An SFC-enabled approach for processing SSL/TLS encrypted traffic in Future Enterprise Networks

Vitor A. Cunha*, Marcio B. de Carvalho†, Daniel Corujo*, Joao P. Barraca*, Diogo Gomes*,
Alberto E. Schaeffer-Filho†, Carlos R. P. dos Santos‡, Lisandro Z. Granville†, Rui L. Aguiar*
*Instituto de Telecomunicações, Portugal
†Institute of Informatics – Federal University of Rio Grande do Sul – Porto Alegre, Brazil
‡Department of Applied Computing – Federal University of Santa Maria – Santa Maria, Brazil
*{vitorcunha,dcorujo,jpbarraca,dgomes,ruilaa}@av.it.pt, †{mbcarvalho,alberto,granville}@inf.ufrgs.br, ‡{csantos}@inf.ufsm.br

***Abstract*—In this paper, we propose an architecture based on NFV and SDN which allows to balance traffic analysis techniques using a Classifier. It steers flows to the appropriate Service Function Chaining (to open traffic or not) according to network requirements (such as, effectiveness, flexibility, scalability, performance, and privacy). The SSL/TLS traffic processing is carried-out by the centerpiece of this work, the SFC-enabled MITM. A Proof-of-Concept was conducted (focusing on our SFC-enabled MITM) which showed that functionalities lost due to encryption (Content Optimization, Caching, Network Anti-virus, and Content Filter) were recovered when processing opened traffic within its Service Function Chains. We also evaluated its impact on performance. The results show that cipher suite overhead plays a role but can be mitigated, the Classifier can alleviate the performance overhead of different traffic analysis techniques, network functions have lower impact to performance, and Service Function Chaining length influences page load time.**

I. Introduction

Network Function Virtualization (NFV) is a novel network paradigm that relies on the virtualization of Network Functions (NFs) that formerly were provided by network devices, appliances, and middle-boxes. Virtualization itself brings benefits to management and operation of networks that include higher flexibility, easier scalability, and reduced CAPEX/OPEX. For instance, OPEX is reduced by the lower cost of more efficient software-based NFV networks (that can scale on-demand from the resource pool, saving on power and local hardware over-provisioning) and are easier to manage (compared to hardware-based traditional networks that are inflexible in management interfaces). In its turn, CAPEX is reduced by adoption of cheaper Common-of-the-Shelf (COTS) hardware, that contributes to a manageable shared resource pool, in substitution of dedicated commercial middle-boxes.

A functionality offered by commercial middle-boxes is SSL/TLS Inspection (also known as Man-in-the-Middle (MITM)), which opens encrypted traffic for processing (*Intrusive analysis*). This is also required because encryption breaks common NFs (*e.g.*, Cache Proxies, Content Optimizers, Content Filters) that rely on content to work. However, a MITM violates the security (and privacy) that endpoints attempted to enforce. In order to avoid this, *Non-intrusive analysis* (statistical [1] and behavioral approaches [2]) were developed. Despite these approaches preserve end-to-end encryption (thus, preserving security and privacy), they are less effective because of a degree of failure in their content guessing algorithms.

The adoption of middle-boxes to process encrypted traffic imposes low flexibility, low scalability, higher CAPEX/OPEX, and less transparency over what is done with the opened traffic. In turn, statistical and behavioral approaches are less effective which hampers network management needs. In order to face these challenges, enterprise network administrators have the need to balance effectiveness, flexibility, scalability, privacy, and performance (since Intrusive analysis can affect flow performance) when choosing how to handle encrypted traffic. They are in demand for a way to dynamically apply the most suitable techniques, always complying with legal obligations over that traffic (for instance, informed consent) and handling sensitive information in an ethical way, but without neglecting their own security and network management needs.

This paper proposes an architecture based on NFV and Software Defined Networking (SDN) to balance effectiveness, flexibility, scalability, privacy, and performance when processing encrypted traffic. These requirements are fulfilled steering encrypted traffic using a Classifier [3] (that can be policy-driven) to determine which kind of processing chain must be applied (*Intrusive* or *Non-intrusive*). As an alternative to MITM functionality of commercial middle-boxes, we aim to leverage the effectiveness and flexibility of a Service Function Chaining (SFC) via a SFC-enabled MITM (SFC-MITM), which is the centerpiece of our architecture.

The contributions of our proposal include: *(i)* flexibility to adjust the encrypted traffic processing in regards to both security and lawfulness, as well as to network management needs, *(ii)* ability to process encrypted traffic availing all NFV benefits, *(iii)* ability to apply processing functions, readily available as Virtual Network Functions (VNFs), over opened encrypted traffic, *(iv)* threat prevention and policy enforcement over content at network level (complementing endpoint solutions), and *(v)* network-enforced ability to upgrade cipher suite support in Legacy Appliances, having the capability to select different ciphers to interact with the appliance from the ones that are used for the remote endpoint.

We conducted a Proof-of-Concept (PoC) focusing on the SFC-MITM component, showing that functionalities (Con-

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

tent Optimization, Caching, Network Anti-virus, and Content Filter) are recovered when processing opened traffic within the SFCs. We also evaluated performance of SFC-MITM comparing it in regards to different cipher suites, traffic analysis techniques, functionalities of NFs, and SFC lengths. The results show that cipher suite overhead can be mitigated, the need of the Classifier due to different performance overhead of traffic analysis techniques, low impact of functionalities, and significant impact of SFC length in regards to load time.

