University of Grenoble Alpes

Internship Report

Exploit of SDN security

Author:
François Richard

June 29, 2018

in partial fulfilment of the requirements of teh "Licence Professionnelle Réseaux Informatiques Mobilité Sécurité" program Institut Universitaire de Technologie – Université Grenoble Alpes

Acknowledgements

Before beginning I want to thanks a few persons that made this wonderful experience possible. I firstly want to thanks Mr Jean-Marc Thiriet that helped me find this internship in Spain. I also want to thanks my internship tutor, Felipe Gil, for always being present and offering to help me if I had some real difficulties during my project. I want to thanks my university tutor Ms Royer Sandra that always made sure I was okay during my internship. Finally I want to thanks the rest of the team of the Lab 509 that welcomed me in a quiet but friendly way and especially Carlos Garridos that was really welcoming from the very first day and helped lessen the burden of suddenly living abroad.

Contents

1 Introduction

In the context of the WINS bachelor, I did an internship of four months at the University of Vigo in Spain. This was my first experience with a work of searcher and I learned a lot during those four months.

As it was for the validation of my bachelor I wanted to work on a network security related project during this internship. Indeed, as I am interested in doing a Master in Network and Security, I thought it would be a good occasion to get some first-hand experience on a security-related subject. As such, I ended up doing an internship where I had to do research about the security of Software Defined Network (SDN) for possible implementation in a 5G core network in a research group at the University of Vigo.

The University of Vigo is a big university with three campus wich total 24 000 students enrolled overall. My internship was more precisely at the Escuela de Telecommunication in the lab 503. the project of that lab is to propose a dynamic gestion of the user on a 5G network by using the SDN technology to handle all the dynamic change on the network. Since I was interested in the SDN technology it is naturally that I joined the group to work on its security aspect as no one was currently trying to develop that particular aspect.

In this paper, we will first give more details about the research group I joined. Then we will talk about the organization of my project during those four months. After that, we will present the concept of SDN and the tools used during the internship. we will follow through with a short summary of the current security problem of SDN before developing one of them and demonstrate the gravity of this security failure. Finally, we will present some solutions to improve the security in an SDN context and conclude on what was gained during these four months.

2 Work environment

As said previously, I joined the research group of the lab 509 during my internship. The aim of this group is to propose a way for the user to always use the shortest path possible to reach internet. To do that they are using the SDN technology to have a dynamic overview of the network and be able to change in real time the rules of the network if need be. In this context, I ended up working on the security of SDN, in general, to try to improve the

security of their prototype later on. Indeed, as they were mainly focusing on getting a working prototype, security was not their priority and I served as a solution to that problem.

3 My Project

Soon after my arrival, the goal of my objective was defined, trying to find ways to improve the security on SDN. To do that a lot of steps were needed and organizing my tasks became a priority to be able to work properly in an efficient way.

3.1 Planification

When I arrived in March, as shown in the gantt diagram, my first task was to get up to date with the State of the Art for both SDN and 5G. This task was very rewarding on a personal note as I got to read papers about the latest advance concerning both of those technologies. After taking my time to read and understand what I was reading I wrote short summaries of about 5-6 pages about each technology and the interesting point that striked me as promising for future development. Moreover, this work on the current state of the Art for both technology let me realize that it was better to orient my project more toward the security of SDN than the security of the 5G.

This choice was based on the few following points:

- Work on the 5G are still in a very early phase which would make it really difficult for me to try to get results in only 4 month
- Compared to the 5G, it is possible to use simulation software to emulate network using SDN based protocol where as the 5G would require actual equipment not easily attainable
- I felt that working on the SDN would benefit me more on the long term and was more likely to yield result during the internship

For all the above reasons, the rest of my work was centered around SDN technology and that's why, for better understanding of the rest of this paper, the next part will be a quick presentation of the SDN.

Figure 1: Gantt of the project.

3.2 SDN

As said before, the basic idea behind SDN is to separate the control plane from the data plane.By doing that, it permits us to separate the processes like configuration, resources allocation, and traffic management in three separate layers: application layer, a control layer and data layer that can communicate with the adjacent plane thanks to API.

The application layer or plane contains all the network and business applications. It has a very basic view of the network organization transmitted from the Northbound API. Once the applications receive this overview of the network, they communicate their requirements to the control plane. The control plane: he is responsible for the decision concerning the gestion of the traffic on the network. His main component is the SDN controller. This controller will change the rules for traffic forwarding to respond to the need of the application of the application plane. Those rules are then transmitted to the data plane.

