SDN-based Trusted Path Control

Stéphane Betgé-Brezetz, Guy-Bertrand Kamga
Alcatel-Lucent Bell Labs
Nozay, France
Email: firstname.lastname@alcatel-lucent.com

Ali El Amrani Joutei, Oussama Maalmi
Telecom SudParis
Evry, France
Email: firstname.lastname@telecom-sudparis.eu

Abstract — **Security of sensitive data in the network is a key issue in a world where such sensitive data can easily be transferred between different servers and locations (e.g., in networked clouds). In this context, there is a particular need to control the path followed by the data when they move across the cloud (e.g., to avoid crossing -even encrypted- un-trusted nodes or areas). In this paper we proposed therefore a new approach which aims to leverage the programmability offered by the SDN technology in order to enforce a trusted path for the transfer of sensitive data in the network. Given a policy related to the sensitive data (e.g., the data should not cross a given area), our approach allows sending this policy to an extended SDN controller (called Trusted Path Controller) which automatically enforces this policy in the SDN network. Two architectures have been investigated: the Out-of-Band architecture (the policy being sent to the Trusted Path Controller via a Web Service interface) and the In-Band architecture (the policy being sent to the Trusted Path Controller via a dedicated "signaling packet"). These two architectures have been implemented in a SDN controller. Experimentations and evaluations have also been performed on a test-bed of SDN switches which allow showing the feasibility of this approach as well as its performances.**

Keywords — SDN network; trust; policy; path control

I. INTRODUCTION

Security of sensitive data in the network is a key issue in a world where such sensitive data can easily be transferred between different servers and locations. This is notably the case for the Cloud environment which offers the ability to provide IT and networking resources on demand; while requiring low effort for the customers to manage these resources. Nevertheless, the enterprises are still hesitant to put their sensitive data in such cloud infrastructures, even for a time-bound project, as they have fears about their security [1]. Moreover, sensitive data as Personally Identifiable Information (PII) are also subject to strong country-based regulatory constraints [2], notably dealing with their locations, and that may be an actual hurdle for companies or administrations to transfer and store these sensitive data in a cloud environment.

The problem of data storage location is then one of the major cloud security issues which is notably debated in the technical community as well as in the public sphere. Some technical solutions, even if not yet fully satisfactory, are however being proposed to control data storage location in order to be compliant to the related policies [3]. But, beyond the only storage location, there is also a need to control the path followed by the data when transferred in the cloud (i.e., either when firstly uploaded in the cloud or when transferred within the cloud between different storage entities). Indeed, solutions as communication protection (e.g., TLS/SSL, VPN) may not be considered as sufficient as they do not prevent an eyedropper to infer some information from the traffic done between the two extremities (e.g., two Virtual Machines). For instance, the monitoring of the level of traffic (even encrypted) between two cloud entities, for instance belonging to two different companies, can be used in order to infer the level of exchanges between these companies. Moreover, some Denial of Service (DoS) attacks can be performed on un-trusted or insufficiently secure nodes located on the path of the sensitive traffic (and then disturbing or blocking this sensitive traffic). We can also note that some regulations may impose direct constraints on the data transport (e.g., new European initiative on the "Schengen of data").

It may then be requested that the flow of sensitive data must cross the network infrastructure only in accordance to specific security or regulatory policies. For instance, a policy should state that the path followed by a sensitive data should not cross a given area or country. Also, other policies should state that the path should not cross an un-trusted node, network, cloud provider, telco, etc.

In order to tackle this problem of trusted path for sensitive data transfer, the emerging technology of Software Defined Networking (SDN) is of particular interest as it allows making the networks more programmable. Indeed, the principle of SDN is to remove the control plane from the network equipment and have it available as a software module called SDN controller. This SDN controller is a programmable entity which allows developing upon it various applications of network flow processing such as Firewall, Network Address Translation (NAT), Deep Packet Inspection (DPI), etc. This programmability offered by SDN (through the SDN controller) can then be exploited in order to dynamically configure a network path that satisfies the security policies related to the sensitive data to convey [4]. The objective of the work presented in this paper is then to propose a new network application (running upon a SDN controller) allowing us to automatically compute and establish such a trusted path compliant with the security policies of the data to transfer.

