Virtual Security Appliances: The Next Generation Security

Yacine Rebahi, Simon Hohberg, Lijuan Shi
Fraunhofer Fokus
Kaiserin Augusta Allee 31
10589 Berlin, Germany
{yacine.rebahi, simon.hohberg,
lijuan.shi}@fokus.fraunhofer.de

Bruno Miguel Parreira, Anastasios Kourtis
Portugal Telecom
Demokritos
{bruno-m-parreira@telecom.pt}
{kourtis@iit-demokritos.gr}

Paolo Comi
Italtel, S.p. A, Innovation & Research
paolomaria.comi@italtel.com

Aurora Ramos
Atos S.A.
Spain
aurora.ramos@atos.net

Abstract— **We report in this paper on research in progress related to network functions virtualization (NFV) and their use in network security. Our objective is to design and develop a virtual Security Appliance (vSA) capable of detecting various network attacks while offering an acceptable level of performance. In this document, we introduce the vSA under construction and show some testing results we have recently obtained**.

Keywords—network function, virtualization, security, security appliance, firewall, IDS, T-NOVA

I. INTRODUCTION

A Security Appliance (SA) is simply a "device" designed to protect computer networks from unwanted traffic. This device can be active and block unwanted traffic. This is the case for instance of firewalls and content filters. A security Appliance can also be passive. Here, its role is simply detection and reporting. Intrusion Detection Systems are a good example. If the SA is in charge of scanning the network and identifying potential breaches (e.g, penetration testing), the SA can be qualified as preventive. Nowadays, Security Appliances combine various security features including firewalling, content filtering, and intrusion detection. For this reason, they are more commonly known as Unified Threat Management (UTM) systems.

Conventional security appliances (in particular firewalls) were deployed at the network border in order to examine the traffic destined to this network. As networks become more complex, it is often necessary to place these appliances between multiple network segments. With NFV and cloud computing, the situation becomes more challenging as entire networks or network segments can be hosted completely within a virtual environment. As a result, security appliances need also to protect virtual environments in addition to physical networks. The burden of this task can be carried by security appliances running on virtual machines. A virtual security appliance (vSA) is a network security service running entirely within a virtualized environment.

As a virtual security appliance might span different security technologies, the type of security being used in this context is more important when it comes to performance. In the past, the performance of the appliance was achieved through dedicated hardware. In virtualized environments, this is not possible for the reason that different applications might run on the same operating system and compete for the same hardware computing resources.

In this paper, we discuss the initial steps towards virtualizing security appliances. Indeed, different issues related to the deployment of such appliances need to be addressed,

- *Security appliance complexity:* identifying whether the appliance is formed by one or more components
- *Internal communication:* in case the SA includes different components, how to ensure proper communication between virtual machines running the different components
- *Integration with the virtual infrastructure:* when setting up the SA components, we also need to ensure that virtual platforms such as OpenStack facilitates the necessary integration including IP addresses provision, traffic mirroring if needed, and so on
- *Impact on the performance:* here a selection of security technologies that could be virtualized might be necessary. For instance, firewalls could be a "good" candidate as they inspect small amounts of data (packet headers) and are in general stateless. In addition to that, they could run as a part of the operating system which will not affect the performance of the system

In order to have a realistic scenario and deal with a broader range of attacks, we propose a virtual security appliance composed of,

- A virtual firewall used as an entry point to the network and which will be in charge of "simple" packet filtering.
- An intrusion detection system that handles deeper packet inspection and issues alerts in case of malicious traffic.

This paper is organized as follows. Section II overviews the context in which this vSA is being specified and developed, and section III provides a high-level description of the vSA architecture. In section IV, we discuss the components that have been implemented so far as well as their related performance. Sections V and VI respectively describe the

978-1-4673-6547-5/15/$31.00 ©2015 IEEE

state of the art as well as future possibilities. Finally, section VII concludes the paper.

II. USE CASES AND CONTEXT

A. Use cases

Security appliances need to be placed in the network traffic path. This applies to both passive and active network elements. Ideally all traffic should pass through the security elements and because of that, these elements are usually placed near the Costumer Edge Router (CE). This applies to both residential and enterprise scenarios, although in the former the requirements are much lighter and therefore security appliances are usually integrated in the CE. In the enterprise scenario, they are composed by specific and dedicated hardware and are placed within the customer domain (see **Figure 1**).

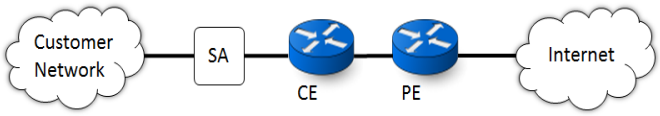

Figure 1. Legacy Security Appliances placement

The process of virtualizing security appliances by itself does not constitute a service requirement. Therefore, it must be as transparent to the network as possible while maintaining the legacy placement and functionality.