The data plane: the data plane is a set of network devices like switches, routers, firewall and so forth. The only job of the data plane is to forward the packet in an efficient way according to the traffic rules he received from the control plane. Those rules are transmitted from one plane to the other by the Southbound API. Currently, the most used SouthBoumd protocol is OpenFlow.

3.2.1 Southbound

Of course, as SDN is still developing, OpenFlow is not the only SouthBound protocol existing. Other protocols that could be used are Extensible Messaging and Presence Protocol (XMPP) QND Cisco OpFlex.

XMMP: share some similarities with the SMTP protocol and is defined in an open standard and follows open systems development to allow interoperability but it is not exempt from weaknesses. It does not guarantee a QoS for the exchange of messages between the XMPP client and server and doesn't allow end-to-end encryption which is a basic security requirement in modern architecture.

Cisco OpFlex: is another Southbound protocol aiming to become a standard policy implementing the language. Contrary to OpenFlow which centralizes all network control functions on the SDN controller, Cisco OpFlex focuses on implementing and defining policies. The reason for that is to

avoid a scenario where the controller scalability and control channel communication from becoming the network bottleneck by leaving some level of decision to the devices by using some legacy protocols. The allows bidirectional communication of policies and networking event as well as monitoring information. This information can then be used to make some adjustments in the network configuration. One of the main advantages of Cisco OpFlx is that it offers a great level of control to the developer for networking control applications and don't rely too much on network vendors.

3.2.2 Northbound

The role of the Northbound protocol is to act as the connection between the application and the control plane. It is a very important part of SDN and as such, it has been actively developed by numerous vendor with the aim of reducing the cost of future IT installation.

Representational State Transfer: or REST follows the software architecture style developed for World Wide Web. The client and the server can be developed at the same or separately and can be from different vendors as well. It is prevalent in a lot of controller architecture as the northbound interface to use with Java API. One of the major drawbacks is that it doesn't show recent event on the network. If a change happened it is necessary to refresh to get the information.

Open Services Gateway Initiative (OSGi): it is a set of specification for dynamic application composition using Java bundles. Those bundles can be installed and updated by a lifecycle API. Among the SDN controller that use OSGi is the OpenDaylight project. OSGi allows the starting, stopping, loading, unloading of Java-based network module without a need to stop then restart the controller.

Intent-Based Approach: this northbound interface allows the application to describe their intent instead of the methods to achieve it. this allows the user or the application to only specify the intent and the SDN controller translate it into low-level configuration commands in the data plane for execution. In this case, the application or the user is not aware of the infrastructure configuration and that allows for more flexibility and automation during application development and the deployment of services. Intent-based Northbound Interface (NBI) is very appealing because it has the following advantages:

- application scalability, the application developer doesn't need any knowl-

edge about the actual network and can concentrate on the application instead

- great portability: As it doesn't need any specifics about the actual network it shows great portability
- Intent-Based directives to the SN help to form a cohesive device configuration to avoid resource conflict
- The management of the network is simplified compared to low-level network resource managing.

However, currently, this solution is not fully developed as it needs a language to translate the user or application intent and multiple projects are in development.

3.3 Software used

During this internship, I was faced with a lot of challenges to hone my skills because I was working with a lot of new concepts. To face and solve those different problems I had to use a variety of software. This part is a quick presentation of each of those different software.

3.3.1 Mininet

Mininet is a program that creates a virtual network with all of its component on a single machine. Thanks to that it is possible to emulate computer, switches and controller by using a single VM. This software was very useful for me because it enabled me to simulate an SDN based network and experiment a bit on the security of this kind of network.

This tool is not always installed by default on a machine and has to be installed with sudo apt-get install Mininet. Once it is installed it is possible to emulate a network with this command: sudo mn

This will create the minimal configuration which includes one OpenFlow kernel switch and two hosts and an OpenFlow controller.

Once the network is emulated it is possible to enter commands to do further tests on it via the same terminal.

mininet ¿ mean that the program is running and it is possible to enter command specific to the program from this point on.

