ports (even if the operating system has crashed). Therefore, it is important to at least secure these port(s) by connecting them to a management virtual network that is inaccessible from customer networks. To improve availability, path diversity is also advisable, for instance, it may be feasible to use cellular connectivity as a back-up. However, there will still be scenarios where it is only economic for management connectivity to be logically not physically separate (e.g. where virtualised network functions are running on remote customer premises and being managed remotely). In such cases, it may not be possible to commit to the normal levels of availability. Access controls will also need to be more stringent if the management infrastructure is only logically separated.

All the above trade-offs between availability and security are applicable irrespective of whether NFV is used or not. However, NFV brings additional challenges. As a hypervisor starts up, it may need network access to read-in (parts of) its own configuration, including software licences and security keys. Clearly this will be problematic if the hypervisor's network connectivity is dependent on a network function that runs on top of the hypervisor (Figure 6), or on top of another hypervisor that may be in the process of re-booting itself, perhaps following a power failure. This problem could be solved by allowing the hypervisor to connect over the same physically separate management network provided to manage the hardware.

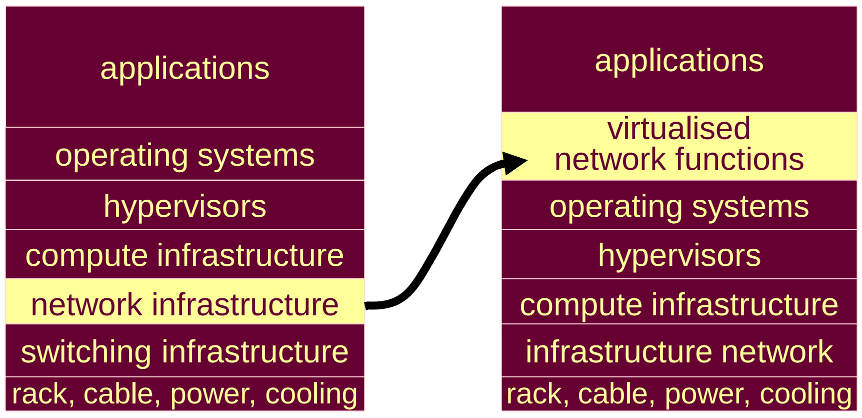


Figure 6: Re-arrangement of the order of dependency between traditional application virtualisation (left) and network functions virtualisation (right)

However, although useful, such an approach starts to erode the security of having a physically separate management network.

The same problem can arise when a virtualised Forwarding Function starts up; needing network access for its own configuration that may depend on connectivity through a second virtualised Forwarding Function, which itself may depend circularly for its configuration on connectivity via the first virtualised Forwarding Function. It would even less secure (though useful) to start giving virtualised processes access to the physically separate hardware management network.

Figure 7 shows a potential solution to these dilemmas between availability and security (using the Topology Validation Example introduced in clause 6.1.1). On the core side of the switch, cabling that is separate from the customer data networks is used. It is not so important whether the management network is physically or logically separated from the data network. More important is that the management network systematically avoids passing through any virtualised Forwarding Functions - it needs to be connected to them, but not through them. Admittedly, within each server the management network passes through a vSwitch. Nonetheless, if the vSwitch process fails, at least the independent management network connects directly to it, so it is likely to be available to fix the vSwitch.

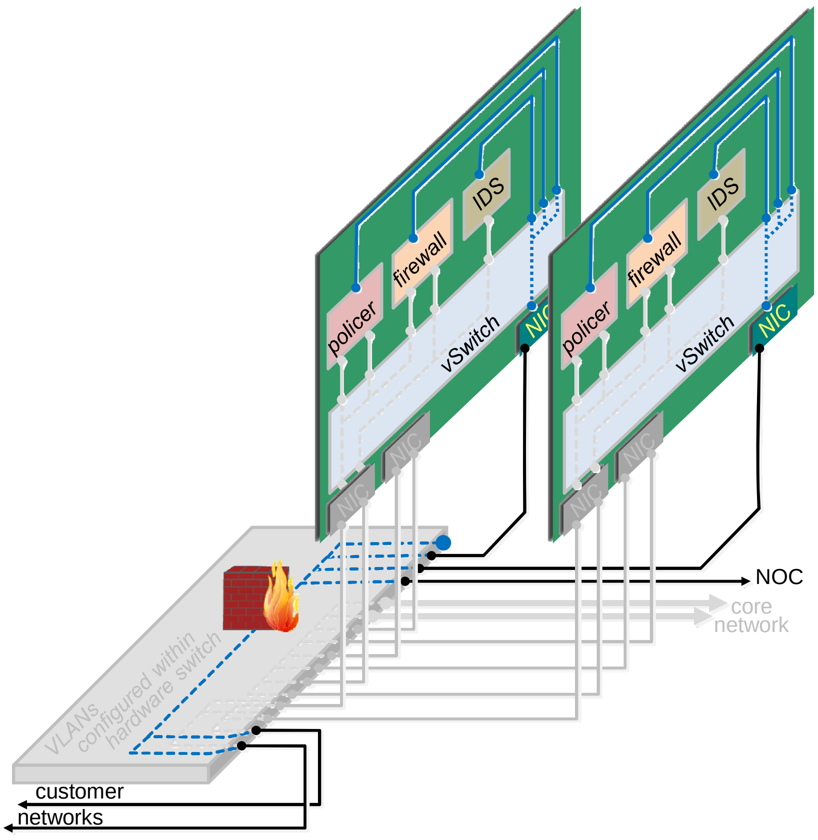


Figure 7: Example management network,
illustrating a mix of physical separation (core side) and logical separation (customer side and VNFs)

On the customer network side of the switch in Figure 7, it can be seen that the management network is only logically separated from the customer's data network, sharing the same physical cabling. As already explained, this might be a valid compromise given that the cost of a separate physical access link to manage every customer site would be prohibitive. It will be noted, that a firewall is introduced to prevent access to the interior management network from the customer side, while still allowing connections to be initiated from the network operations centre (NOC) to manage customer networks. It will also be noticed that a simple firewall within the hardware switch is used, in order to keep to the rule that the management network ought to avoid passing through virtualised Forwarding Functions.