With the NFV paradigm, some changes in the infrastructure are expected, such as the appearance of the NFVI-PoPs. These are small and distributed datacenters which can be used to deploy VNFs. The actual number and placement of NFVI-PoPs is still under heavy discussion, but typically they should be located close to the Provider Edge Routers (PE). With this in mind, the placement of the Security Appliances virtual counterparts (vSA) might lead to two distinct cases,

- Non-virtualized CE – in this case, the vSA will be placed between the CE and the PE. This situation forces a change in the traditional network topology. Nevertheless, this alteration should be almost transparent and the vSA functionality should be maintained (see **Figure 2**)
- Virtualized CE – in this case, the logical placement of the vSA and the CE is identical to the traditional scenario and both functions are moved to the NFVI-PoP. Moreover, the previous physical CE needs to be replaced by a L2 Bridge to extend the customer network broadcast domain to the virtual environment located at the NFVI-PoP (see **Figure 3**)

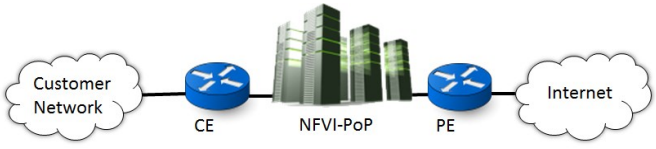

Figure 2. vSA deployed at the NFVI-PoP

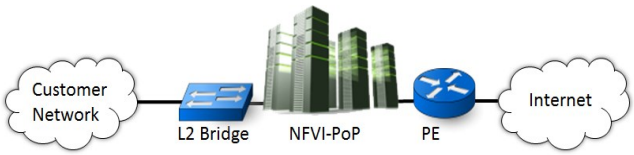

Figure 3. vSA and vCE deployed at the NFVI-PoP

In this paper, only the security appliance will be virtualized, because the use of virtual CE presents other challenges that are outside the scope of this work. Also, from the vSA perspective, the use or not of a virtual CE should be transparent.

Finally, only the enterprise use case will be contemplated since a vSA in a residential scenario would be exaggerated. In this use case, a company wants to take advantage of virtualization to deploy network security functions without compromising its performance and functionality. Moreover, by moving to a cloud environment, the company wishes to maintain the legacy network infrastructure as close to reality as possible. This network infrastructure might include Telco VPN resources which must not be affected by the new services.

B. ETSI NFV ISG

In the area of Network Functions Virtualization ETSI NFV ISG represents one of the most relevant standardisation initiatives. Security appliances are identified by ETSI NFV in [24] as one of the main network functions typically deployed today within enterprise networks as dedicated hardware infrastructure which, in the near future, may become appropriate for a Service Provider to deliver on a VNFaaS basis to the enterprise. NFV management and orchestration specification by ETSI NFV [25] currently provides some examples for VNFD (Virtual Network Function Descriptor) for a virtual firewall to be deployed within a NFV Infrastructure to which the work in this paper is aligned to, however no further work on experimentation or proof of concepts has been done so far by ETSI with vSA [26], so it is expected this work will contribute in that direction to the current NFV state of the art.

C. The T-NOVA project

This work has been undertaken in the context of the European

project T-NOVA that is an Integrated Project co-funded by the European Commission / 7th Framework Programme, Grant Agreement no. 619520. T-NOVA aims at designing and implementing integrated management NFV architecture, including an Orchestrator platform, for the automated provision, management, monitoring and optimization of Virtualised Network Functions over Network/IT infrastructures. Furthermore, T-NOVA introduces novel Marketplace exploring new business cases arising in the NFV scheme, expanding market opportunities by attracting new entrants and lowering barriers the of the networking market for software developers. SMEs and academia can leverage

the T-NOVA architecture by developing innovative cutting-edge Network Functions as software modules, which can be included in the T-NOVA Function Store, and rapidly introduced into the market to monetize them under several billing models options and supporting SLA management. T-NOVA also provides a common intersection point between developers and telecom operators, avoiding the delay and risk of hardware integration and prototyping, leading to more performant networks and reducing time-to market for new VNFs. The vSA virtual security appliance described in this paper is one of the VNFs that will be developed in T-NOVA for proving its value and effectiveness. For more details, we refer to [23].

