International Journal of Control and Automation
Vol. 11, No. 6 (2018), pp.123-136
http//dx.doi.org/10.14257/ijca.2018.11.6.12

SDN MPTCP Sub-Flows Routing Security Against MiTM Attacks

Mouhcine Chliah[1*], Ghizlane Orhanou[2] and Said El Hajji[3]

Laboratory of Mathematics, Computing and Applications - Information Security
Faculty of Sciences, Mohammed V University in Rabat,
BP1014 RP, Rabat, Morocco
[1]*mouhcine.chliah@gmail.com,* [2]*orhanou@fsr.ac.ma,* [3]*elhajji@fsr.ac.ma*

Abstract

Software-Defined Networking (SDN) is a new paradigm for future network architecture, that by decoupling control plane from the forwarding plane will allow us to deploy the control function on platforms with higher capacities than those of legacy network nodes. The abstraction through a standard network API enables development of high value-added network services, such as, pathfinder algorithm that allows forwarding of segments coming from the same packet via different paths, hence limiting the impact of having a sniffer on one path being used.

In this paper, we propose a new way to face attacks based on sniffing like MiTM in SDN Network. We'll use the pathfinder and SLS (Secure Load Sharing) algorithms that have already proven their efficiency in legacy network, as SDN application, to control the Multipath TCP (MPTCP) sub-flow routing.

After an introduction, section 2 will present SDN paradigm and MPTCP security issues, followed by a dedicated section to recall pathfinder and SLS algorithms. Finally, an implementation on SDN is performed to demonstrate the benefits of such routing decision controlled by pathfinder facing attacks based on sniffing.

Keywords: *SDN; Security; OpenFlow; MitM; pathfinder; Graph Theory; MPTCP; Mininet*

1. Introduction

Nowadays, the majority of our networks, are made based on physical topology, starting by a high-level design showing how servers, switches & routers are linked to each other, then if an update is needed (topology change, bandwidth demands, or new service introduction), it's going to be very expensive, complex, and error prone. In addition to that, this is not compatible with the cloud environments which need more flexibility in handling virtual machines mobility, more throughput, more security, but with less and less investments on OPEX "OPerational EXpenditure".

So, what are possible solutions? SDN [11], Software defined networking decouples the control plane and the data planes. It allows to centralize the intelligence and all the logics in more powerful or even virtualized servers, abstract the network topology to upper layers, and last but not least, the possibility to handle more efficiently the bandwidth of existing infrastructure, like using multiple path, re-route traffic based on actual utilization. This new paradigm will be translated to more flexible decisions making.

In this paper, we'll focus on these fundamental advantages of SDN, which are, decoupling control plan from forwarding plan, and transferring the decision making to controllers hosted on more powerful server. Which in return, provide more power and efficiency to network applications like pathfinder and SLS algorithms [1,2,3], that was

Received (February 5, 2018), Review Result (April 30, 2018), Accepted (May 4, 2018)
[*] Corresponding Author

designed mainly to face attacks based on sniffing in legacy networks. We'll introduce the MPTCP [5] (Multipath TCP), which allows the usage of all available interfaces on client & server stations for better bandwidth utilization. Finally, we propose to use pathfinder/SLS algorithms as routing engine for MPTCP sub-flows. Hence, leveraging the security benefits of such algorithms in multipath environment.

So, we start by an introduction to SDN, MPTCP and their security issues related works. In section 3, we give a recall of Pathfinder and Secure Load Sharing algorithms designed based on graph theory [6,7], which allow to discover all available paths from source to destination based on defined criteria, limiting the participation of one node in multiple paths that could be infected. Finally, section 4 will be dedicated to a case study implementation, simulation and analysis of MPTCP sub-flows routing using pathfinder/SLS algorithms.

2. SDN and MPTCP Security Issues:

2.1. SDN Introduction

Since the recent technological advances in the fields of computer science and telecommunications, the need of redesigning and proposing new network architecture has showed up, in light of the fact that the conventional ones started to demonstrate their constraints and limitations. These days, the tendency is to interconnect everything, through enabler technologies, for example, Data Center, Cloud Computing, Virtualization, Internet of Things and more. These novel methodologies require, aside from bandwidth, a simpler and more flexible network, where appliances and virtualization are encouraged. Software Defined Networking (SDN) is the solution.

The SDN paradigm is based on four pillars as illustrated in Figure 1:

1. One of the fundamentals advantages of SDN is the separation between control and data planes. This new concept brings to operators a particular advantage in terms of centralized or semi-centralized programmatic control.

2. The second advantage of using SDN is the ability to use a centralized control plane, which handle out the decision-making logic from the network devices like routers and switches into outer controllers [13]. As a result of this, an abstraction to the whole network will be easily done and available to application developers.

3. Open interfaces between different layers: OpenFlow (OF) [12] is viewed as one of the first software-defined networking (SDN) standards. It initially designed the communication protocol in SDN environments, in order to empower the SDN Controller to pilot directly the forwarding plane of physical and virtual network nodes like switches and routers. So, it can easily and much faster adjust to changing business requirements

4. In SDN environment, it is possible to use network applications that have the ability to monitor traffic and to process capabilities. This will subsequently support; load balancing, a dynamic QoS provisioning, and access control.