The paper is structured as follows. In Section 2, we analyze the related work and position our approach. Then section 3 introduces the general architecture and the proposed interfaces. Section 4 details our implementation, the SDN test-bed used for experimentations, and the obtained results as well as some recommendations. Finally, the conclusion summarizes the contributions and presents some perspectives.

II. RELATED WORK

Different works have already been performed on how to secure the SDN technology for instance by preventing DoS attacks through overloading SDN switches or attacks on SDN applications, or man in the middle attacks between the SDN controller and the switch [5][6]. However, the problem we address in this paper falls within the dual topic on how to use SDN in order to improve network security, and this problem has been less addressed at this stage. Related to this last topic, we can nevertheless mention research works exploiting SDN to better monitor and inspect the traffic for the detection of potential network attacks or malicious softwares [7], and then to reroute this hazardous traffic in a quarantine zone. Other approaches [8] use SDN in order to “slice” the network traffic which improves the user data security (each SDN controller being for instance only able to access to one slice but not to the others); but these approaches do not deal with the computation of a trusted path able to transport a sensitive traffic in the network. The use of SDN technologies for trusted path has in fact already been mentioned [4], but to our best knowledge, it has not really been proposed and carried out a solution to enforce a trusted path satisfying the security constraints of a particular sensitive data to transport in the network.

However, various works are tackling the use of SDN in order to establish a network path that satisfies some constraints of QoS (e.g., bandwidth, latency, jitter) or reliability (e.g., using disjoint multipath). These works cover the intra or inter domains [9][10] and can leverage classical path computation approaches such as RSVP, MPLS Traffic Engineering (MPLS-TE), or IETF Path Computation Element (IETF-PCE). Nevertheless, these approaches are not targeting the establishment of a trusted path satisfying some specific security constraints; even if some approaches mention this as a future direction. For instance, in the scope of the IETF-PCE working group, it is proposed an extension of the Path Computation Element Communication Protocol (PCEP) with the notion of “Explicit Route Exclusions” [11]. Therefore, the approach proposed in this paper consists in leveraging this type of work by using the SDN technology in order to support various security constraints as well as to enable a dynamic path configuration. This will indeed allow protecting the different types of sensitive data transported in the cloud infrastructure as well as supporting the dynamicity and elasticity of such cloud environments (e.g., ensuring the dynamic and per-flow path configuration for VMs migrating to different locations).

III. GENERAL ARCHITECTURE

In this section, we present the general architecture as well as the proposed messages to request the setting and unsetting of a trusted path satisfying some given security constraints.

A. *Architecture principles*

Fig.1 depicts the principle of our proposed architecture allowing to request and configure a Trusted Path (TP) in a SDN network. The core of this architecture is the proposed Trusted Path Controller (TPC) which is running on top of a SDN controller. Let then consider the transport of a sensitive data through the SDN network and between two applications running in two different servers/VMs (called Source and Destination server/VM in Fig.1). The transport of this particular sensitive data flow should comply with a given security policy (e.g., avoid un-trusted switches, areas, providers). Note that the application can be a business application of the cloud user or even a FTP server/client for instance to backup a sensitive file. Indeed, for a backup, it may be requested that a sensitive file (or set of sensitive files) must be transferred by FTP (or SFTP) through a path satisfying a given security constraint; while the other traffic (e.g., Web or HTTP) can use the default path which is then not necessarily satisfying this constraint.

A Trusted Path Agent (TPA) is then running in the Source server/VM in order to request the establishment of the TP for the transfer of the sensitive data flow. As shown Fig.1, the TPA can get the security policy when the application requests the transfer. For this purpose, different mechanisms can be used such as the one proposed in our previous work where data and policy are bundled in a structure called Privacy Data Envelope (PDE). In this case, if the application is executed in a Web browser (connected to a remote Web Server), the TPA can be a browser plug-in intercepting the HTTP messages transporting the PDE (policy being extracted from this PDE) [12]; if the application is a mailer, the TPA can be a mailer plug-in (policy being extracted from the mail containing the PDE) [12]; or if the application is a FTP server, the TPA can be a module using Linux FUSE and intercepting -within the OS- the FTP request made on the PDE file (policy being also extracted from it) [13].