III. ARCHITECTURE

A. Requirements

The virtual Security Appliance (vSA) we are willing to develop aims at fulfilling the following requirements,

- The vSA shall protect the service from malicious traffic
- The vSA shall provide simple traffic filtering as well as deep inspection
- The vSA shall run on virtual machines
- The vSA shall provide appropriate APIs for configuration
- The vSA shall provide an acceptable level of performance
- The vSA shall be flexible enough to enable detection rules revision

B. High-level architecture

The architecture of this appliance is depicted in Figure 4 and includes the following main components,

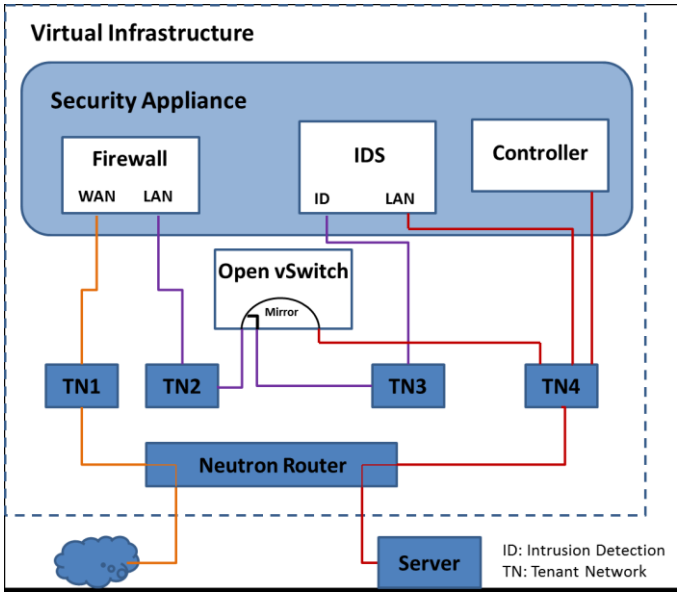

Figure 4. vSA high-level architecture

The firewall: this component is in charge of filtering the traffic towards the service. As discussed later on, this component would run open source firewall software extended to fulfill the requirements of the use case discussed in section II. It is worth mentioning that packet filtering firewalls are often unable to discover packets with malicious payload on their own as they just look at the source address, destination address, protocol, and port number.

The Intrusion Detection System (IDS): In order to improve attack detection, a combination of a packet filtering firewall and an intrusion detection system using both signatures and anomaly detection is considered. In fact, Anomaly detection IDS have the advantage over signature based IDS in detecting novel attacks for which signatures do not exist. Unfortunately, anomaly detection IDS suffer from high false-positive detection rate. It is expected that combining both arts of detection will improve detection and reduce the number of false alarms. In T-NOVA, an appropriate existing signature based IDS (e.g, Snort [1], Bro [2], Suricata [3]) will be extended to support anomaly detection as well. The mode of operation of the IDS component is depicted in Figure 5.

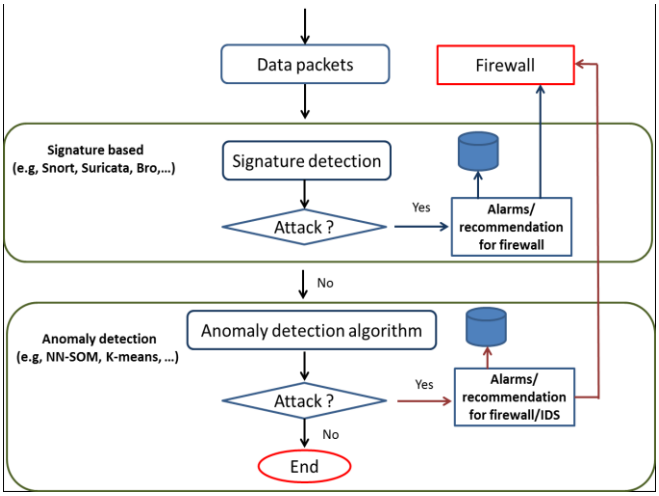

Figure 5. IDS process flow diagram

The different components of the architecture interwork in the following way,

- The data packets are first of all filtered by the firewall
- The data packets are also duplicated (mirrored) through Open vSwitch and sent to the IDS for further inspection. The IDS will monitor and analyze all the services passing through the network
- As a first step, the data packets go through a signature based procedure. This will help in detecting efficiently well know attacks such as port scan attacks and TCP SYN flood attacks
- If an attack is detected at this stage, an alarm is generated and the firewall is informed to revise its rules

- If no attack is detected, the data packets will be passed to an anomaly detection algorithm. In our context, it will be NN-SOM due to its interesting characteristics
- In the same way, if an attack is detected, an alarm is generated and the firewall will be contacted to revise its rules
- If no attack is detected, no further action is required

Open vSwitch: As the firewall and the IDS run on different virtual machines and need to interact with each other, a third component is needed to facilitate this interaction and forward the traffic between the firewall and the IDS virtual machines. For this purpose, Open vSwitch (RFC 7047) is going to be used. It is open source software (client and server) designed to be used as a virtual switch. It can also be extended and controlled using OpenFlow and the OVSDB (Open vSwitch Database) management protocol. For the deployment of Open vSwitch in our architecture, there are two possibilities. First, Open vSwitch is a part of the security appliance, but in this case the deployment on the virtual infrastructure (e.g, OpenStack) will also require a switching functionality (on the virtual infrastructure) to communicate with the outside network. The second option is simply the deployment of Open vSwitch directly on the virtual infrastructure. This solution also prevents switching functionalities duplication.