The paper is organized as follows. In Section II, we present background information and related work. In Section III, we present the architecture of our proposal. In Section IV, we describe the scenario and objectives of the proposal evaluation along with obtained results. Finally, in Section V, we provide final remarks and future steps in the context of this work.

II. Related Work

We start this background section by presenting the *Intrusive Analysis* methods, since it is the highlight of our contribution, followed by the presentation of secure middle-boxes that go beyond SSL/TLS Inspection enforcing additional constraints over opened traffic. These should not be seen only as prior-art against our SFC-MITM solution, but rather as complementary functions that can easily be deployed in our architecture to achieve stricter privacy requirements. Finally, we present NFV and SDN characteristics that are pertinent in the context of encrypted traffic processing.

The industry's major players already offer commercial solutions that realise MITM in their midle-boxes[1][2][3], with wide acceptance within corporate networks. However, these solutions have low flexibility in the sense that they introduce general traffic processing functions. Furthermore, these solutions are mostly available as a closed-source appliance which impairs any kind of audit to assess how the opened traffic is processed.

The Open-Source community delivers some good tools for this kind of MITM. For instance, there is *mitm-proxy*[4] which provides MITM functionality via an HTTP proxy. It features easy *Python* scripting (add-ons) for programmatic flow processing. However, it does not provide any means to process flows externally of the program's code (*i.e.* in a VNF). A similar tool called *sslsplit*[5] also provides MITM functionality. Unlike *mitm-proxy*, it supports generic SSL/TLS over TCP (not just HTTPS) acting as an HTTP/Socks proxy, with Socks being required for protocols other than HTTPS. However, its main drawback is the lack of add-on capabilities. The precursor *sslstrip*[6] is widely used in traffic capture programs to inspect HTTPS. It hijacks HTTP responses exchanging explicit links to HTTPS sites by their HTTP versions. However, this downgrade/striping attack is now ineffective given most enterprises no longer provide a HTTP version of their sites.

[1]https://www.cisco.com/c/en/us/solutions/enterprise-networks/enterprise-network-security/eta.html

[2]https://f5.com/products/security/ssl-visibility

[3]https://www.checkpoint.com/products/url-filtering-software-blade/

[4]https://mitmproxy.org/

[5]https://www.roe.ch/SSLsplit

[6]https://moxie.org/software/sslstrip/

There are also proposals of secure middle-boxes which aim to take special care about protecting traffic opened by SSL Inspection, such as BlindBox [4] or SGX-Box [5]. The general approach is the creation of a secure enclave to process opened traffic, preventing against tampering or data leakage. SGX-Box, for instance, provides a specific language to allow the development of traffic processing code. However, because of the data protection, none of these solutions considered the delivery of flows for processing in external VNFs. mcTLS [6] attempts to solve this issue by extending TLS with multiple decryption keys. Nevertheless, the non-standard nature of these solutions hinders the ability to use today's Off-the-Shelf VNFs (such as, Content Filters/Optimizers or Anti-virus), as these functions would need either be rewritten in the proposed middle-box language or support the proposed TLS extension.

Recently, new network paradigms are breaking the former inflexible network architectures, leveraging advances in traffic processing. As an example of such paradigms, SDN decouples the data-plane from the control-plane, effectively standardizing means for flow-based programmatic central control of whole network topologies [7]. As another example, the NFV paradigm allows the replacement of former physical network components by virtual ones that perform the same functionalities. Making use of virtualization, and the flexibility given by virtual network topologies, we can now have dynamic deployment, replacement, and scalability of NFs. Leveraging SDN and NFV, SFC [8] allows to build Network Services (NS) by chaining in a given order existing NFs, enabling the implementation of Service Function Paths (SFPs).

III. Solution Architecture

Our work assumes a future enterprise network, that is SDN controlled and NFV capable. In addition, it also features on-demand NF instantiation and the ability to perform *Traffic Steering*/SFC amongst NFs. We start by presenting a conceptual architecture for future enterprise networks that balances intrusiveness, flexibility, effectiveness, privacy, and performance when processing encrypted traffic. Then, we zoom into the SFC-MITM (this paper's focus), presenting and describing its own architecture, along with the challenges that such component must tackle.

A. Network Architecture

The data-plane we propose for dealing with SSL/TLS encrypted traffic is depicted in Fig. 1, which is comprised of SFCs that fall under two very distinct types of analysis (intrusive vs. non-intrusive). Each of these chains can be built with COTS NFs. The balance between privacy, performance, more effective security and restored functionality will be enforced by a policy-driven steering classifier which, according to the classification rules, will deliver each flow to a single type of chain (intrusive or non-intrusive).

If traffic is not eligible to be opened by intrusive approaches due to its requirements (*e.g.*, privacy, performance), it is steered to a *Non-Intrusive Analysis* SFC. In this case, it is processed respecting the privacy enforced by the endpoints,

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

which imposes the adoption of NFs based on statistical and behavioral techniques in order to guess the content of the communication. The Deep Packet Inpection (DPI) is an example of a Non-Intrusive function, since its inspection is done over the ciphertext (without tampering with the endpoints security).