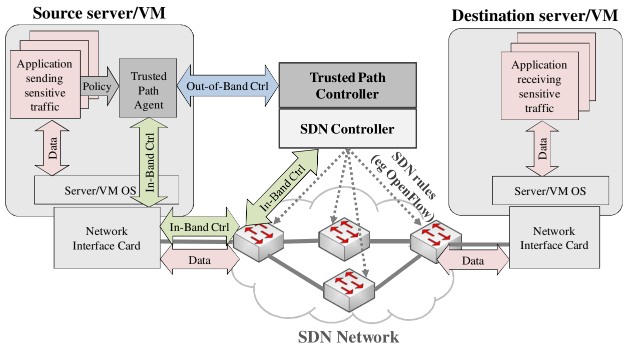


Fig.1. Overall architecture of the SDN-based trusted path control.

The TPA is then in charge to send the policy to the TPC (see Fig.2) which (i) computes a path satisfying the policy (see Policy-aware Path Computing module in Fig.2) and (ii) configures this path within the SDN network by using a SDN protocol as OpenFlow (see Policy-aware Path Configuration module in Fig.2). An acknowledgment is then sent back to the TPA to inform it that the request has been received and that the path is well configured or not (if not, some information is provided within this acknowledgment about the reason of the failure). Note this mechanism allows changing the policy for each file transfer if needed.

There are two possible architectures for the TPA to send the policy to the TPC (see Fig.1):

- *Out-of-Band architecture*: the TPA sends the policy on a Web Service interface (e.g., REST) exposed by the TPC. This Web Service interface is then in charge to trigger the Policy-aware Path Computing module (see Fig.2).

- *In-Band architecture*: the TPA forges and sends to the SDN network a dedicated IP packet ("signaling packet") containing the policy in its payload. When this packet crosses the first SDN switch of the network, it is forwarded to the SDN controller (whether because it does not correspond to any rule in the switch forwarding table, or because it corresponds to a dedicated rule identifying it as a request coming from a TPA). The SDN controller then extracts the policy from the signaling packet with the Policy Extractor module and triggers the Policy-aware Path Computing module (see Fig.2).

In the section IV, we will provide more details on these two architectures with their respective advantages and drawbacks.

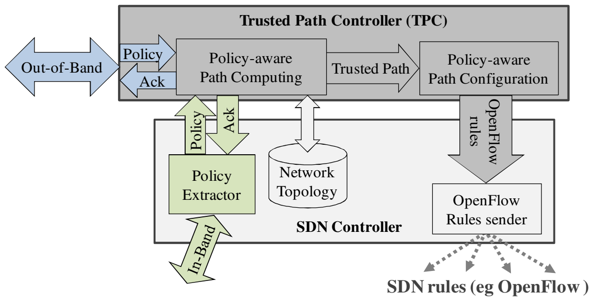


Fig.2. Trusted Path Controller. Arrow "Policy" (resp. "Ack") corresponds to the message *set-trusted-path* (resp. *ack-set-trusted-path*).

B. *Communication to the Trusted Path Controller*

The application (or an administrator) communicates, through the TPA, to the TPC thanks to different messages. They allow setting (and unsetting) a TP and getting the response back. These messages, both applicable for the Out-of-Band and In-Band architectures, are the following:

- ***set_trusted_path***: request to configure a TP satisfying a given policy.
- ***ack_set_trusted_path***: response to the *set_trusted_path* request. In this response, it is specified if the TP has been set or not (and the reason).
- ***unset_trusted_path***: request to release a TP.
- ***ack_unset_trusted_*path**: response to the *unset_trusted_path* request. In this response, it is specified if the TP has been unset or not (and the reason).