SDN controllers come in many flavors, depending on many things: the programming language in which they are implemented, the performance they achieve, the time needed for a programmer to start coding applications for that controller, the type of southbound interfaces it supports, the purpose for which they are intended, the support for distributed control, *etc*. Here are some examples of these SDN controllers: NOX, POX, ONOS, OpenDaylight, Ryu or Floodlight. [13]

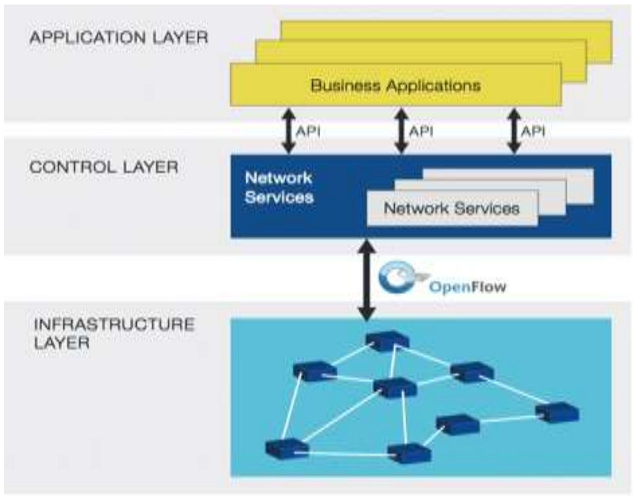


Figure 1. Open Networking Foundation: SDN Layers. [22]

2.1. Multipath TCP Operation

Multipath TCP (MPTCP) goal is to allow a Transmission Control Protocol (TCP) [9] connection to use multiple paths to maximize resource usage and increase redundancy [10].

The redundancy offered by Multipath TCP empowers inverse multiplexing of resources, and therefore increase the TCP throughput to the sum of all available links as opposed to utilizing a single one as required by TCP. Multipath TCP is backwards compatible with TCP.

The basic approach to Multipath TCP is the division of the application's single outbound flow into multiple sub-flows, each operating its own end-to-end TCP session, and the re-joining of multiple input sub-flows into a single flow to present to the remote counterpart application, as presented in Figure 2.

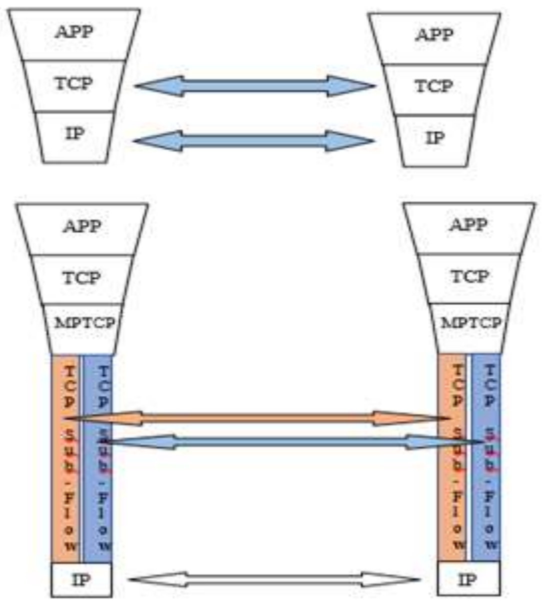


Figure 2. TCP Stack Compared to MPTCP

In more simple definition, MPTCP create connection between two hosts, whiles TCP create connection between two interfaces, as illustrated by Figure 3.