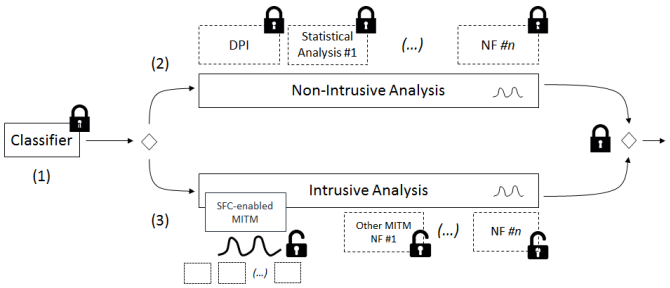

Fig. 1. Future Enterprise Network Architecture (Data-plane)

If traffic is eligible to be opened by intrusive approaches, it can be steered to a *Intrusive Analysis* SFC. In this case, the encrypted traffic is opened by intrusive NFs, such as secure middleboxes (*e.g.*, SGX-Box) or our novel *SFC-enabled MITM* (more on this follows in Section III-B).

Given that all network traffic must go through the classifier, this decision point becomes critical to the performance, scalability and reliability/resilience of the whole network. Therefore, in order to address these issues, the choice was made to not deviate our classifier from the ones in literature. However, this also means that we must select between intrusive (and non-intrusive) inspection using just header fields as classification criteria, as those are the better researched classifiers.

OpenFlow exposes a 12-tuple of header fields (`<ingress port, ether src, ether dst, ether type, vlan id, vlan prio, ip src, ip dst, ip proto, ip tos, port src, port dst>`) to set the policies which transparently steer traffic between intrusive and non-intrusive analysis. While non-SDN solutions are also possible, the number of fields available for classification as well as the ways to solve the performance, scalability and reliability/resilience issues become entirely dependent on the technologies supported by the equipment itself.

Conceptually, using just header fields to classify traffic does not affect the granularity of our solution, as we can enhance this coarse classification with the outcomes of our processing chains. For instance, we can have a DPI function in a non-intrusive chain analyze the Common Name (CN) of the server certificate. As browsers expect this field to have the matching Fully Qualified Domain Name (FQDN) of the destination server or application, this allows to distinguish communications which otherwise would appear to belong to the same entity but whose control over content is in fact quite different (*ie.* an IP from a CDN is a quite fallible way to determine the content being served). Similarly, one could use the intrusive analysis to make more informed decisions about the kind of optimizations that content must fore-go.

The reclassification between intrusive and non-intrusive analysis poses a challenge, as the SSL/TLS session was first established with the wrong endpoint (either the actual remote host or a MITM-alike function). Despite SSL/TLS having built-in session restart methods, those are not applicable when crossing security domains (we do not control the remote private keys), therefore we have no means to perform a smooth handover. We address this by performing a connection reset in the transport layer, which causes the flow to close (having then the user to repeat the request). Once the new request happens, the new classification can be applied.

B. SFC-enabled MITM

The *SFC-enabled MITM* must provide three major functionalities: *(i)* split the SSL/TLS session in two, one facing the client and the other facing the remote server, having in the middle the plain text; *(ii)* handle the delivery and the return of traffic to be processed by its SFCs, tracking the respective SSL/TLS session in which requests must be tunneled once again; *(iii)* perform Layer-7 specific sanitations to circumvent any tamper-detection enforced by endpoints, or that would result in an invalid configuration when sent to another NF.

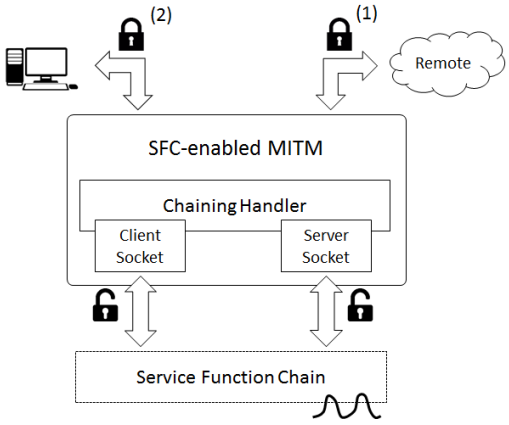

Fig. 2. SFC-enabled MITM Architecture (Data-plane)

The general architecture for this MITM is presented in Fig. 2, where the *Chaining Handler* communicates with the SFC over regular sockets (configured with a transport protocol according to the intercepted application). One must note that, in padlocks 1 and 2, the first refers to a SSL/TLS session negotiated using the remote PKI (acting the MITM as a client, on behalf of the user) and the second is a SSL/TLS session established using certificates issued by our own CA (which the user must trust). While conceptually it holds true that such an architecture would be able to handle any application that uses SSL/TLS, as the MITM itself handles that security layer then passing any of the tunneled data in plain text to the processing NFs, reality shows that some application protocols (such as HTTPS) already have measures that allow the endpoints to detect (and prevent) this kind of attack.