Moreover, the previous message *set_trusted_path* (to request a TP) has the following parameters:

- ***request_id***: identifier of the request. It is incremented each time a *set_trusted_path* the TPC receives a request.
- ***destination_IP***: IP address of the destination of the TP.
- ***port_number***: port number of the destination of the TP (for a TCP/UDP traffic).
- ***excluded_switches***: list of excluded switches given by their characteristics as follows:
  - *switch_id*: identifier of the switch given by the controller.
  - *manufacturer*: company name of the switch manufacturer. All the switches for which the MAC address is within the Organizationally Unique Identifier (OUI) of this manufacturer will be excluded.
  - *type*: Type of the switch (e.g., PHYSICAL_SWITCH, VIRTUAL_SWITCH).
  - *OS_version*: OS version of the switch.
- ***excluded_links***: list of excluded links given by their characteristics as follows:
  - *link_id*: link defined with the switch_id of the two switchs at the extremity of the link.
- ***telcos_id***: identifier of the telco.
- ***location***: location of the switch that can be a country (e.g., FRANCE, USA) or any other geographical entity.

IV. PROTOTYPE AND EVALUATIONS

This section presents the prototype implementing our approach, its experimentations performed on a test-bed of SDN/OpenFlow switches, and the obtained evaluation results (notably regarding the Out-of-Band and In-Band architectures).

A. *Prototype and SDN test-bed description*

The prototype is performed on the Floodlight SDN controller (version 0.90) which has been extended with our proposed TPC module. This module, developed in java, takes as inputs (i) the policy defined in the previous section III.B and (ii) the Network Topology available in the SDN controller (see Fig.2). It computes then a TP compatible with this policy by using a shortest path algorithm performed over the network topology which has been filtered according to the requested policy (e.g., avoid given un-trusted nodes or links). The TP is then sent to the Policy-aware Path Configuration module which is in charge to establish the different SDN rules that need to be enforced in each switch along this path and to send these rules to the SDN controller for the actual OpenFlow-based configuration of the switches.

Regarding the TPA, it has simply been implemented as a Web Service client (using the curl command) for the Out-of-Band case; and as an IP packet forged by the Scapy tool (but any other similar tool can be used) for the In-Band case.

The SDN test-bed is composed of five OpenFlow-enabled switches (see Fig.3), all of type Alcatel-Lucent OmniSwitch 6900-T20. The Floodlight controller (extended with our proposed TPC module) is running in a machine M1 (HP620, Ubuntu). It communicates to each SDN switch via the OpenFlow protocol (see dashed lines in Fig.3). The switches (named SW1 to SW5) are connected together through Ethernet 10 Gb links and the overall connectivity topology is shown Fig.3 (see solid lines). The switch ports are noted P_{ij} where i denotes the identifier of the switch and j the identifier of the port of this switch. Finally, a Client machine M2 (ASUS G75VW, Windows 8) is connected to the switch SW2; while a Web Server machine M3 (HP DC7700, Windows 7) and a FTP Server machine M4 (HP DC7700, Windows 8) are connected to the switch SW5. M2, M3 and M4 are connected to these switches with Ethernet 10 Gb links.

This test-bed aims to illustrate the case of a company having applications deployed over different servers/VMs and that need to exchange different types of sensitive data between them over a SDN network. This is of course a simplified layout regarding a full data center environment but it is however focused on the core part of the mechanism we want to evaluate (i.e., using our proposed extended SDN controller to enforce a trusted path over a network of SDN switches).