6.3 Secured Boot

6.3.1 Background

In all the hosted deployment scenarios in clause 5.1, particularly in those on customer premises, the network operator has to trust its hosting provider's virtualisation platform sufficiently to run VNFs on it, and the platform will similarly want to be able to check that the VNFs are genuine. This is the secured boot problem.

Confusingly, the term 'secure boot' has been used as a name for a specific set of industry specifications as well as for this general area. The present document will use 'Secure Boot' solely for the specific technology and 'secured boot' for the general term. An NFV reference to a secured boot process appropriately includes Secure Boot technology, without exclusively prescribing this technology

Secured boot encompasses the technologies and methods for validation and assurance of boot integrity. Boot integrity validation can invoke numerous assurance factors, including local attestation, remote attestation, attribution, authenticity, configuration management, certificates, cryptographic keys, digital signatures and hardware-specific features. The secured boot process is inclusive of hardware (specialist acceleration hardware as well as general-purpose processing), firmware, hypervisor and OS image validation and assurance factors, including that of the associated security credentials. The desired level of assurance is achieved through the selection of appropriate factors.

A related area of concern is validation of VNF software during both initial installation and subsequent launching of a VNF. Such validation provides assurance that the code loaded into the VNF execution environment is authentic and has not been tampered with in the duration between when it was authenticated and loaded. While validation at the installation stage may be addressed by the VNF manager, validation at the launching stage seems a natural extension of the trusted boot process. To this end, the established trust base could be used to measure VNFs and ascertain that they are known and good. This, however, is still an area of research and development.

Under the circumstances, software-only validation (albeit without hardware-rooted trust) is a useful alternative. Such validation involves checking the digital signatures of VNF modules from different vendors. These signatures need to be backed by well-recognized root certificates. In an NFV environment, there may be a need for hardware to incorporate software from multiple independent software vendors. To minimize certificate management complexity in such cases, it may be desirable to have a single certification authority for VNFs.

6.3.2 Secure Boot and Trusted Boot Technology

Secure Boot has been addressed-and is being resolved-in several fora. The architecture and mechanisms for verifying signed firmware and software images are specified by the Unified Extensible Firmware Interface (UEFI) Forum [i.3], a non-profit industry organisation that involves primarily chip and computer manufacturers.

Secure Boot technology can enable protection against root kits and reset attacks, and detection of unauthorized changes to BIOS (and its configuration), the boot sequence, and the hypervisor or OS (and its configuration). As a result, it is possible to ascertain that a host is booted into a good, known configuration based on hardware-rooted trust. Although supported on servers, the technology is not yet in use widely. Given the relevance of hardware-rooted security to NFV, it is useful to understand the challenges in deploying secure boot technology.

UEFI Secure Boot with the UEFI architecture involves checking the signatures (i.e. signed cryptographic digests) or hash of all UEFI modules as they are loaded. If the signature check fails, the boot stops. (The public keys for verifying signatures are stored in three databases-the *signature database*, *revoked signatures database*, and the *Key Enrolment Key database*-all in non-volatile memory. They are locked for editing before being deployed). There is also an option to approve boot signature manually, via the console, under the control of the administrator.

There is a more general technology to achieve the same purpose (and actually much more)-called *trusted computing*. It is developed by the organisation appropriately called the Trusted Computing Group (TCG [i.2]). The overarching concept is the *chain of trust* to connect all pieces of firmware and software with the hardware. Starting from the *endorsement* key pair, the private key of which is stored within a tamper-resistant hardware chip, 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).

One key specification is that of the trusted platform module (TPM) [i.8], of which a recent version has been also published as the ISO standard ISO/IEC 11889-1 [i.4]. Among other things, TPM provides capabilities to generate-within the chip-cryptographic keys (bound to the endorsement key) and to store the measurements (effectively hashes) of respective boot components.

TPM has been implemented in hardware, and there has been an effort to virtualise it. The latter culminated in a Trusted Computing Group's 2011 specification [i.9], which defined the relevant architecture and deployment models. Virtualisation of TPM allows preservation of the trusted properties of the software running on a physical or virtual machine.

Booting with TPM (sometimes called *trusted* boot to differentiate it from the UEFI Secure Boot) can be performed to assure the security of the booting process as well as Secure Boot does. This also offers better granularity for the boot process, while bringing additional complexity. In some cases, UEFI Secure Boot can co-exist with TPM-based trusted boot.

The National Institute for Standards and Technology (NIST) has also specified various related guidelines [i.6], [i.10] and [i.7].

6.3.3 Secured Boot Summary

The above secure boot and trusted computing technology addresses the secure boot problem, at least in theory. The outstanding question is whether this technology has proved to be feasible to operate at scale, for instance: whether management systems are sufficiently available to manage deployment of replacement keys throughout the infrastructure, including remote customer sites, perhaps because they have expired, or in response to the day-to-day reorganisation of contractual relationships. In particular, it is necessary to know whether procedures for key revocation scale to carrier-size networks. It is also necessary to consider whether key management processes are robust to large-scale power failures, given these technologies may have only been used at scale in data centre environments, and it is harder to provide back-up power across all the distributed sites that comprise a network than it is for a few centralised data centres.