As a result, and given the rise of adoption of HTTPS on the web, we focus on this protocol. Modern browsers have support for HTTP Public Key Pinning (HPKP), which allows any website to use headers to set the expected public key of any

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

certificate in the Trust Chain (therefore invalidating a MITM-alike attack). Some known services may even have their public keys pre-baked into the browser (*i.e.*, Google Chrome and Google services). Nevertheless, as we are within an enterprise context, we have full authority over the terminals. This means that we can disable pre-baked lists using our own version of the browser (similarly to Opera Turbo) and, without disabling HPKP, we can replace these headers with our public key(s) in the MITM therefore subverting this protocol to only allow our devices to connect to those sites via our network (otherwise SSL/TLS will fail). Additionally, there are *integrity* fields that can be appended to anchor tags, which are meant to provide tamper resistance against changes to the referenced content. In order to change that content (for instance, with a Content Optimizer), we must either remove these tags altogether or update them accordingly. Other NFs (such as Content Caches) add identifying headers of the proxy software and requesting host, which we must remove if we want to conceal this fact to the remote endpoint. Lastly, HTTP features Transfer-Encoding and Content-Length headers which are meant to inform the endpoints of how to decode the payload. Yet, because our MITM actually decodes the content on behalf of the client/server, we must make sure to correct these headers before sending them to the NFs.

The SFCs may be achieved through different means. While (in the example of HTTP) one could build a proxy chain using upstream proxies, we are proposing to use instead Traffic Steering. This has the advantage of not requiring to go down to the application level to reconfigure functions on-the-fly (to accommodate a chain change), but instead just seamlessly steer traffic according to a classification, which should alleviate the orchestration efforts.

IV. Evaluation

To evaluate our architecture, we developed a PoC using *mitmproxy* (version 2.0.2) with an addon script which implemented the Chaining Handler. We used Python's *http* module to implement the server/client SFC sockets, allowing for parallel requests via a synchronized pool class that handles four threads (per flow), two for the callbacks of the split SSL/TLS tunnel (one for client, another for server) and two more for the SFC handling of the flow. Linux Policy Based Routing (along with netfilter marking and ARP table mangling) were used as replacement to the OpenFlow-alike *Flow Mods*.

All tests were conducted in a shared Proxmox Virtual Environment 4.4-1/eb2d6f1e, using 4 VMs. One VM to simulate the architecture (with 8 cores and 8GiB of RAM) where all NFs and MITM are instantiated as LXD containers. One VM (with 2 cores and 2GiB of RAM) was used to act as the user. Another VM (with 4 cores and 2GiB of RAM) was used as webserver. Lastly a VM (with 2 cores and 512MiB of RAM) was used to act as Classifier and DPI. The vCPUs are slices of an Intel(R) Xeon(R) CPU X5670, with AES-NI enabled.

The webserver is an Apache 2.4.18 hosting static pages built to check both the architecture performance and the NFs functionalities when processing traffic within the SFC. These pages were built referring to a fixed number of JPEG images (5, 10, 15, 20, 25, and 30) retrieved from INRIA Holidays dataset [9] selecting the 30 smaller ones that range from 83 KiBytes to 483 KiBytes. However, preliminary observations showed these pages provided similar conclusions. Therefore, we continue the evaluations using the 5 JPEG images page.

The DPI is a fork of *ndpi-netfilter*[7], running in an Ubuntu 14.04 LTS. It tries to identify Facebook (by certificate CN), Skype (a sequence of given packet lengths + byte pattern(s) at given offset(s)), Dropbox (combination of protocol, ports and a substring within the payload), and general SSL/TLS on all TCP flows (which analyses packet sequences along with mixed binary and textual patterns to infer information from the handshake), regardless of port. When verdicts overlap, it returns the most specialized label (for instance, despite being SSL/TLS, Facebook receives just its own tag).

Along the evaluation, we collected measurements in regards to Round-Trip-Time (RTT), load time, CPU consumption, and throughput (these parameters are important to measure the user experience). RTT was measured using the *httping* tool, which measures the elapsed time from making a request (HEAD method) to obtaining the response (headers). Load time was measured using a web client simulated with PhantonJS[8] that parses the static pages, detailed previously, downloading the required images and simulating their rendering like an end-user web-browser (which benchmarking tools such as *ApacheBench* or common tools like *curl* or *wget* do not perform). CPU consumption was measured collecting the clock ticks used by the MITM process, which is provided by the `/proc/PID/stat` file. Throughput was measured by downloading an incompressible 10 MiB file, using *wget*. The results show means and confidence intervals (95%) for 30 repetitions of each observation.

We start by evaluating how our proposed architecture behaves, focusing on the performance impact the MITM has and how it compares to non-intrusive methods. Then, we evaluate a simple case-study in which functionality is restored (through use of previous NFs that worked with unencrypted traffic).

A. Architecture evaluation

We start by evaluating the cryptographic influence over the architecture in regards to load time, comparing Direct Access (DA) (client to web server), Stock MITM (S-MITM) (client to web server through the original version of mitmproxy), and SFC-MITM (client to web server through the mitmproxy with our SFC handling). We also compare the cryptographic influence in regards to CPU consumption between the unmodified ("Stock") MITM and SFC-enabled MITM. Our baseline for the evaluation were the cipher suites available by default in modern Linux distributions (Ubuntu 16.04 and Debian 9). These measurements were taken using no cryptography (HTTP), TLSv1.1 or TLSv1.2 (as supported by the respective cipher suite). Table I presents the results that are assessed next.