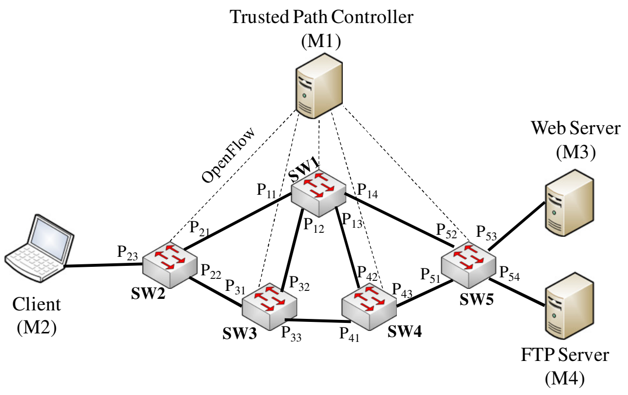


Fig.3. Test-bed of SDN/OpenFlow switches (OmniSwitch 6900-T20)

B. *Experimentations and results*

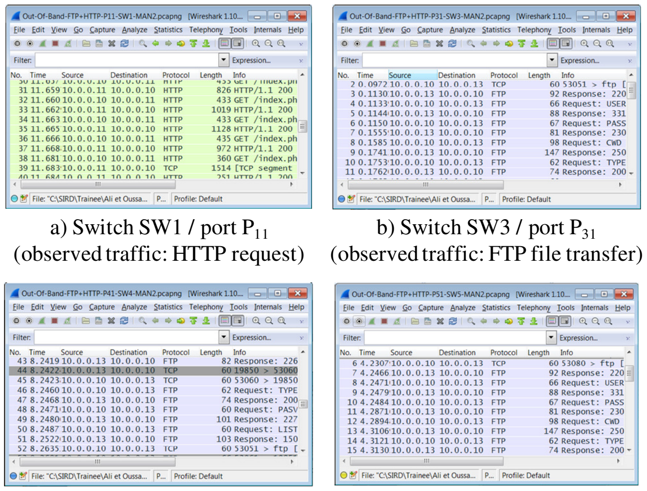
For the experimentation, we have considered the following data exchanges:

- An HTTP request from the Client M2 to the Web Server M3. No policy is requested for this HTTP request.
- An FTP file transfer of a sensitive data from the Client M2 to the FTP Server M4. To perform the FTP transfer of this sensitive file, a policy is requested and consists in avoiding the switch SW1 considered as un-trusted (e.g., as it may belong to an un-trusted network area).

The experimentation is composed of the following steps:

1. From the Client M2, send the policy (i.e., avoid switch SW1) to the TPC for the FTP transfer of a sensitive data to be done from M2 to M4. The two Out-of-Band and In-Band solutions have been respectively tested.
2. From the Client M2, perform the HTTP request to the Web Server M3.
3. From the Client M2, perform the FTP transfer of the sensitive data from M2 to the FTP Server M4.

In order to observe the path followed by the data, it is observed the packets on the following switch ports: P_{11}, P_{31}, P_{41}, P_{51} (see Fig.3). For this purpose, a port mirroring has been configured on each of these switch ports so that the traffic can be captured by the WireShark tool. Fig.4 depicts obtained results in WireShark once the HTTP request and the FTP transfer of the sensitive file have then been performed (in the Out-of-Band case but a similar result is obtained with the In-Band case). We can see that the policy has been well enforced as, while the HTTP request is crossing the switches SW2-SW1-SW5 (shortest path); the sensitive data has been transferred by crossing the switches SW2-SW3-SW4-SW5. The transfer of the sensitive data follows therefore a trusted (but longer) path as the switch SW1 is not authorized (i.e., un-trusted) by the policy applicable for this data.



Out-Of-Band-FTP+HTTP-P11-SW1-MAN2.pcapng [Wireshark 1.10....
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
File: "C:\SIRD\Trainee\Ali et Oussa...
P...
Profile: Default

```
No. Time   Source    Destination Protocol Length Info
 30 11.657 10.0.0.10 10.0.0.11   HTTP       433 GET /index.ph
 31 11.659 10.0.0.11 10.0.0.10   HTTP       826 HTTP/1.1 200
 32 11.660 10.0.0.10 10.0.0.11   HTTP       433 GET /index.ph
 33 11.662 10.0.0.11 10.0.0.10   HTTP      1019 HTTP/1.1 200
 34 11.663 10.0.0.10 10.0.0.11   HTTP       433 GET /index.ph
 35 11.665 10.0.0.11 10.0.0.10   HTTP      1128 HTTP/1.1 200
 36 11.666 10.0.0.10 10.0.0.11   HTTP       435 GET /index.ph
 37 11.668 10.0.0.11 10.0.0.10   HTTP       972 HTTP/1.1 200
 38 11.681 10.0.0.10 10.0.0.11   HTTP       360 GET /index.ph
 39 11.683 10.0.0.11 10.0.0.10   TCP       1514 [TCP segment
 40 11.684 10.0.0.11 10.0.0.10   HTTP       251 HTTP/1.1 200
```

a) Switch SW1 / port P_{11}
(observed traffic: HTTP request)