6.4 Secure crash

Crashes of components of any system are possible, and can cause security problems. Loss of data and state can generally be mitigated with fail-over and back-up provision, but data which is not securely cleared might give attackers unauthorised access to information or capabilities. Within the NFV framework, the key components that might be of risk in this context are:

- VNF component instances (VMs running on a hypervisor):
  - Hypervisors need to ensure that in the event of the crash of a VNF component instance, all file references, hardware pass-through devices and memory are safe from access by unauthorised entities. This is expected behaviour for most hypervisors.
  - If the application running within the VM crashes, but not the VM itself, the hypervisor needs to ensure that no changes to the existing authorisations are made. Note that the hypervisor might be unaware that the application within the VM has crashed.
- Local network resources attached to VNF component instances:
  - This is covered in the point above, as hypervisors are expected to enforce the clean removal of 'local dangling references'.
- References on remote devices to local VNF component instances ('remote dangling references'):
  - Following the crash of a VNF component instance (or of infrastructure it depends on), remote devices might continue to hold references to the interfaces of the crashed process (e.g. MAC addresses, VLAN IDs, virtual network identifiers, IP addresses). Such identifiers are often used to isolate a session from outsiders. Even though such techniques are not secure in the cryptographic sense, they are widely used for convenience and efficiency.
  - The hypervisor needs to ensure that it is not possible for a newly executing VNF component instance to adopt addresses that were recently used by a crashed instance. Otherwise the new instance may be able to place itself in a position where it can adopt the privileges associated with a recently crashed instance.
  - Usually hypervisors or other functions ensure that a new VNF component instance cannot be allocated an address associated with some privilege (e.g. joining a virtual network) without satisfying the same authorisation criteria as were originally needed by the instance that crashed. However, hypervisors ought to also protect against vulnerabilities where an address could be used following a crash despite not being allocated (i.e. vulnerabilities due to a *combination* of a crash *and* spoofing or masquerading).
- Local storage resources attached to VNF component instances:
  - The hypervisor cannot know what storage resources need to be wiped in the event of a crash, as some may be required by new VNF component instances to be instantiated, or may be shared between VNF component instances. In the event of a crash, arrangements need to be made for the hypervisor to wipe local storage that is no longer required (e.g. the VNF Manager might instruct the Virtualisation Infrastructure Manager to request this).

- Remote storage attached to VNF component instances:
  - As in the case of local storage, the hypervisor cannot know which remote storage resources need to be wiped in the event of a crash, as some may be required by new VNF component instances to be instantiated, or may be shared between VNF component instances. Further, the hypervisor may not itself have access to the remote storage. In the event of a crash, arrangements need to be made for the relevant NFV instance to wipe the remote storage (e.g. the VNF Manager might instruct the Virtualisation Infrastructure Manager to request this).
- Swap storage attached to VNF component instances:
  - If the VNF component instance is using swap storage, it needs to be marked as such and the hypervisor ought to wipe it in the event of a crash.

There are other types of security which can be affected by a crash, most notably the availability of a service. Should a VNF component instance crash, *or the routing between it and another entity fail*, then availability will be affected. In this case, the VNF Manager needs to identify the likely cause of the problem and work with the NFV Infrastructure (via the Virtualisation Infrastructure Manager) to work around it. This might require the creation of a new VNF component instance (or set of instances), the re-routing of packets between entities or the creation of new routes between entities. It is also likely to require re-configuration of VNF component instances and possibly the re-establishment of trust relationships between VNF component instances or between VFNCIs and other instances. In some cases, a crash may have fatal impact on a particular VNF instance - in others, availability might be re-established through the measures identified above.

It is clear from the notes above that in several cases, the VNF Manager will contact other entities to act with regards to resources. The information required to identify such actions needs to be derived from two sets of data:

1) VNF component deployment and configuration policy;
2) VNF component instance run-time state.

In order for a VNF Manager to identify the operations required to clean up securely from a crash, information needs to be available not just about state immediately before and after the crash, but also about whether a particular component requires clean-up operations, what operations are needed, and how timely they need to be. Such information needs to be provided by the VNF Descriptor, as it comprises an important set of information for operational decisions by the VNF Manager.

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. However, NFV significantly alters the impact of a potential attack.

6.5 Performance isolation

A range of isolation approaches are available:

- static hardware segregation, e.g. physically separate management infrastructure as in clause 6.2;
- resource reservation allowing guest processes to request dedicated resources, e.g. allocation of complete processing cores or defined ranges of memory to different tenants;
- quotas or payment, either limiting the amount of resources that a guest can reserve, or limiting the amount a guest uses, e.g. pay-by-use processing;
- resource isolation by division, e.g. dividing the available resources between the competing demands, e.g. round-robin processor scheduling or fair queue bandwidth scheduling. Weighted variants are common.

One of the main purposes of virtualisation technology is to isolate one VM from crashes, hangs, loops, etc. of another VM, using techniques like these. However, all these techniques can be highly inefficient in two cases:

- when the granularity at which the resource can be allocated is too coarse (e.g. many small cores can be more efficiently partitioned than fewer more powerful cores);
- when each workload's use of a resource is highly variable.

In practice, this means that pragmatic solutions are common for isolating processing cores and main memory, but isolation of other resources can be more problematic.

6.5.1 Network and I/O partitioning

Network workload 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.

In order to at least localise the problem, a network's topology is often arranged so that the capacity at a server's interface will tend to be the bottleneck. The interior of a network can be made strictly non-blocking using sufficient capacity and multipath routing. However, often competitive pressure drives providers to save on core capacity and rely on the low probability of traffic all focusing on one core link. Any partitioning between customers on high capacity core links would require too much per-packet processing and/or too much per-guest state and churn, therefore they still rely solely on edge techniques.

Algorithms like weighted round-robin are sometimes used to divide the edge bandwidth resource only between active workloads. However, these algorithms hold no memory of how often each guest has been active over time, so intermittent workloads lose out considerably to persistent ones. In other words, by design, these algorithms favour flooding attacks, even in the outgoing direction.