[7]https://github.com/betolj/ndpi-netfilter

[8]http://phantomjs.org/

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

TABLE I
CIPHER SUITE IMPACT ON MITM PERFORMANCE

	Direct Access	Stock mitmproxy		SFC-enabled MITM	
TLS version / Cipher Suite	**load time [ms]**	**load time [ms]**	**CPU [ticks]**	**load time [ms]**	**CPU [ticks]**
NONE / UNENCRYPTED	24.56 ± 0.82	171.04 ± 3.80	19.46 ± 0.57	263.94 ± 4.83	26.65 ± 0.54
TLSv1.1					
DHE_RSA_AES_256_CBC_SHA	75.50 ± 1.46	389.45 ± 10.40	46.97 ± 1.39	607.58 ± 10.20	56.66 ± 1.12
DHE_RSA_AES_128_CBC_SHA	74.29 ± 1.14	403.48 ± 7.62	48.94 ± 1.55	595.90 ± 10.91	55.72 ± 1.12
RSA_AES_256_CBC_SHA	46.91 ± 1.24	352.27 ± 8.62	42.21 ± 1.36	436.51 ± 8.22	48.32 ± 1.13
RSA_AES_128_CBC_SHA	44.91 ± 0.98	365.29 ± 9.84	44.50 ± 1.58	431.32 ± 6.75	47.79 ± 1.10
ECDHE_RSA_AES_128_CBC_SHA	46.40 ± 0.87	362.58 ± 7.97	44.29 ± 1.32	445.62 ± 9.01	48.88 ± 1.11
ECDHE_RSA_AES_256_CBC_SHA	46.66 ± 1.19	359.85 ± 9.88	43.57 ± 1.36	438.85 ± 6.85	48.74 ± 1.02
TLSv1.2					
DHE_RSA_AES_256_GCM_SHA384	73.79 ± 1.26	402.45 ± 9.58	48.53 ± 1.20	597.55 ± 10.99	56.43 ± 1.38
DHE_RSA_AES_128_GCM_SHA256	72.03 ± 1.31	398.97 ± 13.99	48.16 ± 2.01	597.70 ± 10.45	56.54 ± 1.37
DHE_RSA_AES_256_CBC_SHA256	80.41 ± 1.84	407.22 ± 8.44	49.79 ± 1.45	609.77 ± 9.92	57.75 ± 1.08
DHE_RSA_AES_128_CBC_SHA256	81.53 ± 1.60	407.48 ± 12.18	50.00 ± 2.09	614.02 ± 10.70	58.32 ± 1.16
RSA_AES_256_GCM_SHA384	44.24 ± 1.00	357.97 ± 6.42	42.94 ± 1.15	433.94 ± 9.61	47.98 ± 1.37
RSA_AES_128_GCM_SHA256	42.79 ± 0.84	357.64 ± 10.73	43.06 ± 1.71	433.06 ± 7.43	47.81 ± 1.05
RSA_AES_256_CBC_SHA256	48.77 ± 0.95	356.09 ± 13.43	42.85 ± 1.79	425.45 ± 8.43	46.73 ± 1.44
RSA_AES_128_CBC_SHA256	49.88 ± 1.37	360.97 ± 8.19	43.58 ± 1.25	436.00 ± 9.21	47.96 ± 1.28
ECDHE_RSA_AES_256_GCM_SHA384	46.32 ± 2.56	365.65 ± 9.22	44.26 ± 1.34	437.81 ± 10.05	48.42 ± 1.49
ECDHE_RSA_AES_128_GCM_SHA256	43.45 ± 0.78	361.75 ± 7.48	43.27 ± 1.37	427.02 ± 8.18	47.04 ± 1.33
ECDHE_RSA_AES_256_CBC_SHA384	51.44 ± 1.01	370.91 ± 12.18	45.45 ± 1.75	446.27 ± 10.12	48.77 ± 1.09
ECDHE_RSA_AES_128_CBC_SHA256	52.26 ± 2.18	360.33 ± 11.61	44.26 ± 1.61	429.47 ± 8.00	47.12 ± 1.31

Having already separated TLS version, we must now breakdown cipher suite evaluation according to its four components: Key Exchange/Agreement, Authentication, bulk-data Encryption, and Integrity. First, we consider Diffie-Hellman (DH) + RSA, Elliptic-Curve Diffie-Hellman (ECDH) + RSA and just RSA for Key Exchange/Agreement. As expected, DH + RSA was far slower than just RSA (when performing a direct access, $\approx 30ms$ slower, which is $\approx +90\%$), being ECDH + RSA only slightly slower (on average) than using just RSA (negligible difference as they are very near each others' interval). However, in relative terms, the difference becomes less pronounced when using a MITM. The stock *mitmproxy* has a penalty of about $\approx 50ms$ ($\approx +10\%$) for DH + RSA vs. just RSA, while our version with the SFC addon has a penalty of about $\approx 150ms$ ($\approx +33\%$) for the same comparison. These relative differences between MITMs are justifiable by the contribution that (different) internal traffic processing has over the results, as we can see when comparing against the unencrypted control, which shows that (in our worst case) adding DH + RSA has a similar relative penalty (over unencrypted) in both MITM, $\approx (225 - 230)\%$.