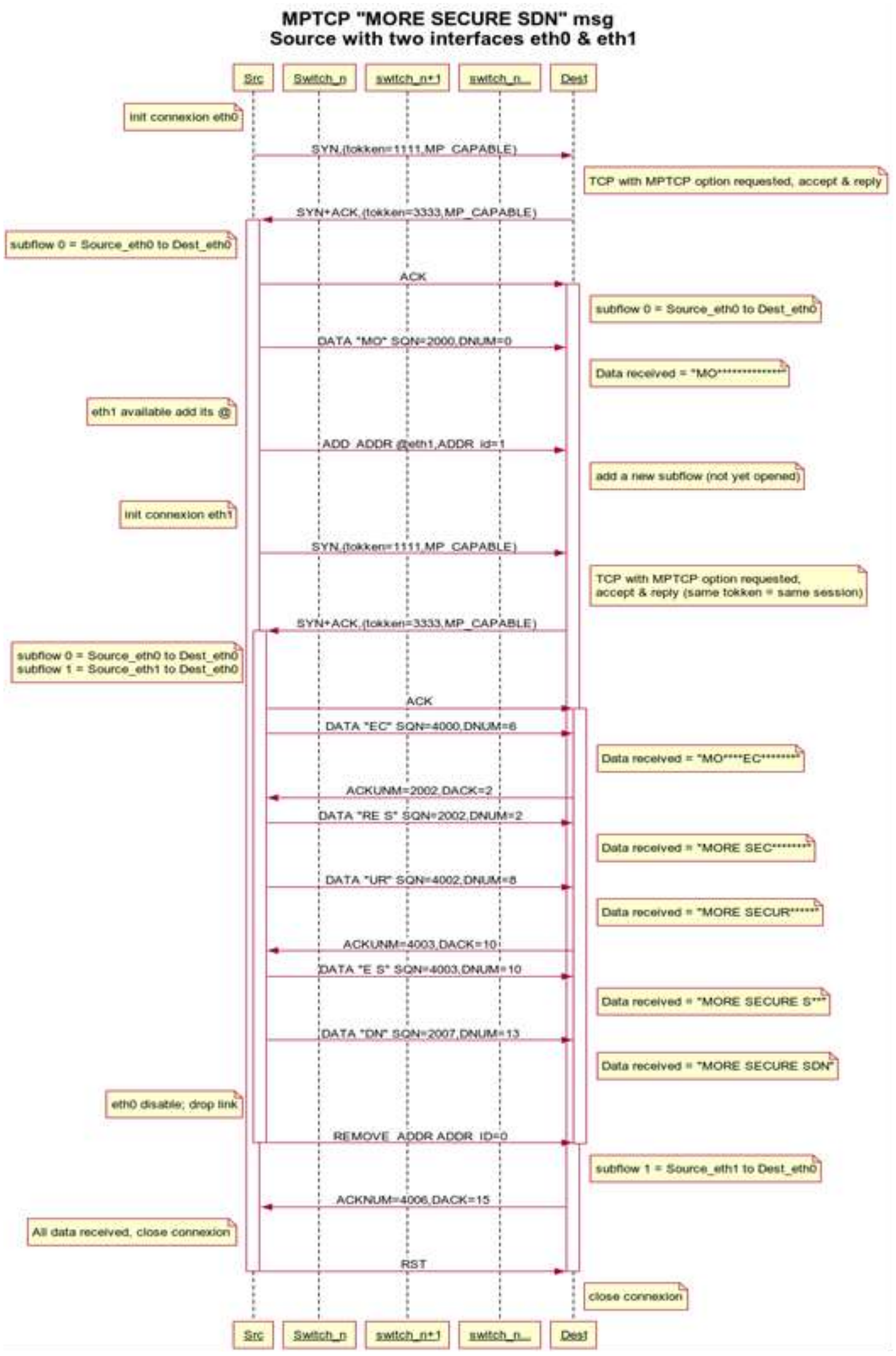

MPTCP "MORE SECURE SDN" msg
Source with two interfaces eth0 & eth1
Src
Switch_n
switch_n+1
switch_n...
Dest
init connexion eth0
SYN,(tokken=1111,MP_CAPABLE)
TCP with MPTCP option requested, accept & reply
SYN+ACK,(tokken=3333,MP_CAPABLE)
subflow 0 = Source_eth0 to Dest_eth0
ACK
subflow 0 = Source_eth0 to Dest_eth0
DATA "MO" SQN=2000,DNUM=0
Data received = "MO**************"
eth1 available add its @
ADD_ADDR @eth1,ADDR_id=1
add a new subflow (not yet opened)
init connexion eth1
SYN,(tokken=1111,MP_CAPABLE)
TCP with MPTCP option requested, accept & reply (same tokken = same session)
SYN+ACK,(tokken=3333,MP_CAPABLE)
subflow 0 = Source_eth0 to Dest_eth0
subflow 1 = Source_eth1 to Dest_eth0
ACK
DATA "EC" SQN=4000,DNUM=6
Data received = "MO****EC********"
ACKUNM=2002,DACK=2
DATA "RE S" SQN=2002,DNUM=2
Data received = "MORE SEC********"
DATA "UR" SQN=4002,DNUM=8
Data received = "MORE SECUR******"
ACKUNM=4003,DACK=10
DATA "E S" SQN=4003,DNUM=10
Data received = "MORE SECURE S***"
DATA "DN" SQN=2007,DNUM=13
Data received = "MORE SECURE SDN"
eth0 disable; drop link
REMOVE_ADDR ADDR_ID=0
subflow 1 = Source_eth1 to Dest_eth0
ACKNUM=4006,DACK=15
All data received, close connexion
RST
close connexion
Src
Switch_n
switch_n+1
switch_n...
Dest

Figure 3. MPTCP Dataflow "MORE SECURE SDN" Message Exchange

Through this paper, we'll look at SDN and MPTCP from different perspectives. For example, when we speak about SDN security, most of the research [20] focus on the new features being introduced like; dynamic flow control [18], where it's possible via a network application to control dynamically network flows, or the centralized view of the network, thus, a network application naturally has a view of all connected user plane, and it can control all data plane in a centralized manner [19], in addition to network programmability and much more features not cited here.

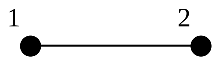
Concerning MPTCP, it's being seen as an answer [21] to some inherent shortcomings in single-path TCP, by allowing multihomed end-hosts to utilize simultaneously more than one interface, for a higher throughput and availability.

For the rest of this paper, we'll focus more on programmability aspect of a network being managed by SDN Controller. Then we'll introduce the pathfinder and SLS (Secure Load Sharing) algorithms as a SDN application, taking advantage of the centralized view of the network, allowing us to better determinate all possible paths from source to destination. Finally, by using the pathfinder and SLS results, when routing the different MPTCP sub-flows, we'll not only use efficiency the existing bandwidth, but we'll also take full advantage of security aspect introduced by these algorithms.

3. Pathfinder Algorithm Recall

3.1. Network Modelization:

Using the Graph theory we'll modelize the network as a directed graph $G = (H \cup S, E)$, where H is the set of hosts, S is the set of Switch & Routers, and E is the set of all links within the network. Each edge $(e_i, e_j) \in E$ is defined with its bandwidth, latency, delay, etc. To simplify, we'll call these characteristics a link weight $w_{ij} = w(e_i, e_j) \geq 0$ .We'll define also $\mathcal{M} = \{M_1, M_2, ...M_n\}$ as the set of all messages exchanged in the network, where a message $M_k (1 \leq k \leq n)$ is defined with the triple (s_k, d_k, l_k), where $s_k \in H$ is the source, $d_k \in H$ is the destination, and l_k is the expected load. Finally, we'll use the function $m_k(e_i, e_j) \in \{0,1\}$ to indicate if the message M_k is moving through the link (e_i, e_j).



In the rest of this study, we'll focus on symmetric graphs only, in the sense that if node *a* is linked to node *b*, then node *b* is linked to node *a* via the same link, and also there is no loops, *i.e.*, there is no link from node *a* to itself.

Another important definition is the adjacency matrix [7], defined as a square matrix with elements indicating whether pairs of nodes are adjacent or not in the network.

More generally, this matrix will be defined as:

$$A = w_{ij} \text{ Where } w_{ij} = \begin{cases} =0 & edge(e_i, e_j) \notin E \\ \neq 0 & edge(e_i, e_j) \in E \end{cases} \tag{1}$$

$$\begin{pmatrix} w_{00} & \cdots & w_{0j} \\ \vdots & \ddots & \vdots \\ w_{i0} & \cdots & w_{ij} \end{pmatrix}$$

Where the edge (e_i, e_j) represents the link between the nodes $i, j \in H \cup S$,and W_{ij} represents the link's weight between them.

Hence, the graph theory will be used to translate a computer network to a graph, as shown below in this example, Figure 4.