Isolating usage of local network resources is hard for another reason. Large numbers of remote users can send incoming traffic that makes demands on the local resources, both bandwidth and interrupt processing. This problem has as much to do with preventing identifier duplication/spoofing as with scheduling technology, but even without any malicious intentions, isolation is hard for incoming workloads.

Network QoS (quality of service) differentiation can be used to ensure more critical tasks are less likely to contend for network resources. However, QoS adds extra dimensions of complexity for network function developers and network operators and it deliberately does not address the performance isolation problem for the bulk of regular traffic.

Therefore isolation of bandwidth usage is not a fully solved area, for all the above reasons (the highly distributed resource, high demand variability and no control over incoming remote sources). Problems of bandwidth performance isolation already existed in the networking and cloud industries before NFV, which does not seem to introduce any new threats. However, it is self-evident that the network resource is the critical one for network functions, so the lack of good solutions to these problems becomes critical for NFV.

6.5.2 Shared core partitioning

If a particular technology relies on fewer more powerful cores relative to another that may use more less powerful cores, there will be economic pressure to share the more powerful cores between guests. It has been shown to be possible for an attacker to exploit this lack of complete isolation to extract side-information about a victim process, but the low signal to noise ratio makes it very hard to extract useful information, e.g. learning a password by measuring key-stroke timings. Sufficient side-information leading to extraction of a victim's cryptographic keys seems a somewhat distant possibility [i.22], but it ought not to be discounted.

6.5.3 Acceleration hardware partitioning

NFV is more likely to use hardware acceleration where available, e.g. accelerators for functions like encryption or packet filtering (pattern matching). Technology is being developed to enable virtualised applications to use such hardware where available and to otherwise execute the equivalent functions on a general purpose processor. This is sometimes possible via an API that allows the application to be aware of the specialist hardware. Alternatively the application has to specifically test if the special hardware is present and call it explicitly through a modified API (paravirtualisation). There may be new vulnerabilities where these specialised resources are less easy to partition than general purpose resources like CPU, memory & storage.

6.5.4 Shared memory partitioning

To achieve high throughput, it is tempting for hypervisor developers to allow direct pass-through, circumventing isolation between virtual machines. Modern microprocessors are being enhanced with a hardware I/O memory management unit (IOMMU), so that a VM can programme network interfaces to use DMA (direct memory access) when accessing the VM's own memory, but not that of other VMs. This extends the virtualisation support of the traditional MMU to include DMA for the high performance I/O needed by network interfaces such as Ethernet or Fibre Channel. With Intel processors IOMMU is called Intel Virtualisation Technology for Directed I/O (Intel VT-d); ARM manufacturers can add their own IOMMU to processors that already support virtualisation, such as the Cortex A15, or optionally they can license ARM's System MMU (SMMU) technology.

A number of serious vulnerabilities have been identified in IOMMU technology, such that one VM can access another's memory; prevent another VM from executing; install a malicious hypervisor below the system hypervisor; or take down the entire computer (see clause 2.7 on I/O virtualisation in [i.16]).

These problems are being addressed by building support for partitioning between VMs ('shared IOMMU') into each item of I/O hardware and its supporting chipset. In theory shared IOMMU can resolve the dilemma between performance and security. However, security then depends on the implementation of each card, not on system software, so the security of each hardware implementation has to be assessed. Standards help to simplify this task, the best known of which is SR-IOV (standing for single root I/O virtualisation), from the PCI special interest group, which is for PCI-express cards coupled with Intel Virtualisation Technology for Connectivity (Intel VT-c).

6.5.5 Attacks on the resources of the virtualisation infrastructure

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. All hypervisor architectures consume a low level of resources as part of their normal operation, which is unlikely to impact the operation of VMs. However, if an attacker - whether an external entity or one of the hosted VMs (a "noisy neighbour") - can find a way to force the hypervisor to consume too many resources or fail in other ways, the operation of the entire system is likely to be affected.

Standard mitigation techniques to deal with noisy neighbours typically involve QoS controls where the hypervisor specifies maximum flow rates on storage and/or storage interfaces - see clause 6.5.1.

Some of the possible resources that could be consumed, causing hypervisor platform performance problems and associated NFV impact, include:

- local storage attacks (e.g. contriving to make the hypervisor fill up local storage with logs)
- remote connection attacks (e.g. remote log storage, remote control channel degradation)
- memory pressure (e.g. consumption of kernel memory)
- CPU attacks (e.g. attacks that cause scheduler unfairness)
- OS resource attacks (e.g. consumption of file handles, event channels, etc.)

There are some techniques (see clause 6.5.4) which can help mitigate against such attacks, but if a host is brought to its knees by such an attack or failure, performance isolation cannot be guaranteed. In this case, independent proactive monitoring is a key technique to detect that the virtualisation infrastructure and consequently its associated VMs and VNFs are floundering. Monitoring could watch for anomalous traffic behaviour and/or for degraded performance of network, I/O, processing, memory and storage. Monitoring will need to be in place at both the infrastructure level and the level of guest VNFs. These two layers will require interfaces with the management & orchestration system to consume monitoring information, then act on it accordingly.

6.6 User/Tenant Authentication, Authorization and Accounting

The introduction of NFV brings new security issues when it comes to AAA, as it implies using identity and accounting facilities at two or more layers: the network and virtualisation infrastructure (e.g. identifying the tenant or guest service providers) and the network function (e.g. identifying the end-users). What is more: the E2E Use Cases document ETSI GS NFV 001 [i.13] describes architectural patterns (NFVIaaS, VNFaaS, VNPaaS, VNF Forwarding Graphs, etc.) that suggest the stacking of identities can occur at more layers if all patterns become commonplace, as shown in Figure 8.