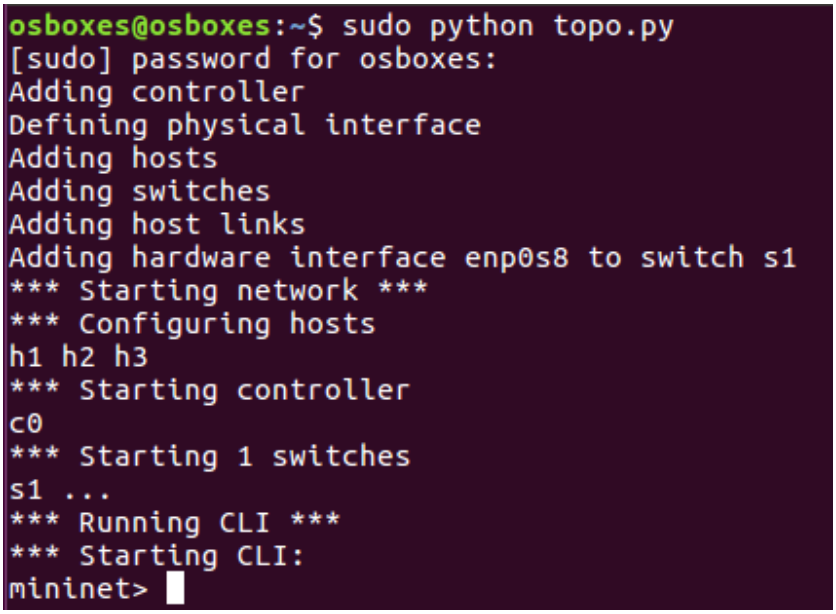

Figure 2: Mininet running normally.

It is also perfectly possible to use another controller not generated by Mininet and it will be the case during this project.

Finally, you also have the possibility to attach physicals interface to one of the switches created, this gives us the possibility to add another machine that will serve for more specific test on the network.

3.3.2 Floodlight

Floodlight is an Open SDN controller. It is a Java-based controller with a huge community still developing it every day. This controller is compatible with Mininet and also possess a web interface which makes it easier for beginner to have a quick overview of all the information.

I chose to use this controller because it is a controller developed by the Open Networking Foundation and as such a lot of resources are available to help the beginner learn how to use it and that let me save some time to focus more on doing actual test.

It is also one of the most used controllers that can be installed with minimum dependency and was designed to be high-performance. Furthermore, the web interface that gives a rapid overview of the network is a real advan-

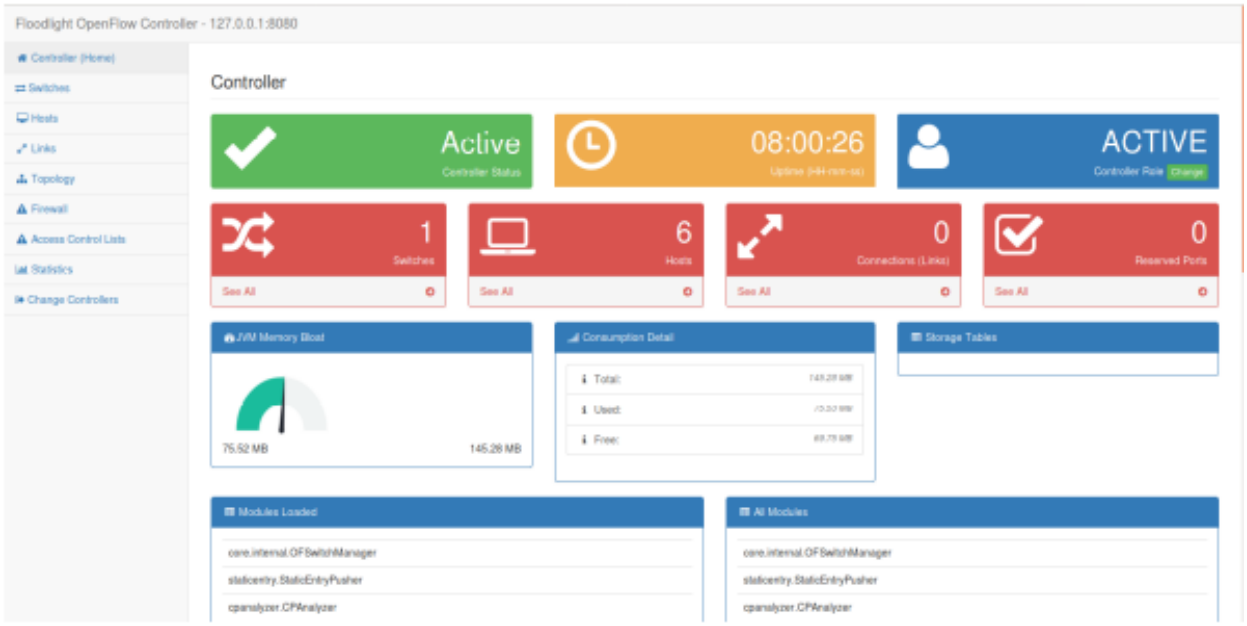

Figure 3: home tab.

tage to understand the state of the whole network with but a single glance. The different tab each gives more detailed information about a specific part of the network.