The Controller: On the one hand, in the vSA the firewall actively blocks unwanted traffic which is an effective measure to protect against a variety of known attacks. On the other hand the IDS detects suspicious traffic passing the firewall and generates alerts. The vSA Controller now combines both functionalities to allow the vSA to actively react to attacks by analyzing the alerts generated from the IDS and adapting the firewalls configuration in order to stop the attack.

IV. Implementation and performance

A. Technology selection

In the context of this paper, we focus more on the use of firewalls as the implementation of the entire appliance is still ongoing. To be more specific, a Firewall (FW) is a program/device that simply filters the network traffic. It controls the traffic (in and out) using one of the following methods,

- Packet filtering
- Proxy service
- Stateful inspection

As performance is one of the main issues when deploying software versions of security appliances, we will **first** provide a short evaluation (partly based on [4] and [5]) of firewalls software that can run in virtual environments. The idea is not to go through all the relevant existing software but just the most popular ones that could be extended to fulfill the use case requirements.

Firewall	Evaluation
Vyatta VyOS [6]	VyOS is a community fork of Vyatta, a Linux based network operating system that provides software-based network routing, firewall, and VPN functionality ▪ Supports paravirtual drivers and integration packages for virtual platforms. ▪ Completely free and open source **pros:** open source, large user base, REST APIs, high performance, root shell, support for IDS
pfSense [7]	The pfSense project is a free network firewall distribution, based on the FreeBSD operating system with a custom kernel and including third party free software packages for additional functionality OS: FreeBSD **pros:** Open source, Web User Interface, very easy to use, large community, root shell, integration of external packages **cons:** incomplete bgp/ospf, xml config, no config cli, no REST APIs
Halon [8]	Halon Virtual Security Router (VSR) is an OpenBSD-based firewall, router, VPN and load balancing appliance focusing on security, flexibility and manageability. OS: OpenBSD **pros:** Open source, Web User Interface, SOAP APIs, juniper-style config/rollback/commit, inexpensive, root shell, pkg_add **cons:** small community, unknown vendor, no IPS functionalities
m0n0wall [9]	m0n0wall is a project aimed at creating a complete, embedded firewall software package that, when used together with an embedded PC, provides all the important features of commercial firewalls. m0n0wall is based on FreeBSD, along with a web server, PHP and a few other utilities. The entire system configuration is stored in one single XML text file to keep things transparent. Provides packet filtering, VPN, NAT, IPS OS: FreeBSD **pros:** open source, very small image, root shell, pkg_add **cons:** less feature complete than pfSense
Vuurmuur [10]	Vuurmuur is a powerful firewall manager built on top of iptables that works with Linux kernels 2.4 and 2.6. It has a simple and easy to learn configuration that allows both simple and complex configurations. OS: GNU GPL **pros:** open source, no iptables knowledge required, human readable rules syntax, Ncurses GUI, no X required, potential integration with IDS/IPS, traffic volume accounting **cons:** no REST APIs for configuration, less feature complete than pfSense

From the above table, the open source firewalls that are richer and more complete are Vyatta VyOS and pfSense. In addition to that, VyOS seems to support REST APIs for configuration which are important in the integration with the rest of the T-NOVA framework.

These two options will be evaluated from the performance point of view and the best one will be utilized as a component within the vSA.

B. Performance

Firewalls are often implemented in routers to control packet flows. If the packet filtering process generates an extra overhead, this will, certainly, affect the performance of the system and lead to degradation in its time response.
To study the performance of firewalls, benchmarking techniques are needed. Unfortunately, activities in this area are very scarce. As an example, the IETF Benchmarking Methodology Working Group [11] produced several Request for Comments (RFCs) describing benchmarking terminology and methodology for a wide range of networking devices. Performance benchmarks related to firewalls are discussed in RFC 2647 and RFC 3511. The suggested methodologies are intended to be standard benchmarking for all classes of firewalls. Unfortunately, this makes them too general to be applied to a particular class of firewall. So far, it seems to us that the methodology suggested by Kean and Mohd [12] for evaluating firewalls performance is well suited. This methodology suggests the following metrics,

Throughput: The maximum rate at network layer which none of the received packet is dropped by the firewall without activating filtering rules. In RFC 2647, the throughput is defined as the actual payload that is received per unit of time
Latency: The time interval starting when the last bit of input frame reaches the input interface of the firewall, and ending when the first bit of the output frame is observed at the output interface of the firewall
Jitter: Measures the variation in delay of the received packet
Goodput: The rate at which packets are forwarded to the correct destination interfaces of the firewall, excluding any packets dropped due to the rule set definition. The goodput could be seen as the opposite of the Packet Loss Rate (PLR) which is the ratio of the lost packets to the total of transmitted packets

C. Testbed setup

For simplicity reasons, we have used Iperf [13] for generating IP traffic in our tests. In fact, other IP traffic generators such as D-ITG [14], ostinato [15], and IPTraf [16] could have also been utilized. Iperf mainly generates TCP and UDP traffic at different rates. Diverse loads (light, medium, heavy) and different packet sizes are also considered. For analyzing IP traffic, we used "tcpdump" for capturing it and "tcptrace" to analyse it and generate statistics. As for the virtualization, VirtualBox [17] was used.