While the figure only shows so-called infrastructure (vertical) integration, collaborative (horizontal) integration is also possible, for example: federation of different NFV infrastructure providers, or traversal of several NSPs to provide VNPaaS. Issues related to identity will need to be solved for potentially any kind of horizontal and vertical integration patterns.

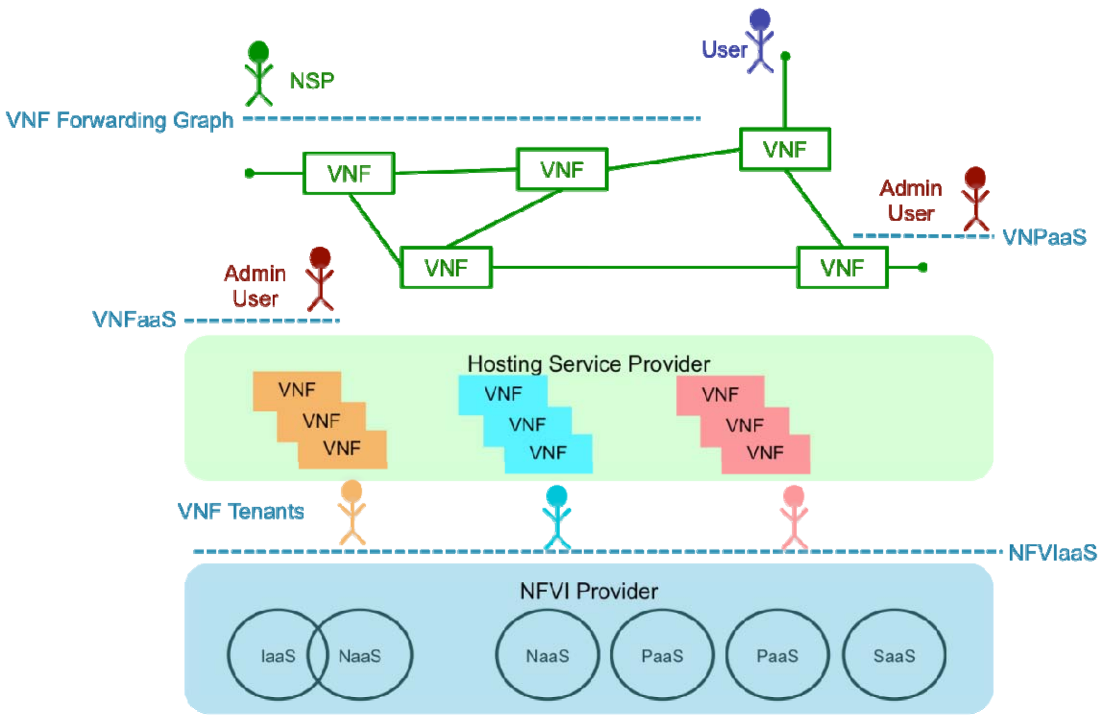


Figure 8: Example of multi-layered identities

A generalized AAA schema for identifying users utilizing a particular tenant infrastructure and/or a tenant acting on behalf a user is required to support these patterns and the new operational an business models they will bring. Current AAA mechanisms assume there will be a single identity, single policy decision and enforcement points, a single level of policy, and a single accounting infrastructure. Even if a strict separation by tunnelling (as current practice seems to suggest) would be feasible without breaking some of the promised NFV enhancements in what relates to scalability, agility and resilience, there exist risks related to each one the three components in AAA:

1) Authentication procedures can imply privacy breaches associated to the disclosure of user information at layers that are not intended to consume certain identity attributes.

2) Authorization risks are mostly related to privilege escalation produced by wrapping unrelated identities not verifiable at a given layer. A well-known example in Telco infrastructures is the multi-frequency phone hacking of not that many years ago.

3) Accounting needs to be performed at all the underlying infrastructure layer(s), and they will not only require accounting at the granularity of virtualised applications that use the infrastructures, but also at the coarser granularity of the tenants running virtualised network functions. In summary, the capability of billing the billers.

To address privacy issues in authentication, mechanisms for the generation and validation of identity tokens, as well as for the exchange of identity attributes need to be validated in a multi-layered multi-tenant environment. Directly connected to this attribute exchange, appropriate authorization processes will require the availability of methods for policy routing (or migration) and composition, applicable not only to users and tenants, but to VNF deployment units as well. These requirements claims for a consistent composition of current cloud AuthN/AuthZ techniques (like Keystone) and network practices (mostly based on RADIUS and DIAMETER), probably through proposals intended to go beyond the state of the art in identity federation.

For accounting, apart from the existing cloud mechanisms on resource usage (processing, storage, memory, local network capacity, licences, etc.) an infrastructural network function will be needed that consists of the following functional blocks (FBs):

- Traffic packet acquisition at full-line rate.
- Traffic classification and accounting per service provider, per user and per application.
- Policy decision function, intended to raise an alarm whenever a specific threshold has been achieved according to the detected traffic and the policies.
- Policy enforcement, intended to enforce the rule in the data plane.

While traffic acquisition FBs typically have to be distributed, traffic classification/interpretation FBs need to be consolidated from different nodes into a central location, to achieve gains in scale and operational simplification.

All this requires the support of a trust framework that the experience has shown cannot be derived hierarchically from a single global root, while it does not seem realistic to expect that pure peer-to-peer agreements will be able to support the highly dynamic scenarios NFV will bring. It is crucial to find an appropriate combination of flexibility, manageability and security properties for the trust framework supporting the AAA infrastructural services.

It is still to be established whether NFV introduces any need for integrated support for identities between any VNF and its infrastructure layer. Also, if it does, whether Cloud AAA already provides sufficiently rich support for multi-domain, multi-layer identities. If not, NFV introduces no new AAA problem.