As for bulk-data encryption, we compare AES with two block sizes (128 and 256 bits) along with two cipher modes (CBC and GCM). However, given the available cipher suites tend to force different digests for different block sizes, we must take care when analyzing these results. We observed that performance is similar between AES 128 and 256 bits (when using same digests), having GCM outperformed CBC across the board. As for digests, we observed that going from SHA256 to SHA384 (while also increasing block size from 128 to 256 bits) yields results within the margin of each other, therefore the impact in this scope is negligible. Given the similarity in performance between just RSA and ECDH, similar performance with both block sizes, and GCM cipher mode performing faster than CBC. For further evaluation of our SFC-MITM, we used the *ECDHE_RSA_AES_256_GCM_SHA384* cipher suite as it provides the best compromise between highest security and best performance.

Regarding CPU time, we can see that going from unencrypted to encrypted in both MITM versions has an overhead of $\approx 25 - 30$ ticks. Comparing SFC-MITM to S-MITM, we can observe a rise of ≈ 10 ticks for the most computationally demanding ciphers suites, while less demanding ones have a smaller rise (≈ 3 ticks). The introduction of SFC support in the MITM has some impact in CPU consumption, but it can be mitigated with appropriated choice of cipher suite.

We also evaluated the influence in regards to RTT, load time, and throughput of different traffic analysis techniques comparing DA (no traffic analysis), DPI (Non-Intrusive analysis), S-MITM (Intrusive analysis: opened traffic is processed within mitmproxy), and SFC-MITM (Intrusive analysis: opened traffic is processed by a SFC without NFs). Table II, Traffic Analysis shows the collected results.

Comparing DA to DPI, we can observe their confidence intervals overlap in all measurements both for Plain Text and for the chosen cipher suite. Thus, we can conclude that statistically they have the same performance. In regards to RTT and load time, we can observe that SFC-MITM imposes a rise comparable both to DA and DPI and to unencrypted and encrypted content, but we can also observe that most of this rise is due to the introduction of a MITM and cryptography. Thus, we can conclude that most of the impact in regards to RTT and load time comes from the MITM approach through mitmproxy (with an added penalty for SFC) and adoption of encryption. In regards to throughput, most confidence intervals overlap, except SFC-MITM when processing encrypted traffic.

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

TABLE II
OVERHEAD INFLICTED BY INTRUSIVENESS AND NETWORK FUNCTIONS IN CHAIN

	Plain Text			TLSv1.2 *(ECDHE_RSA_AES_256_GCM_SHA384)*		
Traffic Analysis	**rtt (ms)**	**load time (ms)**	**throughput (MiB/s)**	**rtt (ms)**	**load time (ms)**	**throughput (MiB/s)**
Direct Access	6.49 ± 0.21	24.56 ± 0.82	111.66 ± 5.45	6.51 ± 0.21	43.66 ± 1.00	104.53 ± 6.62
Non-Intrusive (DPI)	6.87 ± 0.22	25.48 ± 0.79	111.19 ± 4.81	6.39 ± 0.17	44.34 ± 0.92	99.02 ± 6.52
Stock *mitmproxy*	13.40 ± 0.44	171.04 ± 3.80	114.19 ± 5.85	36.76 ± 2.22	357.21 ± 8.10	96.94 ± 4.92
SFC-enabled MITM (0 NFs)	20.18 ± 0.57	263.94 ± 4.83	113.77 ± 5.24	45.71 ± 1.53	421.43 ± 7.81	82.78 ± 2.51
Intrusive NFs						
Anti-Virus (HAVP)	20.76 ± 0.56	279.60 ± 6.33	107.41 ± 3.88	45.24 ± 0.97	484.83 ± 8.22	80.32 ± 2.25
Content Cache (squid)	13.70 ± 0.45	218.88 ± 3.74	105.55 ± 6.20	39.61 ± 0.58	309.96 ± 5.92	81.94 ± 2.34
Content Filter (e2guardian)	21.99 ± 0.72	268.79 ± 4.90	108.72 ± 3.57	46.53 ± 1.26	440.56 ± 8.10	80.22 ± 1.95
Content Optimizer (ziproxy)	20.80 ± 0.42	296.72 ± 6.22	114.49 ± 5.29	46.21 ± 1.21	459.13 ± 7.29	83.92 ± 1.93
Both CC+CO (squid+ziproxy)	15.92 ± 0.60	184.53 ± 4.97	103.98 ± 4.33	41.37 ± 0.84	271.83 ± 7.09	74.28 ± 2.53
All of above in chain	14.61 ± 0.64	210.23 ± 4.93	112.84 ± 4.74	45.05 ± 1.53	295.11 ± 10.86	78.29 ± 2.35
SFC length *(with ziproxy)*						
SFC-enabled MITM (1 NFs)	20.98 ± 0.59	299.24 ± 6.08	112.64 ± 5.22	43.55 ± 0.74	453.35 ± 7.76	84.01 ± 2.34
SFC-enabled MITM (2 NFs)	21.06 ± 0.34	324.94 ± 5.45	111.62 ± 4.63	48.21 ± 1.36	476.00 ± 7.71	83.77 ± 2.70
SFC-enabled MITM (3 NFs)	24.82 ± 0.99	350.39 ± 8.53	113.56 ± 4.34	45.95 ± 0.90	515.08 ± 9.74	82.44 ± 1.91
SFC-enabled MITM (4 NFs)	23.56 ± 0.65	363.89 ± 7.51	109.35 ± 3.74	51.14 ± 2.19	546.91 ± 9.31	83.74 ± 2.78
SFC-enabled MITM (5 NFs)	25.22 ± 0.77	391.90 ± 8.20	109.30 ± 5.21	51.82 ± 1.80	577.41 ± 10.81	81.06 ± 2.31
SFC-enabled MITM (6 NFs)	25.17 ± 0.51	426.66 ± 6.95	108.06 ± 4.07	50.73 ± 1.87	605.98 ± 8.84	81.42 ± 2.72
SFC-enabled MITM (7 NFs)	28.21 ± 0.81	445.30 ± 8.26	112.79 ± 3.86	50.96 ± 1.44	625.52 ± 12.13	84.64 ± 2.89
SFC-enabled MITM (8 NFs)	26.99 ± 0.56	475.94 ± 6.15	111.07 ± 3.97	51.72 ± 1.27	663.68 ± 11.84	82.83 ± 2.86
SFC-enabled MITM (9 NFs)	28.01 ± 0.56	504.00 ± 12.20	109.81 ± 4.48	54.81 ± 1.69	677.40 ± 12.75	81.40 ± 2.27
SFC-enabled MITM (10 NFs)	29.61 ± 0.65	522.68 ± 7.78	112.64 ± 3.31	53.16 ± 1.01	718.69 ± 13.11	83.67 ± 3.02