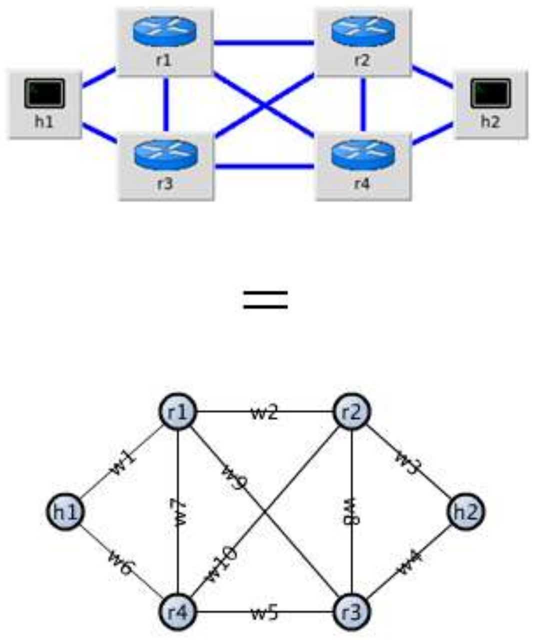


Figure 4. Modeling a Computer Network as a Graph

3.1. Related Works:

The first research published in [1], was focusing on limiting the impact of attacks based on sniffing like MitM [8] on legacy networks, by dispatching segments from one packet via different available paths. To do so, several conditions on links' selection were defined:

- First Condition: The possible paths $(P_i ... P_j)$ from source to destination should provide the same weight $(w_i = w_j)$. In such situation, the router will not bother about the bit rate or packet size to be used on each link. But the most important case is to have links going through **different intermediate nodes**. Hence, in the event that one node is subject to sniffing or MiTM attack, the attacker will have the capacity to get just a part of the entire message that streams through contaminated node, and we are certain that the node is not part of any other used path.

- If not possible to apply points suggested by condition 1, then we'll look for paths with different intermediate nodes, regardless of available bit rates on these paths.

Taking into account Graph Theory Adjacency Matrix, we've developed a new algorithm named pathfinder (See. Figure 5) that evaluates every single conceivable path from Source to Destination. Then, another algorithm named SLS "Secure Load Sharing" looks over the given paths for all combinations that meet the conditions proposed above [2].

```
Input:
G=(HuS,E)
N := Number of vertices.
Integer path[n] := Will be used to store paths during exploration.
Boolean usedNodes[n] := Will be used to lock vertices already used.
Integer source := Source
Integer target := Destination
Integer tmp := tmp variable to be used to store total distance for given path.

pathFinder(source,0)

Funct pathFinder (Integer position, Integer depth)
    path[depth]:=position
    IF (position==target) THEN  // We reached the target
        tmp := 0 // initiate total distance.
        FOR(Integer i := 0 ; i <= depth ; i++)                    // display the solution
            DISPLAY path[i]
            IF (i==0) continue;
            tmp := adjaMatrix[path[i-1]][path[i]] + tmp;
        END FOR
        DISPLAY <<" = "<<tmp<< endl;
        return;
    END IF
    usedNodes[position] := true // We mark the vertices being used.
    FOR (Integer i: = 0; i<n; i++) {                    // We continue checking remaining vertices.
        IF (adjaMatrix[position][i]==0 || usedNodes[i]) THEN continue END IF
        pathFinder (i,depth+1)
    END FOR
    usedNodes[position] := False; // unlock the vertices.
END Function
```

Figure 5. Pathfinder Algorithm [6]

It's important to note, that the initial version came with a limitation of being able to handle a network with up to 12 nodes only [1], and that restriction was bypassed with the new version named evolved pathfinder [3], that pushed that limit to more than hundreds node.

The remainder of this paper will be focusing on the simulation of pathfinder as an application within a network managed by a SDN controller. So first, we'll start by an introduction to the simulation environment. Next, we'll implement the pathfinder as SDN application. Then, the developed module will be used as a base decision to route MPTCP sub-flows. Finally, using the simulation results, we'll provide a deep analysis and advantages of using pathfinder, not only to face attacks based on sniffing, but also a better and efficient way for bandwidth utilization.

4. Pathfinder with MPTCP in SDN Environment

In this section, we aim to perform an end to end simulation of MPTCP using a pathfinder as a route decision maker for sub-flow routing.

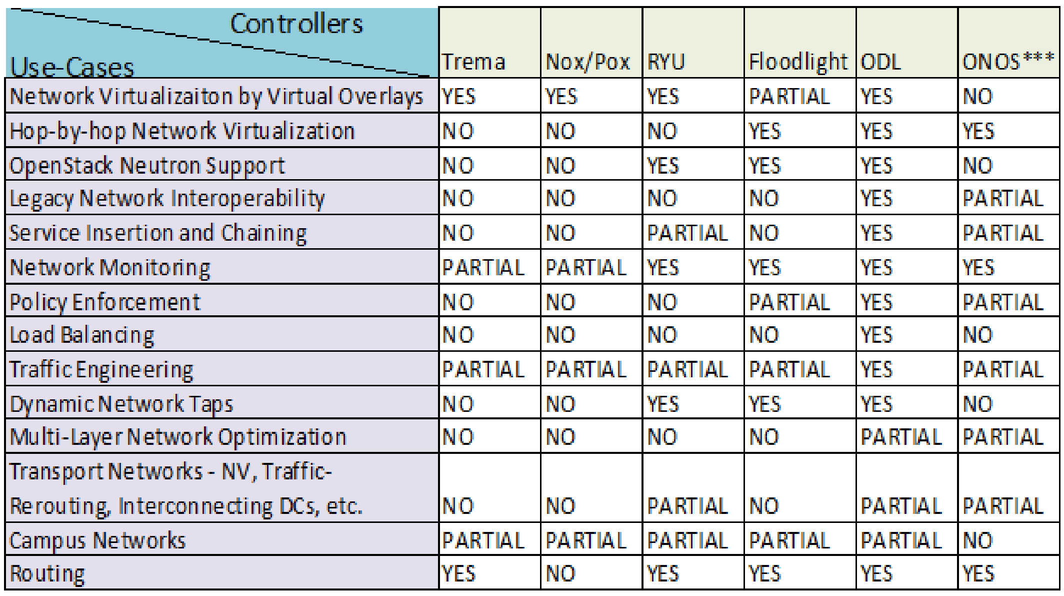
Mininet is an open source network emulator [14,15]. It creates a network of virtual hosts, switches, controllers, links, and application codes on a single machine (VM, cloud or native). The main characteristics that pushed us to use Mininet are:

- Flexibility in creating new topologies and features.
- Applicability in real life: the topologies and configuration done using Mininet could be used in real life as well as on real SDN controllers.