6.7 Authenticated Time Service

The security of an instance of a VNF component depends on the integrity of messages that synchronise system clocks and messages that query clocks. The integrity of timing messages depends not only on the integrity of the information carried within the message, but also the integrity of the time taken to deliver the messages themselves. For instance, an attacker with control over message forwarding could exploit the assumption that a half-round-trip-time correction has to be applied to a timing message by delaying the message more in one direction than the other.

Given such problems, clients that require secure time typically query a diverse set of time services and reject outliers. In a virtualised environment, even if a VNF sends out queries through multiple interfaces, the hypervisor represents a singular point of control and there is no other way to communicate between the VNF and any network interface. Therefore, a compromised hypervisor might be able to contrive to tamper with all timing requests in a co-ordinated way in order to shift a VNF's conception of true time by rendering its majority voting defence ineffective. Of course, if an attacker can compromise a hypervisor, many other simpler attacks become possible as well. However, this does not negate the need to at least consider the possibility of attacks on a VNF's conception of time.

NFV shares this requirement with general application virtualisation facilities (cloud). However, the problem is perhaps more acute in the more physically distributed scenarios envisaged for NFV. Nonetheless, if the cloud industry converges on a sufficiently suitable solution, it is recommended that NFV simply uses it. Otherwise, the NFV community will have to take the lead and place this requirement onto the agenda of a suitable standards organisation, in order to converge on a single standard solution.

Another subtle potential attack that could be introduced by NFV is where an attacker detects the time skew introduced by the hypervisor relative to a hardware clock. Although such a technique has not yet been shown to be useful for mounting a specific attack, similar techniques have been used for an attacker to detect the work being done by another process sharing the same core (see clause 6.5.2).

Before any action is initiated to close off these potential vulnerabilities, further research is needed to demonstrate that the ideas for timing attacks proposed here could be feasible.

6.8 Private Keys within Cloned Images

Images for running on virtual machines are expected to use asymmetric key pairs, for example RSA or ECDSA (elliptic curve digital signature algorithm) for authentication and securing communications. This means that somewhere a private key is held as part of that key pair. This raises the following questions:

- How are these private keys protected and distributed?

  Assume images serve as blueprints from which new VMs are deployed. Among other things, an image typically captures the OS, applications, and their configuration settings. An image ought not contain any sensitive data (such as private keys and passwords). Instead, such data could be injected or provisioned when a cloned image is first booted. Ideally, private keys, root passwords, and the like are unique for each cloned image. It obviously matters who has control of these sensitive data. In the case of hosted network operator deployment, for example, the VNF rather than NFV Infrastructure administrator ought to have the control.

  If an image cannot be free of sensitive information, it needs to have confidentiality protection in addition to customary integrity protection and access control. In this case, secure key management is also necessary.

- Does each copy of a cloned image use the same key pairs or unique key pairs?

  The answer is use-case dependent. If the cloned image is for failure recovery, it might use the same key pair of the failed VM, for example for TLS (or IPsec) authentication. If the cloned image is for handling excessive load (as in auto scaling), it might use a new key pair. In each scenario, the entity communicating with an image ought to be able to reliably ascertain the authorized use of the secure credential by both that image and its cloned images.

- If one image is compromised, are they all?

  For example, assume that RSA key pairs are used for transport layer security (TLS) protected connections. The public portion of the key is in an X.509 certificate. If each clone image uses the same key pair, then if one is compromised, they all are.

Each clone image ought to have a unique key pair, therefore operational procedures are needed to create, sign and track each unique X.509 certificate. Without easy-to-follow procedures, operational practices will tend towards the lazy but insecure approach of cloning images with their keys.

This issue is common to both NFV and cloud more generally. However, this issue is raised here because insecure embedding of private keys is widespread in cloud images, and we wish to ensure that NFV deployments do not inherit these bad practices.

For NFV, it is useful to employ operator-controlled certification authorities (CAs) in addition to well-known ones. CAs controlled by an operator (i.e. with the operator's self-signed root certificate) are suitable for supporting services that are internal and do not involve interactions with entities outside of its NFV infrastructure. Examples of such services are those dealing with management, orchestration, and operations. The use of operator-controlled CAs for internal services simplifies certificate management and the related operations. It also reduces the threat surface because of a shorter chain of trust. Regardless of the types of CAs, it is essential to follow best practices for creating, signing, verifying, and tracking certificates, and for safeguarding private keys.

6.9 Back-Doors via Virtualised Test & Monitoring Functions

In virtual network functions, the functions will be provisioned, configured, tested, and debugged using software interfaces, perhaps remotely. These interfaces have very different security characteristics and requirements than traditional physical interfaces. This clause seeks to highlight the risk of debug and test interfaces in NFV devices and provide input to development of mitigations and protections in circumstances where these interfaces are required to be enabled in the field.

6.9.1 Operational Test, Monitoring and Fault Tracing

Interfaces provided for test, monitoring and diagnostics in operational systems (at run-time) are common loop-holes exploited by attackers. Unofficial access is often arranged so that operational staff can fix problems remotely. However, the problems are not confined to unofficial procedures - official test and monitoring interfaces can also be overlooked when hardening a system.