Out-Of-Band-FTP+HTTP-P31-SW3-MAN2.pcapng [Wireshark 1.10....
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
File: "C:\SIRD\Trainee\Ali et Oussa...
P...
Profile: Default

```
No. Time   Source    Destination Protocol Length Info
  2 0.0972 10.0.0.10 10.0.0.13   TCP         60 53051 > ftp [
  3 0.1130 10.0.0.13 10.0.0.10   FTP         92 Response: 220
  4 0.1133 10.0.0.10 10.0.0.13   FTP         66 Request: USER
  5 0.1144 10.0.0.13 10.0.0.10   FTP         88 Response: 331
  6 0.1150 10.0.0.10 10.0.0.13   FTP         67 Request: PASS
  7 0.1555 10.0.0.13 10.0.0.10   FTP         81 Response: 230
  8 0.1585 10.0.0.10 10.0.0.13   FTP         98 Request: CWD
  9 0.1741 10.0.0.13 10.0.0.10   FTP        147 Response: 250
 10 0.1753 10.0.0.10 10.0.0.13   FTP         62 Request: TYPE
 11 0.1762 10.0.0.13 10.0.0.10   FTP         74 Response: 200
```

b) Switch SW3 / port P_{31}
(observed traffic: FTP file transfer)

Out-Of-Band-FTP+HTTP-P41-SW4-MAN2.pcapng [Wireshark 1.10....
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
File: "C:\SIRD\Trainee\Ali et Oussa...
P...
Profile: Default

```
No. Time   Source    Destination Protocol Length Info
 43 8.2419 10.0.0.13 10.0.0.10   FTP         82 Response: 226
 44 8.2422 10.0.0.13 10.0.0.10   TCP         60 19850 > 53060
 45 8.2423 10.0.0.10 10.0.0.13   TCP         60 53060 > 19850
 46 8.2460 10.0.0.10 10.0.0.13   FTP         62 Request: TYPE
 47 8.2468 10.0.0.13 10.0.0.10   FTP         74 Response: 200
 48 8.2471 10.0.0.10 10.0.0.13   FTP         60 Request: PASV
 49 8.2480 10.0.0.13 10.0.0.10   FTP        101 Response: 227
 50 8.2487 10.0.0.10 10.0.0.13   FTP         60 Request: LIST
 51 8.2522 10.0.0.13 10.0.0.10   FTP        103 Response: 150
 52 8.2635 10.0.0.10 10.0.0.13   TCP         60 53051 > ftp [
```

c) Switch SW4 / port P_{41}
(observed traffic: FTP file transfer)

Out-Of-Band-FTP+HTTP-P51-SW5-MAN2.pcapng [Wireshark 1.10....
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
File: "C:\SIRD\Trainee\Ali et Oussa...
P...
Profile: Default

```
No. Time   Source    Destination Protocol Length Info
  6 4.2307 10.0.0.10 10.0.0.13   TCP         60 53080 > ftp [
  7 4.2466 10.0.0.13 10.0.0.10   FTP         92 Response: 220
  8 4.2471 10.0.0.10 10.0.0.13   FTP         66 Request: USER
  9 4.2479 10.0.0.13 10.0.0.10   FTP         88 Response: 331
 10 4.2484 10.0.0.10 10.0.0.13   FTP         67 Request: PASS
 11 4.2871 10.0.0.13 10.0.0.10   FTP         81 Response: 230
 12 4.2894 10.0.0.10 10.0.0.13   FTP         98 Request: CWD
 13 4.3106 10.0.0.13 10.0.0.10   FTP        147 Response: 250
 14 4.3121 10.0.0.10 10.0.0.13   FTP         62 Request: TYPE
 15 4.3130 10.0.0.13 10.0.0.10   FTP         74 Response: 200
```

d) Switch SW5 / port P_{51}
(observed traffic: FTP file transfer)