To run our tests, we decided to use two hosts. On the first one, we have installed the Iperf client and server and on the second one, we have setup the firewall under tests. This setup is in fact in line with the recommendations provided in RFC 2647. The characteristics of the used hosts are as follows,

1st Host System (Client/server)

CPU	Intel(R) Xeon(R) CPU E5540 @ 2.53GHz (4 Cores)
RAM	12 GB
Graphics	NVIDIA G94GL [Quadro FX 1800]
OS	Ubuntu 3.11.0-26.45-generic 3.11.10.12

Guest System (Client/server)

CPU	2 CPUs with 95% execution cap
Acceleration	VT-x/AMD-V, Nested Paging
RAM	512 MB (Client), 1024 MB (Server)
Graphics	12 MB Video Memory
OS	Lubuntu 14.10 x86-64 (Linux Kernel 3.16.0-28-generic)
Network Adapter	1 Gbit/s

2nd Host System (Firewall)

CPU	Inter(R) Core(TM)2 CPU T7200 @ 2.00GHz (2 Cores)
RAM	2 GB
Graphics	RV515/M54 [Mobility Radeon X1400]
OS	Ubuntu 3.13.0-45.74-generic 3.13.11-ckt13

Guest System (Firewall)

CPU	2 CPUs with 95% execution cap	
Acceleration	VT-x/AMD-V, Nested Paging	
RAM	1208 MB	
Graphics	12 MB (Vyos), 10 MB (PfSense) Video Memory	
HDD	8 GB (Vyos), 2 GB (PfSense)	
OS	*PfSense*	Release 2.1.5 (FreeBSD 8.3-RELEASE-p16)
	Vyos	Release 1.1.0 - Helium (Debian 6.0.10 - Squeeze, Linux Kernel 3.13.11-1-amd64-vyos)
Network Adapter	1 Gbit/s	

D. Testing scenarios

The undertaken tests are based on three main scenarios,

- **Scenario one** (No firewall): Here, we configure and check the connectivity between the Iperf client and server without a firewall in between. This enables us to test the capacity of the communication channel
- **Scenario two (**TCP traffic with firewall and no rules): Here, we check whether the introduction of a firewall (running on a virtual machine in between) generates extra delay. We also test the capacity of the firewall in this context
- **Scenario three** (with firewall and increasing number of rules): the objective of this scenario is to study the effect of introducing rules into the firewall. To achieve this scenario, some scripts for both pfsense and Vyos are implemented to generate rules in an automatic way. The scripts are shell scripts using specific API commands and generate blocking rules for random source IP addresses (excluding those used in the test setup) and the WAN interface. For pfsense, the easyrule function is extended and for VyOS, the "configure" environment (set of commands) is used**.** In this scenario, some tests are also performed using UDP instead of TCP

E. Tests results

When no firewall is used between the Iperf client and server, one can note that the throughput of the communication remains good (700 Mbit/s) as long as the number of parallel connections does not exceed 7 connections. When the number of connections goes beyond this value, the throughput decreases very fast to reach 0 when 20 connections are opened (Figure 6). One can also notice that the Round Trip Time (RTT) is severely affected when increasing the number of connections between the Iperf client and server (Figure 7).

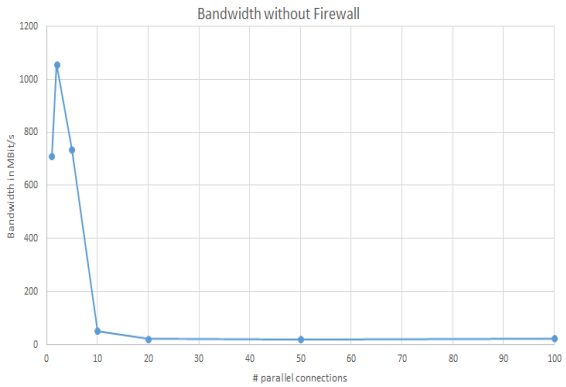

Figure 6. Throughput without firewall

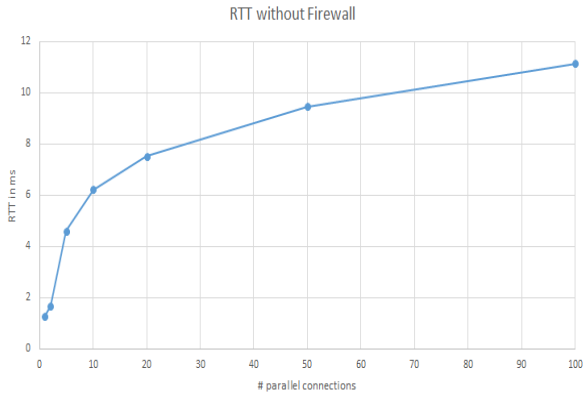

Figure 7. RTT without firewall

The results obtained from a firewall (pfsense or Vyos) being settled between the Iperf client and server, the variation of the throughput and the RTT are depicted in Figure 8 and Figure 9 respectively. One can note that pfsense, in both cases, presents a more stable behavior when the number of connections increases.