As more parts of a network run on virtualised platforms, it becomes possible for the operator to execute ad hoc diagnostic processes using the same virtualisation platforms in order to trace faults or monitor performance. Types of monitoring methodologies introduced will depend upon specific use case and deployment scenarios of these VNFs, such as whether these VNFs run on VMs that reside on the same blade server or spread over VMs spanning multiple blade servers. It will often be convenient to compromise the isolation provided by a VM in order for such a diagnostic process to inspect it. This could introduce additional complexities and security risks due to lack of exposed interfaces between the VNFs or use of proprietary protocols and shared memory aimed at providing optimized operations. For instance, an operator might place a virtual switch in promiscuous mode then run up a virtualised test process to snoop the traffic from the VNF under test and perhaps duplicate the traffic stream into a diagnostic centre. Alternatively, shared memory techniques might be naïvely used to improve the performance of the monitoring process, but this could compromise the integrity of the VNF being monitored. Even if the monitoring VNF is allowed only read-only access to memory shared with the other VNF being monitored, this could compromise the confidentiality of policies or other intellectual property held in that memory. Instead, VNFs can and ought to be designed to provide properly access controlled inspection interfaces. It is particularly important that library functions and development kits provide such interfaces, so that the developers of each VNF do not have to.

It is unclear whether the trend towards using NFV for ad hoc testing and monitoring is likely to improve or degrade security. On the one hand the ability to run-up arbitrary test processes makes it hard to assess the security of a system (e.g. internal fraud). On the other hand, virtualisation technology could be used to create a more structured approach for authorising whether testing and monitoring can be conducted, which diagnostic functions are allowed, and who is allowed to run them.

6.9.2 Developer's debug and test interfaces

Debug and diagnostic (test) interfaces added to network functions at design and implementation time are sometimes left accessible at run-time, so that they can be exploited by bad actors to execute denial of service (DoS) and in some cases be used to penetrate and gain privilege in a network system. In proprietary hardware appliances these debug and test interfaces are often physical ports, jumpers, and switches that can be physically secured. The low volume and diversity of these proprietary platforms makes them a less attractive target for attack.

The viability of this "security by obscurity" posture changes with the adoption of NFV. The functions necessarily will be provisioned, configured, tested, and debugged using software interfaces, perhaps remotely. Software and remote interfaces have very different security characteristics and requirements than the traditional physical interfaces. Further, the adoption of particular vendor's functions could be wide-spread, making a successful vendor's products an attractive target.

6.9.2.1 Industry Experience Example 1

The example below is based on real observations made by contributors to the present document. The intention is not to denigrate any particular product, but to show the serious consequences of this type of problem, which has been seen in several products.

Real-time operating system kernels are available commercially that provide low level services for embedded computation applications. These are used in a wide variety of appliances and products ranging from spacecraft to military and commercial aircraft, from networking devices and communications equipment to home security systems. The development (and runtime) environment for these products often includes several tools designed to run in the embedded device to facilitate both debugging and updating of the embedded application(s).

These tools are required to operate 'remotely' from the actual device, usually over a network interface (wireless or wired), as the devices do not typically have human accessible interfaces for those tasks. These tools can provide remote debugging capabilities via a local agent that runs in the target embedded system.

The agent listens on a well-known network port for debugging commands to read and modify memory and call functions in the operating system kernel and manage application tasks. The vulnerability can exist because embedded system vendors could neglect to disable the local agent or protect its network port with robust authentication and authorization. Leaving the network port and local agent open has allowed attackers to capture data from the system, install new application tasks, or even replace the entire operating system. Many systems built over such kernel products can be deployed with this debugging interface both enabled and unprotected. For instance, [i.5] shows that nearly the whole industry did not have secure processes to detect or close off an open debug service in the virtualisation infrastructure beneath their products.

Such environments sometimes make it easy for equipment vendors to leave management, debug and diagnostic security credentials embedded in the firmware images (intentionally or not). The embedding of security credentials in the images allows attackers to discover them and potentially gain access to the devices through the network interface (typically using FTP or Telnet).

The embedded systems exposing these vulnerabilities are examples that highlight the need for specific attention in Virtualised Network Function architecture, design, and development. Like VNFs, the affected embedded systems run an operating system and are implemented as software elements and might be required to provide remote management and diagnostic interfaces.

6.9.2.2 Industry Experience Example 2

In the application specific integrated circuit (ASIC) industry, particularly in network controllers, industrial controllers, and network switching silicon, it has been common to have special purpose interfaces for debug and test. These interfaces include electrical (JTAG or joint test action group buses) as well as software and firmware interfaces. These interfaces have been exploited by both DoS attacks and privilege escalation attacks. A notable example of this type of attack is the result of a 'buffer overflow' vulnerability in a particular network controller's firmware. Once the buffer overflow is exploited the attacker can put the device into a 'Remote Management' mode. Arbitrary code can then be executed on the controller allowing DoS, penetration, and breach of confidentiality attacks.

6.9.2.3 Guidance for the NFV ISG working groups:

While these issues are not necessarily specific to NFV, the migration of network functions from hardware appliances/devices to software running in VMs intensifies the requirements for securing or disabling debug and test interfaces. As a proprietary, integrated appliance, the network function could have physical ports, jumpers, or switches to enable/disable special modes and these can be sufficiently protected by the physical security of the data centre. In an NFV environment, that physical security characteristic is no longer sufficient.

The central recommendation of this contribution is to discourage 'security by obscurity' and the use of fixed, embedded authentication credential for debug and test for any virtualised network function. Ideally, debug and test interfaces for developers need to be completely (and irreversibly) disabled when the functions are provisioned in the production environment. If an image cannot be free of sensitive information, it needs to have confidentiality protection in addition to customary integrity protection and access control. In this case, secure key management is also necessary.

6.10 Multi-Administrator Isolation

The ETSI Technical Committee on Lawful Interception (TC LI) has documented an initial view of the additional problems and considerations that need to be taken into account when virtualising network functions in an environment that needs to support lawful interception [i.31].

Usually there is a simple hierarchy of administrators, where the administrator of the virtualisation infrastructure has privileges greater than or equal to the privileges of any of the administrators of the virtualised functions executing on the system. However, lawful interception requirements highlight the need to generalise to scenarios in which the administrator of the virtualisation infrastructure enjoys lower privileges than the operator of certain virtualised functions.