Fig.4. Wireshark traffic captures. The non sensitive HTTP request crosses SW2-SW1-SW5 (SW1 being untrusted); while the FTP transfer of the sensitive data crosses SW2-SW3-SW4-SW5 (trusted path)

In addition, the following performance times have been measured during the experimentation:

- *PolicySendingAndAck*: time to send the policy to the TPC module and get back the ack.
- *OpenFlowRulesConfiguration*: time to configure the OpenFlow rules in each switch.

Note that due to constraints in the Floodlight implementation, the SDN switches are actually configured (according to the computed trusted path) when the first packet of the sensitive flow is crossing the switches. Therefore, the total time to configure all the SDN switches is the time OpenFlowRulesConfiguration multiplied by the number of switches to configure.

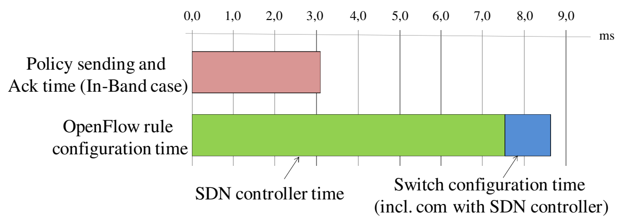


Fig.5. Policy sending time and OpenFlow rule configuration time

The performance times have been evaluated after having performed the previous experimentation 10 times. The PolicySendingAndAck time (to send the policy and get the ack) is of an average of 61 ms in the Out-of-Band case, and of an average of 3 ms in the In-band case.

The OpenFlowRulesConfiguration time per switch is of an average of 8,6 ms (and of course does not depend on the Out-of-Band or In-band architecture). Moreover, this OpenFlowRulesConfiguration time is split between (i) 7,5 ms (i.e., 87% of the time) spent in the SDN controller and (ii) 1,1 ms (i.e., 13% of the time) spent in the switch configuration (including the communication with the SDN controller). Fig.5 depicts these different values for the In-Band case.

V. DISCUSSIONS

Let analyze the In-Band and Out-of-Band architectures according to different criteria.

First, the policy sending *performance* (i.e., PolicySendingAndAck time) is then better for In-Band than for Out-of-Band.

Also, in the In-Band case, the TPA has not to be configured with the address of the TPC (as the *set-trusted-path* packet will be automatically forwarded to it when crossing the first SDN switch) and this improves the system *configurability*. Indeed, for the Out-of-Band, all TPAs must then be configured with the address of the TPC.

Regarding *security*, we can note that with the Out-of-Band architecture and its exposed Web Services, the TPC could be more easily attacked by a hacker to take control of the network. In-Band attacks are also possible but they could be more easily filtered by the SDN controller.

Finally, regarding *portability*, the In-Band architecture requires the *set-trusted-path* signaling packet (which embeds the policy in its payload) to be entirely forwarded to the SDN controller (i.e., with the payload) through the OpenFlow Packet-In. This aspect is mentioned in the OpenFlow specification from the release 1.2 by setting the field *max_len* of the OpenFlow Packet-In with the value OFPCML_NO_BUFFER (meaning that the packet is not buffered in the switch). However, it requires the switch to be configured accordingly. The Out-of-Band architecture has of course not this constraint as the *set-trusted-path* request is not transmitted through OpenFlow.

Fig.6 summarizes this comparison. Despite the last point on portability, we could then conclude that the balance falls more in favor of the In-Band architecture.