- Scalability: the prototyping environment must be scaling to large networks that can range from hundreds to thousands of forwarding devices on only a computer.

By using Mininet, we can create, edit & share SDN components with other networks and perform interactions. It may include Hosts, Switches, Controllers and Links. A host on Mininet runs standard Linux network software. Each one provides processes with exclusive ownership virtual network interface, ports, addresses and routing tables (such as ARP and IP). The OpenFlow switches created by Mininet provide the same packet delivery semantic that would be provided by a hardware switch. Both user-space and kernel-space switches are available. In addition, in Mininet simulation, the controllers can be run on the real or simulated network as long as the machine on which the switches are running has connectivity to the controller. If desired, this network emulator creates a standard controller inside the local simulation environment, and virtual connections can also be created among the elements through their virtual interfaces. We can also use remote controllers on a physical or a virtual machine.

For the rest of this section, we'll focus on OpenDayLight (ODL) as it proposes a large number of features as illustrated in Figure 6.



Controllers Use-Cases	Trema	Nox/Pox	RYU	Floodlight	ODL	ONOS***
Network Virtualizaiton by Virtual Overlays	YES	YES	YES	PARTIAL	YES	NO
Hop-by-hop Network Virtualization	NO	NO	NO	YES	YES	YES
OpenStack Neutron Support	NO	NO	YES	YES	YES	NO
Legacy Network Interoperability	NO	NO	NO	NO	YES	PARTIAL
Service Insertion and Chaining	NO	NO	PARTIAL	NO	YES	PARTIAL
Network Monitoring	PARTIAL	PARTIAL	YES	YES	YES	YES
Policy Enforcement	NO	NO	NO	PARTIAL	YES	PARTIAL
Load Balancing	NO	NO	NO	NO	YES	NO
Traffic Engineering	PARTIAL	PARTIAL	PARTIAL	PARTIAL	YES	PARTIAL
Dynamic Network Taps	NO	NO	YES	YES	YES	NO
Multi-Layer Network Optimization	NO	NO	NO	NO	PARTIAL	PARTIAL
Transport Networks - NV, Traffic-Rerouting, Interconnecting DCs, etc.	NO	NO	PARTIAL	NO	PARTIAL	PARTIAL
Campus Networks	PARTIAL	PARTIAL	PARTIAL	PARTIAL	PARTIAL	NO
Routing	YES	NO	YES	YES	YES	YES

Figure 6. SDN Controllers Comparison [13]

OpenDaylight Controller runs in a JVM. Being a Java application, it can potentially run on any operating system that supports Java. So, we will use one virtual machine with 2G Ram minimum, to run the Mininet emulated network and the OpenDaylight controller. This latter exposes open northbound APIs which are used by applications, in our case, the pathfinder as shown in Figure 7.

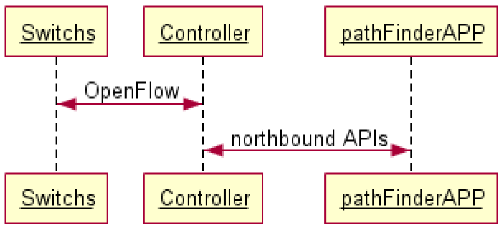


Figure 7. PathFinder Application Dataflow

In the following, we'll focus on the implementation of pathfinder as SDN application, taking advantage of the four pillars of SDN. Then on MPTCP sub-flows routing based on pathfinder/SLS output. After that a security analysis is performed.

4.1 Pathfinder Implementation as SDN Application

For this simulation, we'll use a network with 4P "Provider" routers in full mesh topology, and 12 PEs "Provider Edge" routers dual homed (Figure 8). The simulation was realized on a VM with 2 CPUs (1GHz each) and 4 Go RAM. The result showed that the same algorithm is more efficient in SDN environment, mainly because of the centralized approach of SDN.

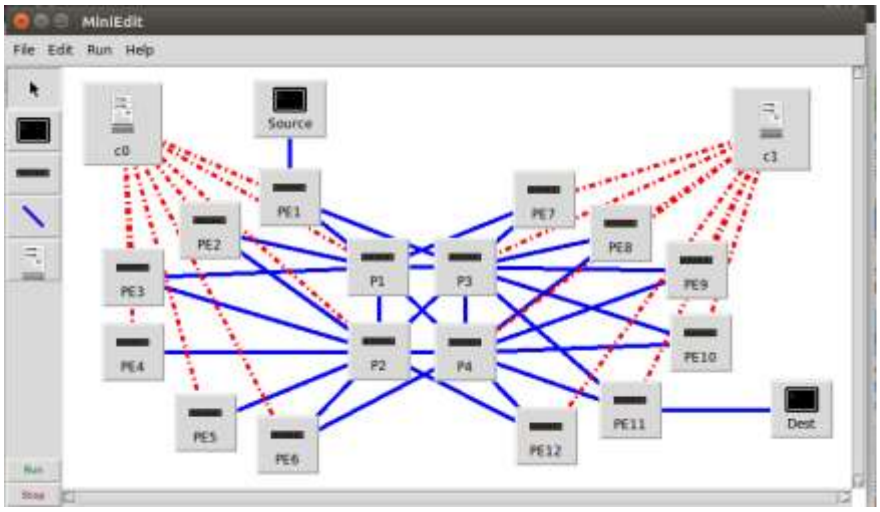


Figure 8. 1st Case Study

In commercial environment, most of service providers install their production platforms with local and geo-redundancy configuration. Moreover, a group of controllers is installed per zone to manage a limited number of switches and routers. Such configuration, will allow pathfinder to handle globally more network elements, with much more efficiency, as illustrated in Figure 9.