The home tab is the one that is opened when you arrive on the web-ui (user interface) and if the statistics are enabled (in the statistics tab) it gives a lot of information about the network, number of switches, number of hosts, the percentage of resources used etc...

The switch tab gives information about the switch connected to the controller. In this case, we can see only one switch because for my project I simulated a small network with only one switch.

The host tab gives information about the different hosts of the networks: MAC, IP address (IPv4 and IPv6) to which switch each host is connected to and on which port.

The topology tab is perfect to see a scheme of the current network to better visualize it. Here we can see one switch with all the host connected to it.

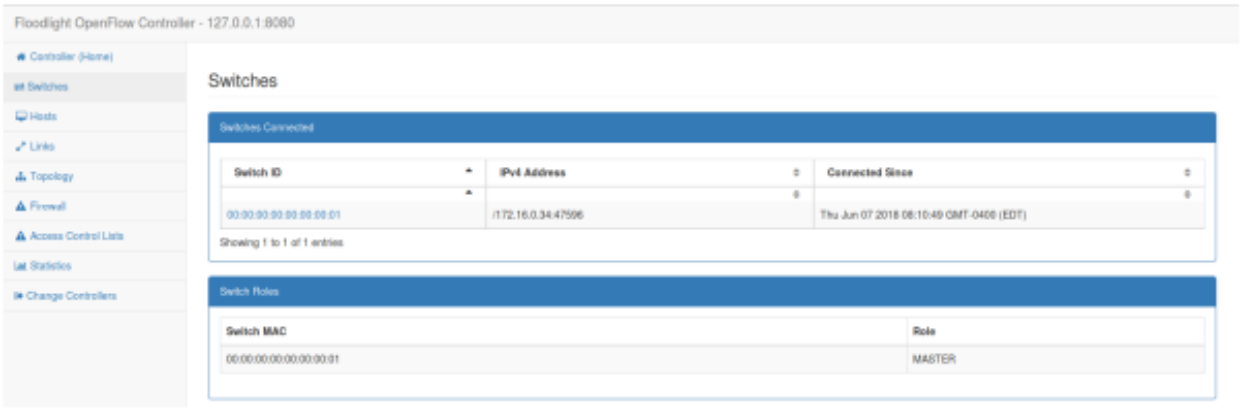

Figure 4: switch tab.

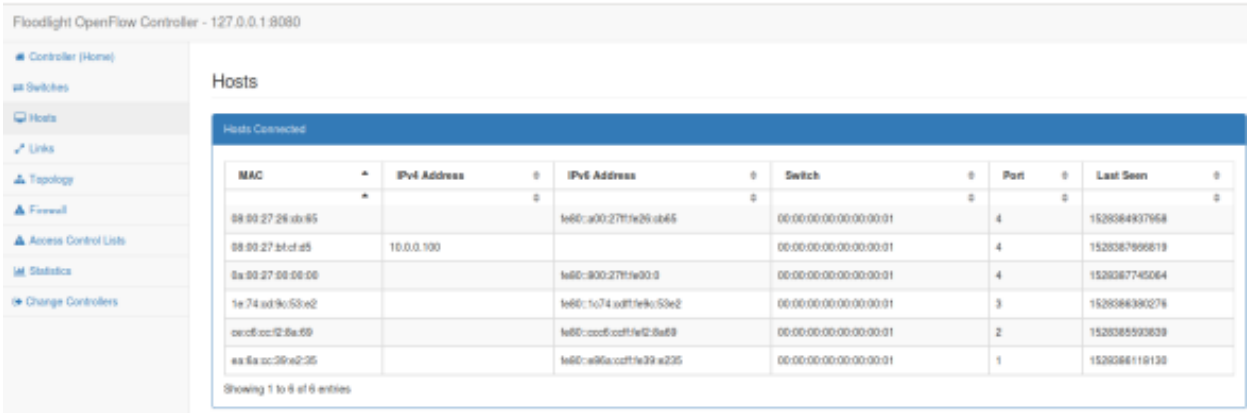

Figure 5: host tab.