	Out-of-Band	***In-Band***
Performance	-	+
Configurability	-	+
Security	-	+
Portability	+	-

Fig.6. Comparison of Out-of-Band and In-Band architectures

VI. CONCLUSION

In this paper we have proposed a new approach which leverages the programmability offered by the SDN technology in order to enforce a trusted path to transfer sensitive data in the network. Indeed, given a policy related to a sensitive data (e.g., data should not cross nodes of a given area), our approach allows sending this policy to a Trusted Path Controller which automatically enforces it in the network. The two In-Band and Out-of-Band architectures have been investigated and implemented. Finally different experimentations and evaluations have been performed on a test-bed of SDN switches (OmniSwitch 69000-T20) which has allowed us to show the feasibility of this approach as well as some of its performances.

Different perspectives can be mentioned notably by making further analysis on the scalability of the two architectures (knowing however that the additional traffic is related to signaling and also only to the transfer of the sensitive data) or on their security (e.g., by signing the IP signaling packet or by certifying the trusted path configuration). In addition, the approach will be carried out in a full data center environment (e.g., integration of the Trusted Path Controller within a cloud management platform such as OpenStack and its Neutron network component). For this purpose, it will be considered the inter and intra data center parts of the trusted path as well as the use of distributed SDN controllers in order to provide the end-to-end trusted path in such environments.

ACKNOWLEDGMENT

The authors would like to greatly thank Alcatel-Lucent Enterprise, and especially Madani Dahili, for offering an access to their equipment and lab as well as for their very helpful support. They also thank Laurent Bernard (Télécom SudParis), Monsef Tazi (Télécom SudParis), and Dominique Verchère (Alcatel-Lucent Bell Labs) for all their help and advices.

REFERENCES

[1] Cloud Security Alliance (CSA), “The notorious nine cloud computing top threats in 2013”, February 2013.

[2] Article 29 Data Protection Working Party, “Opinion 05/2012 on cloud computing”, WP 196, Brussels, July 2012.

[3] S. Pearson and G. Yee (Eds.), “Privacy and security for cloud computing”, Springer, 2013.

[4] D. Pitt, “Trust in the cloud: the role of SDN”, Network Security, Elsevier, vol. 2013, no 3.

[5] S. Scott-Hayward, G. O'Callaghan, S.Sezer, S. "SDN security: A survey". In IEEE conference on SDN for Future Networks and Services, SDN4FNS, Trento, Italy, 2013.

[6] R. Kloti, V. Kotronis, P. Smith, "OpenFlow: A security analysis," IEEE International Conference on Network Protocols, ICNP, Göttingen, Germany, 2013.

[7] S. Shin, G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks", IEEE International Conference on Network Protocols, ICNP, Austin, USA, 2012.

[8] V. Kotronis, D. Schatzmann, B. Ager, “On bringing private traffic into public SDN testbeds”, ACM SIGCOMM workshop on Hot topics in software defined networking, Hong Kong, 2013.

[9] F. Derakhshan, H. Grob-Lipski, H. Roessler, P. Schefczik, M. Soellner, “Enabling Cloud Connectivity Using SDN and NFV Technologies”, Mobile Networks and Management, Lecture Notes of the Institute for Computer Sciences, Vol. 125, Springer, 2013.

[10] F. Salvestrini, G. Carrozzo, N. Ciulli, "Towards a Distributed SDN Control: Inter-Platform Signaling among Flow Processing Platforms", IEEE SDN Conference on Future Networks and Services, SDN4FNS, Trento, Italy, 2013.

[11] E. Oki, T. Takeda, A. Farrel, “Extensions to the Path Computation Element Communication Protocol (PCEP) for Route Exclusions”, IETF RFC 5521, April 2009.

[12] M. Ghorbel, A. Aghasaryan, S. Betgé-Brezetz, M.P. Dupont, G.B. Kamga, S. Piekarec, “Privacy data envelope: concept and implementation”, Ninth Annual International Conference on Privacy, Security and Trust, PST, Montréal, Canada, 2011.

[13] S. Betgé-Brezetz, G.B. Kamga, M.P. Dupont, A. Guesmi, “End-to-End Privacy Policy Enforcement in Cloud Infrastructure”, IEEE Conference on Cloud Networking, CloudNet, San Francisco, USA, 2013.