Thus, we can conclude that introducing SFC support into mitmproxy had an impact in regards to throughput. These results confirm our assumption that, performance wise, the Classifier is important to choose the appropriated analysis (Intrusive or not) to best fit the performance requirements.

We evaluated the impact of each function type (Table II, Intrusive NFs). Overall, the results show that all NFs have a similar impact in our system, being most results (RTT, load time, and throughput) within the confidence interval of each other. One exception is when Content Cache (squid) is in the SFC because its cache functionality improves all measurements. Other exception is Content Optimizer (ziproxy) that achieved the best results in regards to throughput.

We evaluated the impact of the SFC length (Table II, SFC length) varying its length from 1 to 10 NFs. In regards to RTT, we can observe an increase of $\approx$ 1ms per NF both for Plain Text and for the chosen cipher suite. In regards to load time, we can observe a higher rise ($\approx 25 - 30$ms) per NF accumulating $\approx$ 270ms when the 10 NFs are in place. In regards to throughput, we can observe that all measurements overlap both in Plain Text and in chosen cipher suite groups. In this sense, we can conclude that the throughput is not affected by the SFC length, but by the computational overheads (manifested with cryptography).

Our experiments exhibit an expected impact of performance introduced by MITM approaches, which emphasize that NFV can be a interesting enabler of MITM approaches due to easier scalability achieved by the paradigm. These observations also show that the SFC approach is appropriated to handle traffic opened by the SFC-MITM through NFs. It must be noted the required care about the SFC length in regards to load time.

B. Functionality evaluation - Case Study

We considered an enterprise which has remote workers connected through constrained connections which require optimization to communicate with other workers/access content in the main office. Bring-Your-Own-Devices (BYODs), sophisticated malware, and disgruntled employees may compromise endpoint security, therefore displacing security enforcement to the network is a requirement of our scenario. Throughout the enterprise network, it is prohibited access to inappropriate content, which requires filtering capabilities in the network to block such content. Since ISPs started adopting data caps, the enterprise requires caching the content to reduce its Internet bill. Data caps also apply to the connection to remote offices enforcing the optimization requirement in that network link.

As our architecture allows for any existing NF to process traffic, we used well established open source solutions to evaluate these functionalities: a Content Optimizer (ziproxy); a Content Filter (e2guardian); an Anti-Virus (HAVP); and a Content Cache (squid). However, with the rise of adoption of encryption, their functionalities "out-of-the-box" are hindered because they rely on the content to work properly. We decided to check whether their functionalities could be recovered by the SFC-MITM with these solutions acting as NFs. Although both e2guardian and squid could act as their own MITM, for the sake of manageability, we chose to use only our MITM to perform this task. Other NFs would not work properly without the SFC-MITM delivering the Plain Text to them.