As a third step, we also wanted to check the impact on the setup when we increase the number of rules in the firewall. Indeed we started with 10 rules, then 100 rules and ended up with 1000 rules. The performance results are depicted in figures 10 to 13. The case with 100 rules was omitted because the behavior of the firewalls in this case is similar to the one with 10 rules. One can clearly see that in all these cases pfsense behaves better than Vyos.

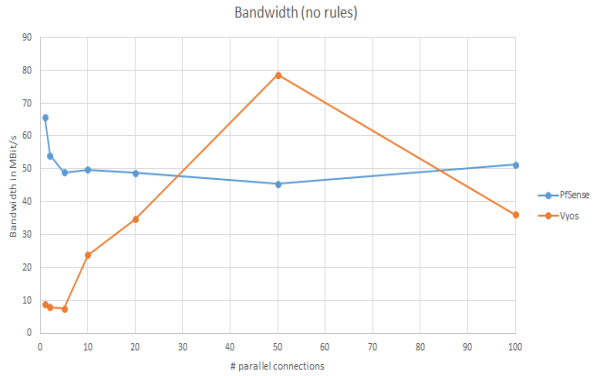

Figure 8. Firewall comparison without rules (Throughput)

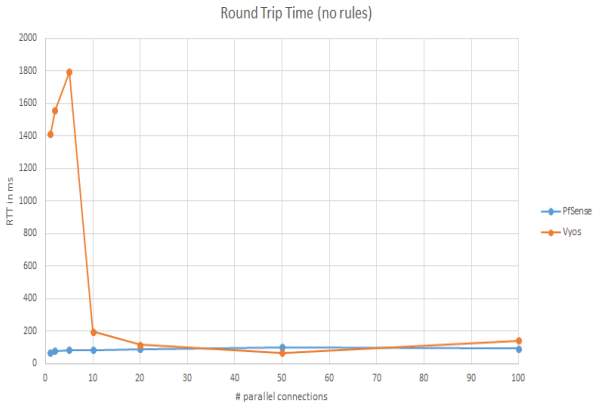

Figure 9. Firewall comparison without rules (RTT)

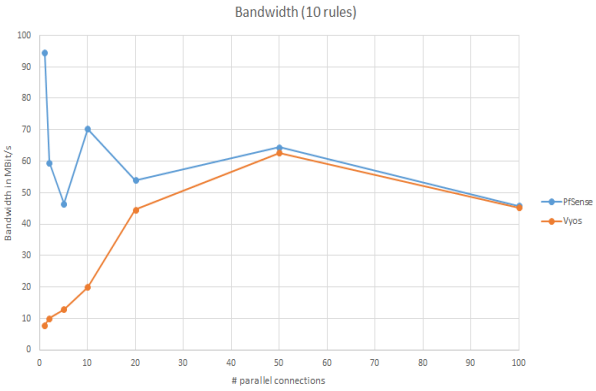

Figure 10. Firewall comparison with 10 rules (Throughput)

One of our objectives was also to investigate the impact of UDP on the behavior of the firewalls. To achieve this, Iperf was used here as well to generate UDP traffic. One can see that VyOS behaves slightly better than pfSense when it comes to packet jitter (Figure 14). Contrary to this fact, no

difference could be distinguished between pfSense and VyOS with respect to goodput (Figure 15) as the lines describing the behaviors are superimposed.

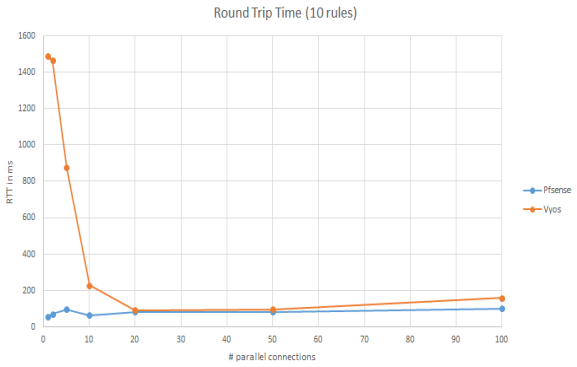

Figure 11. Firewall comparison with 10 rules (RTT)