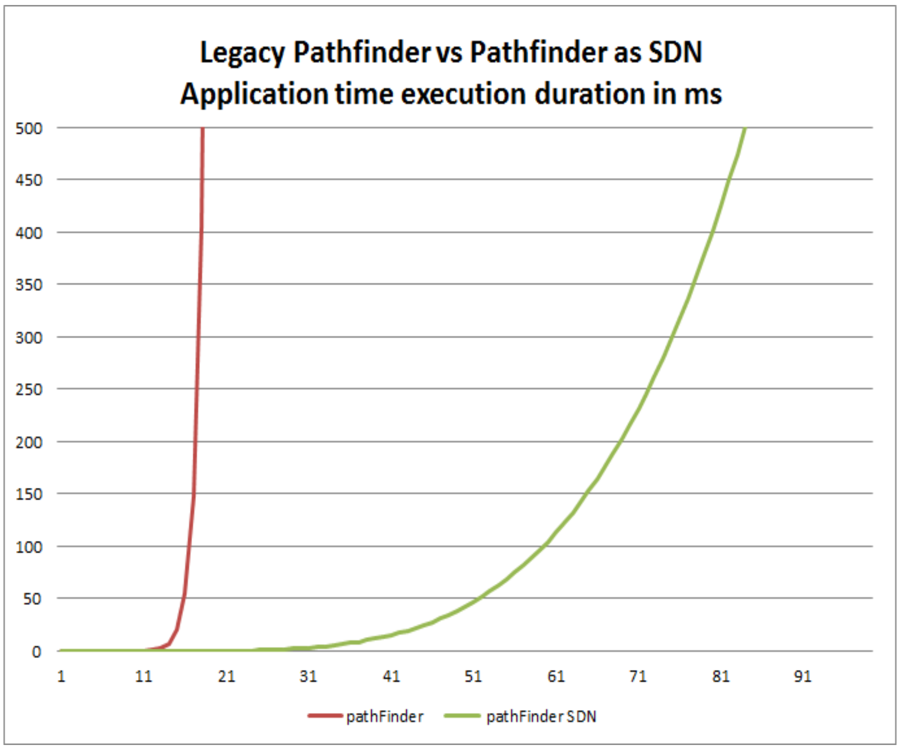


Figure 9. Comparison of Pathfinder Execution Time between Legacy & SDN Environment

To recall, the pathfinder will be used as a decision maker for MPTCP sub-flows routing, hence a rapid and efficient path selection is a key against MiTM attacks.

4.2 MPTCP Simulation using Pathfinder for Sub-Flows Routing:

Now that we've re-simulated the pathfinder as SDN application, we'll move to the second case study illustrated in Figure 10 and Figure 11, showing a network with 4 switches only- for simplicity- and two hosts with MPTCP capability.

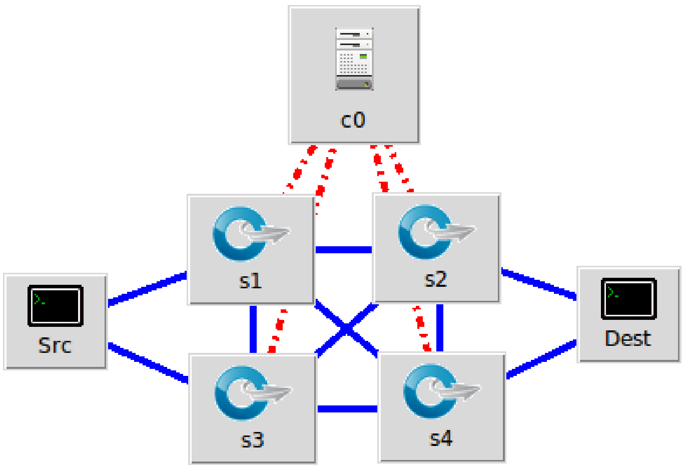


Figure 10. 2nd Case Study

The four switches are configured in full mesh connectivity. Without pathfinder, this network wouldn’t work without the use of protocols like spanning tree protocol (STP) [16], that blocks redundant links to avoid loops but provide much more limitation, like: having one path from source to destination, limiting the total possible bandwidth, being a single point of failure, and presenting a security issue if one node inside the path is infected by a sniffer.

Using the multipath version of pathfinder/SLS, we’ll be able to:

1. Use as much as possible the bandwidth provided by the network, as no link will be blocked.
2. Fast failover detection & redirection with the help of SDN.
3. Finally, increasing the complexity of executing a successful sniffing attack, as two successive segments from same packets will be forward via totally different paths.

The solution has been designed as shown in Figure 12. Following the open flow standard, the switch will share a PacketIn message with the controller. This message contains the captured packet received from the source node. There are two reasons why this might happen; there could be an explicit action as a result of a match asking for this behavior, or from a miss in the match tables, or a ttl error [17]. After that the controller will share the actual topology reflecting the available link and the packet header with the pathfinder, which calculates the possible paths from source to destination, that will be examined by SLS algorithm conditions to propose a possible and safe combination of paths. Then the controller will send flows to be installed on concerned switchs. Finally, the first packet got forwarded from source to destination.

Once a second packet needs to be forwarded via the second interface/path, the controller will automatically request from concerned switch to install the new flows, based on initial answer from Pathfinder application, then the second packet will get forwarded to destination.

The Figure 13 shows directly the flows that were installed by the controller on the first switch as example.