7 General Concerns

7.1 Safety vs. Complexity

There is a lot of talk about 'attacks' in the present document; however the problem is as much about accidents as malice. Virtual networks are already hard for humans to model in their minds, so virtualising the nodes as well as the networks only increases the chances of errors. Accidental misconfigurations can in themselves cause service problems, or they can leave unintended vulnerabilities, which will cause service problems if exploited. Complexity also makes it harder to defend against internal threats.

This is a general and vague warning, but it is an important one.

7.2 Altered Norms and Procedures

We consider two types of altered norms that might not strictly be unique to NFV but could be more likely to happen with NFV than in today's environment: *broken procedure*, and *software vulnerabilities*. The warnings in this clause are general and vague, but still important.

7.2.1 Broken Procedure

In past times when most network functions came in a box connected by cables, few attacks involved connecting together networks that were not intended to be connected together. Such attacks would require someone to walk into a network facility with a box and patch it in, so they were rarely even considered.

Once software for all the different network functions is widely available, and once server blades with lots of physical connections are sitting in network facilities ready to host a wide range of network functions, the attack surface has expanded in subtle ways. Even if all the platforms are properly access controlled, security relies on a new set of procedures. The most vulnerable parts of a system are usually the human procedures, not the technical protocols and designs.

For instance, even if the monolithic operator deployment model is used (see clause 5.1), the risk of an 'inside-job' is increased, because the staff of the network operator no longer need physical access to network operations centres to execute attacks that could previously only be mounted by entering the physical building.

In the past, it may not have been worth exploiting a broken procedure just to run an application. But if it is now feasible to run a function that connects together networks that should be partitioned, it may become worth someone's effort to exploit the same broken procedure.

7.2.2 Software Vulnerabilities

There are usually three types of software flaws, namely design flaws, implementation flaws, or configuration flaws. Software vulnerabilities could exist in any software including that compiled for bespoke hardware appliances (referred to as non-NFV hereafter) as well as NFV, so they are certainly not new to NFV. However, the risks from software vulnerabilities could be higher with NFV than with traditional bespoke appliances for the following reasons:

1) VNFs are expected to run on commodity software and hardware. This consolidates the efforts of both attackers and defenders, but it is unclear whether this makes systems easier to attack or easier to defend. Given attackers only need to find one vulnerability, while defenders have to find and fix them all, it could be concluded that consolidation aids defenders more than attackers. If this optimistic view is incorrect, compared to non-NFV, NFV would face the following three changed norms:

    - There are more people conducting vulnerability research on commodity software, resulting in more reported and unreported (aka zero-day) vulnerabilities.

    - There are more people writing exploiting code for commodity software, resulting in more attack tools available, such as virus and rootkits.

    - There are more people trying to exploit commodity software (for fun, fame, or finance). In fact, many exploits are now even automated, and the first external visitors to new servers added to the Internet are typically not human beings, but search robots and malicious scanning and exploitation tools.

2) NFV builds on cloud security technology developed to isolate individual virtualised applications from each other. Network functions could provide critical networking infrastructure, which is a higher value target than individual virtualised applications from an attacker's perspective. Therefore, virtualisation technology will need to be re-assessed before it can be considered suitable for protecting critical network infrastructure. For example, if a virtualised BGP router were taken over by an attacker, perhaps exploiting a critical vulnerability in the underlying commodity virtualisation infrastructure software, it could be used to advertise address space owned by a victim to hijack traffic from that victim. This could result in large scale of service disruption or heavy data loss. Such an attack would be difficult to launch without attacking network infrastructure.

Once identified, vulnerabilities in software can be fixed through security patches whereas hardware vulnerabilities are much more costly to fix. However, security patching is not necessarily a panacea:

1) Security patches may require a reboot and, if the patch is to be applied to the compute infrastructure, rebooting commodity servers hosting multiple network functions could cause service disruption, particularly if many commodity servers have to be rebooted over a short period.

2) Security patches are not always in time. There are many security vulnerabilities that had been discovered and shared privately (for commercial, political, or malicious purposes) long before they were made public. How to handle such zero-day vulnerabilities in commodity software could pose challenges for NFV.

Because it is possible to update software frequently, it tends to be updated much more often than hardware. This is because software makes it feasible to be early to market, leaving feature extensions, bug-fixes and security updates for later. The more frequent update cycle of software will make it less likely that each update of a VNF or of the underlying compute infrastructure will have been so exhaustively tested for security vulnerabilities, hence, secure software update procedures need to be defined.

Annex A (informative): Authors & contributors

The following people have contributed to the present document:

Rapporteur:

Mr Bob, Briscoe, BT

Other contributors:

Mr Dacheng, Zhang , Huawei

Mr Peter, Ashwood-Smith, Huawei

Mr Uwe, Janssen, DT

Mr Bob, Moskowitz, Verizon

Mr, Tao, Wan, Huawei

Mr Marc, Millier, Intel Corporation

Mr Brian, Skerry, Intel Corporation

Mr. Kapil Sood, Intel Corporation

Mr Igor, Faynberg, Alcatel-Lucent

Ms Hui-Lan, Lu, Alcatel-Lucent

Mr Mike, Bursell, Citrix

Mr Diego, Lopez, Telefonica

Mr Kurt, Roemer, Citrix

Mr Ashutosh Dutta, AT&T

Mr Shucheng Liu (Will), Huawei

Mr Fernando Gont, Huawei.

Ms Marie-Paule Odini, Hewlett-Packard

Apologies to anyone omitted.

Bob Briscoe's work on the present document was part-funded by the European Union's Seventh Framework Programme FP7/2007-2013 under the Trilogy 2 project, grant agreement n°317756.

History

Document history		
V1.1.1	October 2014	Publication