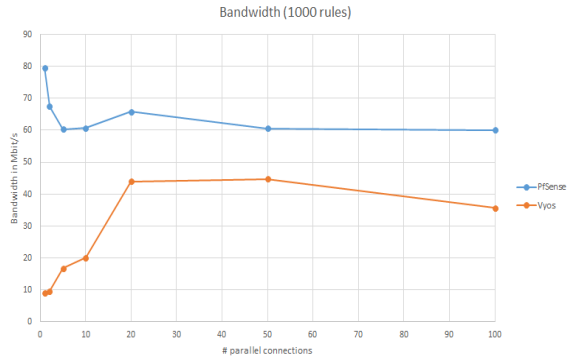

Figure 12. Firewall comparison with 1000 rules (Throughput)

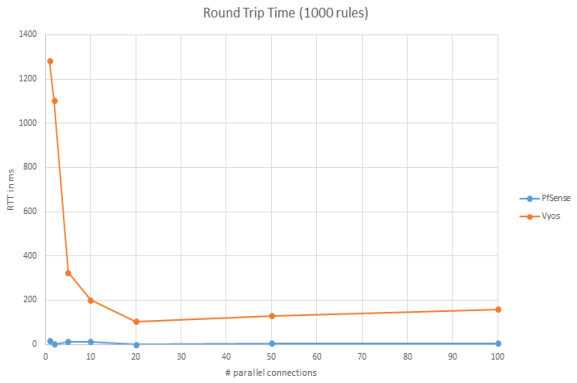

Figure 13. Firewall comparison with 1000 rules (RTT)

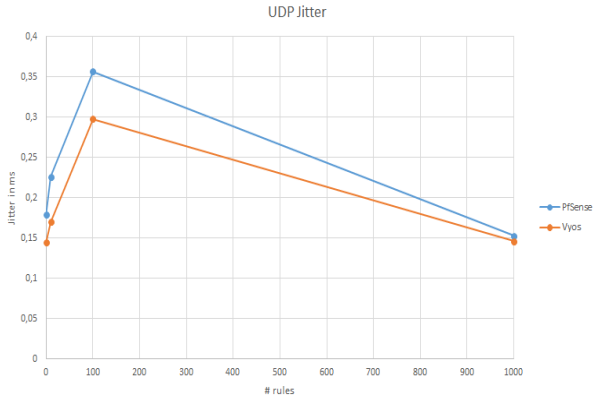

Figure 14. UDP jitter comparison

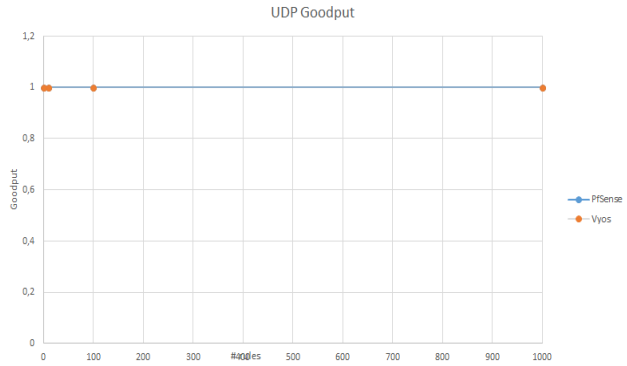

Figure 15. UDP goodput comparison

V. State of the Art

The use of virtualization in intrusion detection was addressed by several research activities that were partly summarized in [18]. Intrusion detection based on virtual machines offers isolation of the monitored environment as well as interesting features such as fast startup, shutdown and recovery. To enhance the security in virtualized environments, the authors of [19] developed an architecture that monitors the calls issued by the hypervisor. In [20], the log files of the virtualized IDS systems were analyzed using big data techniques in order to cope with attacks quickly. In [21], a distributed architecture was proposed where a central controller manages separate instances of IDS settled for the different users. The IDS instance will monitor the activities of the user and build a knowledge profile that will be used as a basis for further monitoring of this user. *The approach that has some common points with the one described in this paper* was discussed in [22]. Here, Intrusion Detection Systems as a Service (IDSaaS) is implemented based on Amazon Elastic Compute Cloud (EC2). Indeed, instances of EC2 VM type are created to run security components in the infrastructure. The detection mechanisms provided are completely controlled by the users. Moreover, IaaS platforms traditionally offer security services to increase protection on virtual resources.

AWS not only offers ACLs on each managed interface but also provides monitoring on network and server usage, port scanning activities, application usage and unauthorized intrusion attempts [27]. Although the purpose of our work is to design and implement a virtual security appliance as a service, the focus of this paper is more on the investigation of issues related to the deployment on well-known virtual infrastructures such as OpenStack, VNF lifecycle, interaction with the orchestrator, and performance.