The functionalities provided by the Anti-virus (HAVP) and Content Filter (e2guardian) were checked through individual tests giving to them the encrypted traffic or the traffic opened

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

TABLE III
REMOTE OFFICE AND DATA CAPS OPTIMIZATION SCENARIOS *(TLSv1.2 ECDHE_RSA_AES_256_GCM_SHA384)*

	Data to Remote Office *(Client)* [KiB]				**Data from Internet *(Router)* [KiB]**				**Page Load Time *(Client)* [ms]**			
#Req	**squid**	**ziproxy**	**squid+ziproxy**	**SFC-MITM**	**squid**	**ziproxy**	**squid+ziproxy**	**SFC-MITM**	**squid**	**ziproxy**	**squid+ziproxy**	**SFC-MITM**
1	1491.60	148.16	148.87	1491.40	1514.32	1514.32	1515.28	1513.87	446	480	454	430
2	1493.26	148.11	148.44	1491.61	0	1514.78	0	1510.64	302	445	251	433
3	1505.84	148.31	148.65	1491.61	0	1499.46	0	1519.49	300	534	300	423
4	1492.55	148.21	148.55	1491.55	0	1526.71	0	1505.78	301	425	252	440
5	1495.46	148.21	148.55	1491.35	0	1509.53	0	1503.10	316	462	250	453

by the SFC-MITM within the SFC. In this case, Anti-Virus successfully detected the "virus" signature of the eicar test file and the Content Filter blocked pages containing inappropriate words to enterprise environments. This evaluation shows that their functionalities were recovered when acting as NFs within an SFC to process traffic opened by the SFC-MITM. Thus, they can be useful again for enterprise's networks.

The functionalities provided by the Content Optimizer (ziproxy) and Content Cache (squid) were checked through a sequence of 30 requests to the same static page, detailed previously. The squid cache was emptied before each iteration. Table III shows the received data on the client (Remote Office Optimization) and on the Router from the Internet (Data Caps Optimization). We also combined an SFC with both NFs (squid+ziproxy). The SFC-MITM values are provided as a control because it reflects a chain without NFs. The values are referred to the initial 5 requests to the static page with 5 images because the next requests show the same pattern.

Remote Office Optimization is in charge of the Content Optimizer (ziproxy). Table III shows that ziproxy reduces the amount of traffic received in the client from ≈ 1491KiB to ≈ 148KiB. It was configured to convert images to gray scale ones. Although this configuration sounds unrealistic, it visually exposes, as intended, that its functionality was recovered.

Data Caps Optimization is in charge of the Content Cache (squid). Table III shows that squid reduces the amount of traffic received on the Router from the web server from ≈ 1519KiB to 0 between the first request (clean cache) and the second one. These results may be exaggerated due to the static nature of the pages evaluated. Even though nowadays most pages are dynamic and only certain assets may be cached, this still exposes that cache functionality was recovered inside the SFC, as intended, allowing further study, such as leveraging this as a local cache to serve content when the mobile link is dropped.

The functional evaluation showed that SFC-MITM empowered with functionalities provided by the NFs can be appropriated for achieving enterprise network requirements.

V. CONCLUSION

It was shown that the lost functionality taken by the mass-adoption of encryption could be recovered using the proposed MITM and SFCs built from existing plain-text VNFs. However, as expected, the MITM method introduces a significant computational overhead, which we have addressed with the proposed network architecture. The architecture allows to make on-the-fly tradeoffs between privacy, performance, security, and functionality, by choosing the most appropriate intrusive or non-intrusive SFC to process that given flow. The results confirm that non-intrusive methods have a significant performance advantage, while intrusive methods make-up for their losses in restored functionality. The study shows that the architecture is promising, calling for further research of policy definition schemes and control-plane decisions for classification, and its relation with data security in NFV/NFs.

ACKNOWLEDGMENT

This work is supported by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project Smart EnterCom with Nr. 021949 (POCI-01-0247-FEDER-021949)] and by CAPES (Brazilian Federal Agency for Support and Evaluation of Graduate Education) within the Ministry of Education of Brazil, under the project FCT number 0409/2016 entitled "Um Ecossistema para Funções Virtualizadas de Rede".

REFERENCES

[1] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. New York, NY, USA: ACM, 2005, pp. 50–60.

[2] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security*, ser. AISec '16. New York, NY, USA: ACM, 2016, pp. 35–46.

[3] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," Internet Requests for Comments, RFC Editor, RFC 7665, October 2015.

[4] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "Blindbox: Deep packet inspection over encrypted traffic," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 213–226.

[5] J. Han, S. Kim, J. Ha, and D. Han, "Sgx-box: Enabling visibility on encrypted traffic using a secure middlebox module," in *Proceedings of the First Asia-Pacific Workshop on Networking*, ser. APNet'17. New York, NY, USA: ACM, 2017, pp. 99–105.

[6] D. Naylor, K. Schomp, M. Varvello, I. Leontiadis, J. Blackburn, D. R. López, K. Papagiannaki, P. Rodriguez, and P. Steenkiste, "Multi-context tls (mctls): Enabling secure in-network functionality in tls," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. New York, NY, USA: ACM, 2015, pp. 199–212.

[7] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426, Jan. 2015.

[8] D. Bhamare, R. Jain, M. Samaka, and A. Erbad, "A survey on service function chaining," *Journal of Network and Computer Applications*, vol. 75, pp. 138–155, 2016.

[9] M. D. Herve Jegou and C. Schmid, "Hamming embedding and weak geometry consistency for large scale image search," in *Proceedings of the 10th European conference on Computer vision*, October 2008.