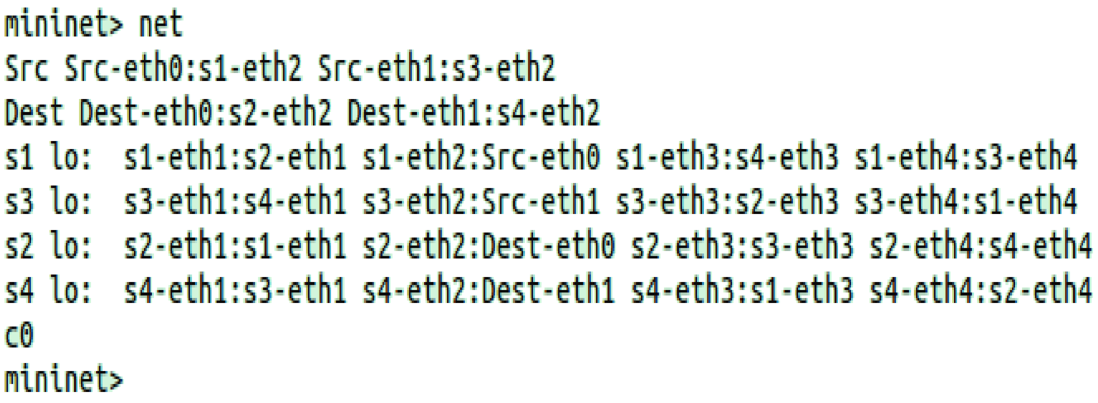

```
mininet> net
Src Src-eth0:s1-eth2 Src-eth1:s3-eth2
Dest Dest-eth0:s2-eth2 Dest-eth1:s4-eth2
s1 lo:  s1-eth1:s2-eth1 s1-eth2:Src-eth0 s1-eth3:s4-eth3 s1-eth4:s3-eth4
s3 lo:  s3-eth1:s4-eth1 s3-eth2:Src-eth1 s3-eth3:s2-eth3 s3-eth4:s1-eth4
s2 lo:  s2-eth1:s1-eth1 s2-eth2:Dest-eth0 s2-eth3:s3-eth3 s2-eth4:s4-eth4
s4 lo:  s4-eth1:s3-eth1 s4-eth2:Dest-eth1 s4-eth3:s1-eth3 s4-eth4:s2-eth4
c0
mininet>
```