VI. NEXT STEPS

This work was implemented and tested in a VirtualBox environment, which does not count as a complete NFV infrastructure. Future work for the proposed vSA appliance includes deployment in a completely virtualized network environment, e.g. OpenStack, in order to be tested and validated in more complicated scenarios. The OpenStack deployment may raise several networking issues in terms of automated VNF deployment, Service Function Chaining (SFC), traffic forwarding and inter-VM communication, required for the vSA to function properly. The automated and functional integration of this work's vSA to OpenStack's networking environment, and more specifically to Neutron service, is non-trivial and remains to be substantiated and implemented as Neutron at the moment does not offer much freedom and flexibility on arbitrary traffic steering. The future added value of this work would be an automated, flexible and efficient Security Appliance for virtualized network infrastructures.

In order to support direct traffic forwarding, meaning the virtual network interface of one Virtual Network Function Component (VNFC) to be directly connected to another VNFC's virtual network interface, a modification on Neutron's OVS needs to be applied. Each virtual network interface of a VNFC is reflected upon one TAP-virtual network kernel device, a virtual port on Neutron's OVS and a virtual bridge connecting them. This way, packets travel from the VNFC to Neutron's OVS through the Linux kernel. The virtual kernel bridges of the two VNFCs need to be shut down and removed. Then an OVSDB rule needs to be applied at the Neutron OVS, applying an all-forwarding policy between the OVS ports of the corresponding VNFCs.

After the integration of the vSA with OpenStack, the vSA will also be integrated with the entire T-NOVA framework. The related issues such as the VNF Descriptor, the VNF lifecycle, the interaction with the orchestrator, and VNF monitoring will be implemented.

VII. CONCLUSION

In this paper, we have proposed a type of architecture for a virtual security appliance combining different security technologies including firewalling and intrusion detection. We have also explained how this appliance could be deployed in a virtual environment. As the implementation is still ongoing, we have focused, in this paper, only on the selection and use of the firewalling functionality. Here, different test scenarios were defined in order to choose between several potential open source "virtual" firewalls. In addition to that, an overview of the current status of the virtual security appliance implementation as well as the challenges being faced was provided.

REFERENCES

[1] Snort, link: https://www.snort.org/

[2] Bro, link: https://www.bro.org/

[3] Suricata, link: http://suricata-ids.org/

[4] http://www.reddit.com/r/networking/comments/1rpk3f/evaluating_virtual_firewallrouters_vsrx_csr1000v/

[5] http://en.wikipedia.org/wiki/Comparison_of_firewalls

[6] Vyatta VyOS, link: http://vyos.net/wiki/Main_Page

[7] Pfsense, link: https://www.pfsense.org/

[8] Halon, link: http://www.halon.se/

[9] m0n0wall, link: http://m0n0.ch/wall/

[10] Vuurmuur, link: http://www.vuurmuur.org/trac/

[11] BMWG, link: https://www.ietf.org/proceedings/32/charters/bmwg-charter.html

[12] http://core.kmi.open.ac.uk/download/pdf/11778682.pdf

[13] Iperf, link: https://iperf.fr/

[14] D-ITG, link: http://traffic.comics.unina.it/software/ITG/

[15] Ostinato, link: https://code.google.com/p/ostinato/

[16] Iptraf, link: http://iptraf.seul.org/

[17] VirtualBox, link: https://www.virtualbox.org/

[18] J. D. Araujo, Z. Abdelouahab, "Virtualisation in Intrusion Detection System: A Study on Different Approaches for Cloud Computing Environments", IJCSNS International Journal of Computer Science and Network Security, Vol 13, No 11, Nov 2013

[19] Bharadwaja, S.; Weiqing Sun; Niamat, M.; Fangyang Shen; , "Collabra: A Xen Hypervisor Based Collaborative Intrusion Detection System," Information Technology: New Generations (ITNG), 2011 Eighth International Conference on , vol., no., pp.695 -700, 2011

[20] Shun-Fa Yang; Wei-Yu Chen; Yao-Tsung Wang; , "ICAS: An inter-VM IDS Log Cloud Analysis System," Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on , vol., no., pp.285-289, 15-17 Sept. 2011

[21] Dhage, S.N., Meshram, B.B.,Rawat, R., Padawe, S., Paingaokar, M., Misra, A. Intrusion Detection System in Cloud Computing Environment, in International Conference and Workshop on Emerging Trends in Technology(ICWET 2011) –TCET, Mumbai, India. 2011

[22] Turki Alharkan, Patrick Martin. IDSaaS: Intrusion Detection System as a Service in Public Clouds. In 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Co mputing, pp 685-687, 2012

[23] T-NOVA project, link: http://www.t-nova.eu/

[24] ETSI NFV ISG. ETSI GS NFV 001 v1.1.1 Network Functions Virtualisation; Use Cases. s.l. : ETSI, 2013.

[25] ETIS NFV ISG. ETSI GS NFV-MAN 001 V1.1.1 Network Function Virtualization (NFV) Management and Orchestration. 2014-12

[26] ETSI NFV ISG. ETSI GS NFV-PER 002 V1.1.2 Network Functions Virtualisation (NFV); Proof of Concepts; Framework. 2014-12

[27] Amazon Web Services: Overview of Security Process; 2015-08