Figure 11. 2nd Case Study: Interface Configuration

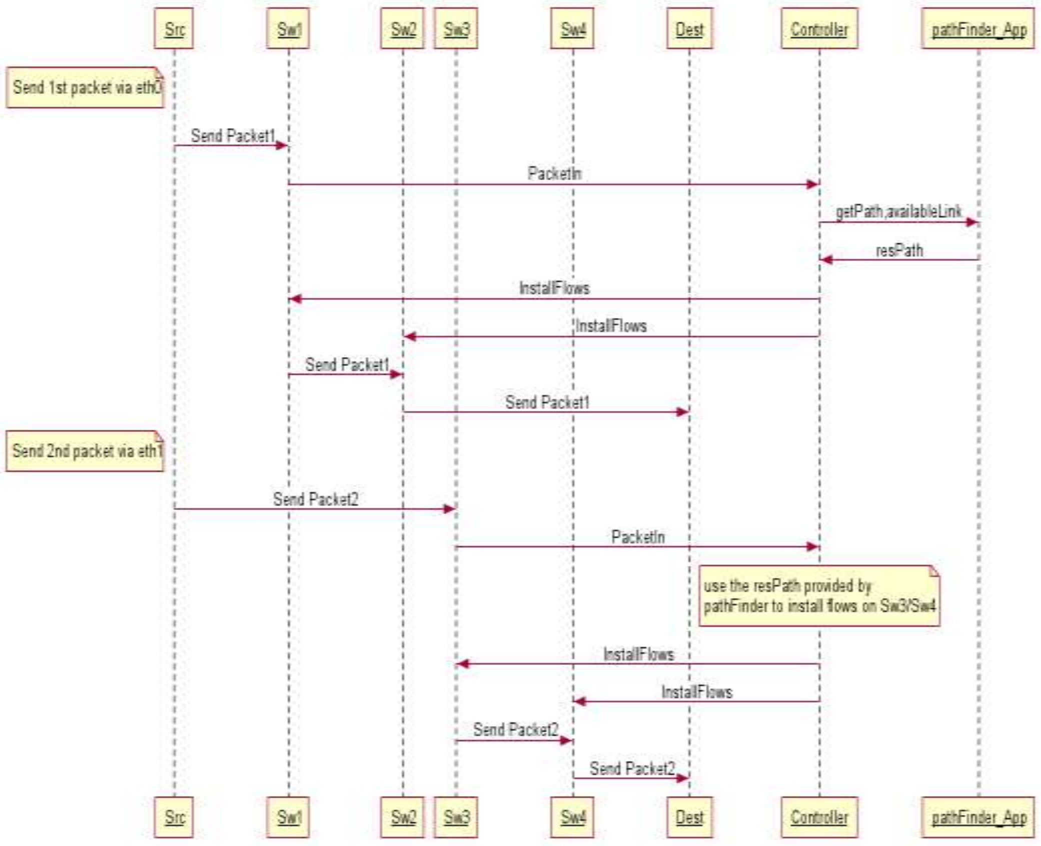


Figure 12. End to End Dataflow including Pathfinder Application

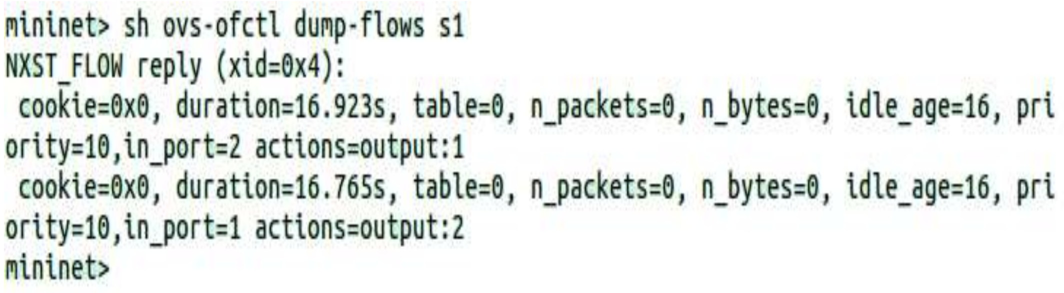

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=16.923s, table=0, n_packets=0, n_bytes=0, idle_age=16, pri
ority=10,in_port=2 actions=output:1
 cookie=0x0, duration=16.765s, table=0, n_packets=0, n_bytes=0, idle_age=16, pri
ority=10,in_port=1 actions=output:2
mininet>
```

Figure 13. Flows Installed by Controller on Sw1

4.3 SDN Security Against MiTM Attack Analysis:

Within this research, we've used the SDN network programmability, and the centralized view features, to develop a more reactive pathfinder/SLS algorithms. Which in turn help in better paths' selection based on: bandwidth, delay, intermediate nodes, and the most important point, one node can be part of one path only. Hence, limiting the impact of having an infected node in more than one path being used for the different MPTCP sub-flows routing.

The security gained with the usage of multipath SLS has already been validated with simulation under the research [4], where a sniffer attack was emulated using Wireshark, and we demonstrated how it was impossible to get the full message using one probe. The actual release of pathfinder/SLS increase the complexity of MiTM attack success.

We can still push the simulation, and use an IDS "Intrusion Detection System", to develop a dynamic pathfinder able to react and change possible routes based on intrusion detection, hence moving from a passive routing implementation in a multipath environment, to an aware security routing mechanism.

5. Conclusion

In this paper, we've focused on the programmability, flexibly, and elasticity aspects of a network managed by SDN, to develop an application to control more efficiently the MPTCP sub-flows routing within SDN network.

The purpose of this paper was to introduce a new way to face attacks based on sniffing like MiTM attacks in SDN environment. For this, we've developed pathfinder & Secure Load Sharing algorithms as an application, interfacing with SDN controllers, to take command over the MPTCP sub-flows routing decision, by ensuring the utilization of paths with different intermediate nodes, limiting the utilization of multiple infected paths.

Future works will push the simulation to take more advantages from SDN by using parallel programming, taking the benefit of having multiple SDN Controllers, and introduce a new module for real time optimization, to handle better bandwidth demand, and automatic network configuration & optimization to face attacks like DDoS.

References

[1] M. Chliah, G. Orhanou and S. E. Hajji, "New secure routing method & applications facing MitM attacks", 2014 International Conference on Next Generation Networks and Services (NGNS), Casablanca, **(2014)**, pp. 92-95.

[2] M. Chliah, G. Orhanou and S. E. Hajji, "New Secure Load Sharing algorithm in Network Layer", International Journal of Security and Its Applications, vol. 10, no. 1, **(2016)**, pp. 155-166.

[3] M. Chliah, K. Chetioui, G. Orhanou and S. ElHajji, "Evolved pathfinder algorithm for secure load sharing in the network layer", 2016 International Conference on Engineering & MIS (ICEMIS), Agadir, **(2016)**, pp. 1-6.

[4] M. Chliah, G. Orhanou and S. E. Hajji, "Countering MitM Attacks Using Evolved PathFinder Algorithm", IJCAC, vol. 7, no. 2, **(2017)**, pp. 41-61.

[5] MultiPath-TCP.org, 'MultiPath TCP', 2016. [Online]. Available: https://www.multipath-tcp.org/ [Accessed: 15-Apr-**2018**].

[6] R. Diestel, "Graph Theory", 4th Edition, Springer, vol. 173, **(2012)**.

[7] J. D. Fekete, "Visualizing networks using adjacency matrices: Progresses and challenges", 2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics, Huangshan, **(2009)**, pp. 636-638.

[8] M. Conti, N. Dragoni and V. Lesyk, "A Survey of Man In The Middle Attacks", IEEE Communications Surveys and Tutorials, vol. 18, no. 3, **(2016)**, pp. 2027-2051.

[9] C. Raiciu, M. Handley, S. Barre and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", 2011. [Online]. Available: https://tools.ietf.org/html/rfc6182. [Accessed: 15-Apr-**2018**].

[10] M. Kheirkhah, I. Wakeman and G. Parisis, "MMPTCP: A multipath transport protocol for data centers", IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, (**2016**), pp. 1-9.

[11] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey", Proceedings of the IEEE, vol. 103, no. 1, **(2015)** January, pp. 14-76.

[12] Open Networking Foundation, 'OpenFlow', 2016. [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow [Accessed: 15-Apr-**2018**].

[13] O. Salman, I. H. Elhajj, A. Kayssi and A. Chehab, "SDN controllers: A comparative study", 2016 18th Mediterranean Electrotechnical Conference (MELECON), Lemesos, Cyprus, **(2016)**, pp. 1-6.

[14] Mininet, 'Mininet', 2016. [Online]. Available: http://mininet.org/ [Accessed: 15-Apr-**2018**].

[15] M. J. Mišić and S. R. Gajin, "Simulation of Software Defined Networks in Mininet environment", 2014 22nd Telecommunications Forum Telfor (TELFOR), Belgrade, **(2014)**, pp. 1055-1058.

[16] G. S. Antonova, "Spanning Tree Protocol Interoperability with Other Loop Prevention Algorithms", 2007 Canadian Conference on Electrical and Computer Engineering, Vancouver, BC, **(2007)**, pp. 1098-1101.

[17] Flowgrammable Team, 'Openflow messages', 2016. [Online]. http://flowgrammable.org/ [Accessed: 15-Apr-**2018**].

[18] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks", In Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, **(2010)**.

[19] H. Kim and N. Feamster, “Improving network management with software defined networking”, Communications Magazine, IEEE, **(2013)**.

[20] S. Shin, L. Xu, S. Hong and G. Gu, “Enhancing Network Security through Software Defined Networking (SDN)”, 2016 25th International Conference on Computer Communication and Networks (ICCCN), Waikoloa, HI, **(2016)**, pp. 1-9.

[21] K. Ballantyne, W. Almuhtadi and J. Melzer, “Bandwidth aggregation using MPTCP and WMN gateways”, 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Vancouver, BC, **(2016)**, pp. 1-6.

[22] ONF White Paper on SDN, "Software-De ned Networking : The New Norm for Networks",available at https://www.opennetworking.org/images/stories/downloads/sdnresources/white-papers/wp-sdn-newnorm.pdf, **(2012)**.