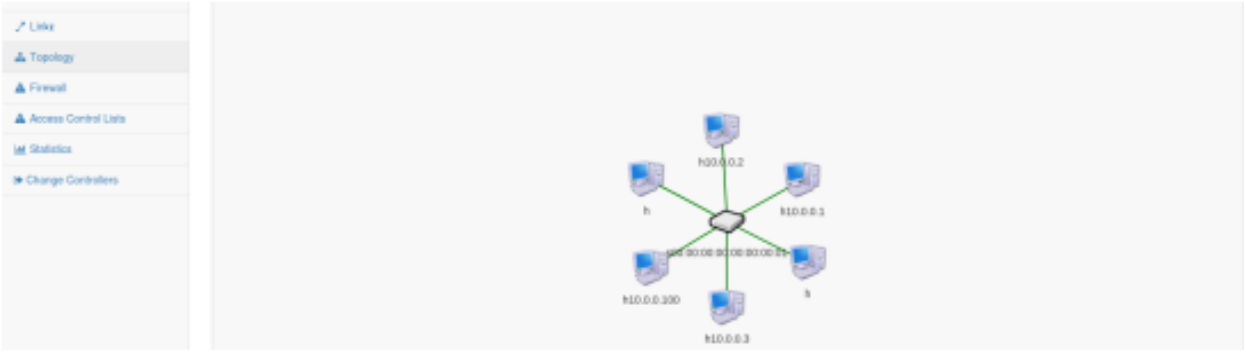

Figure 6: topology tab.

3.3.3 sFlow-RT

Sflow-RT is a sort of monitoring solution that gives information about the flows on the network. It can show the flow going through different machines that send the information in real time. Currently, the sFlow agents are installed by default on a lot of distribution. Once the service is started, a web-ui is available and we can see the information we want represented via graphics. It separates the information very minutely to enable the user to follow exactly the type of data he wants

It is also possible to enter manually the description of the data flow you want to follow, by selecting a few tags, for it, the kind of data and the filters to apply to this flow.

In figure 8, for example, I created a flow called “flow” that take as key values the IP destination, the IP source and the source of protocol and the value for the graphic are defined as bytes and no filter is applied. Once this flow is created on sFlow-RT it is possible to see the traffic corresponding to it in two different ways.

The first way to have more details on the information captured by that flow is to select it in the flow tab.It will then display relevant information for each key selected during the creation of the flow as well as information related to the type of value we associated to it (here the bytes).

On the figure 9 here we can see more details on the IP source and destination as well the stack of protocols used for each case. In this example, I started a ping between two hosts emulated by Mininet and we can see the two IP addresses and the stack of protocols: eth.ip.icmp

This provides more insight than the other way to get information out of

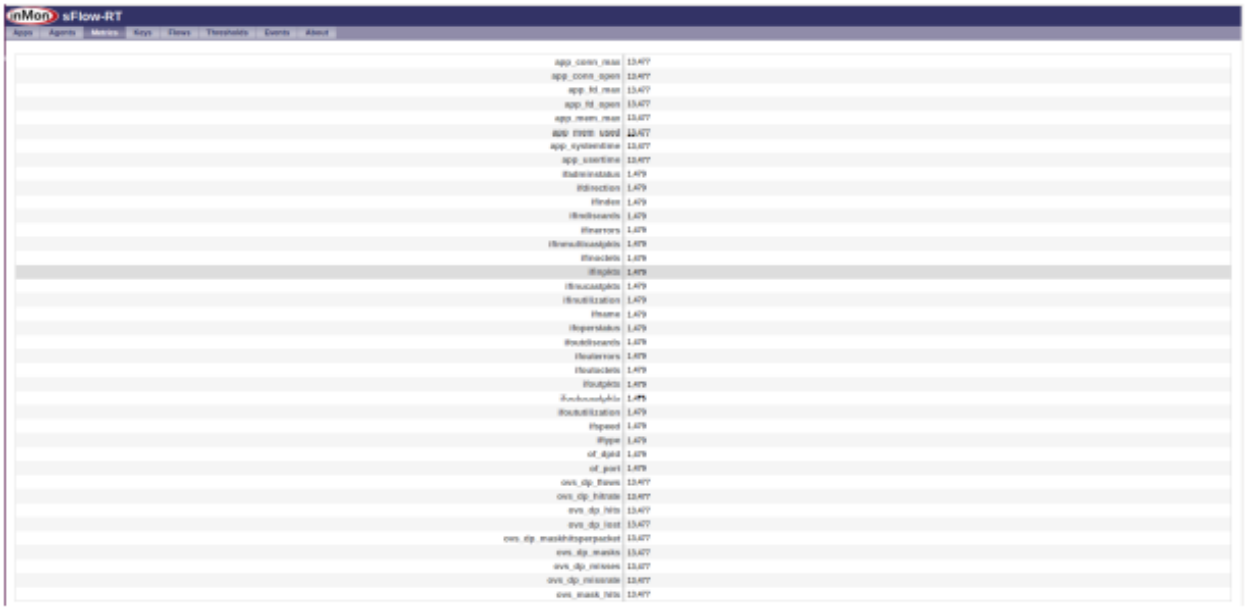

Figure 7: list of flows .

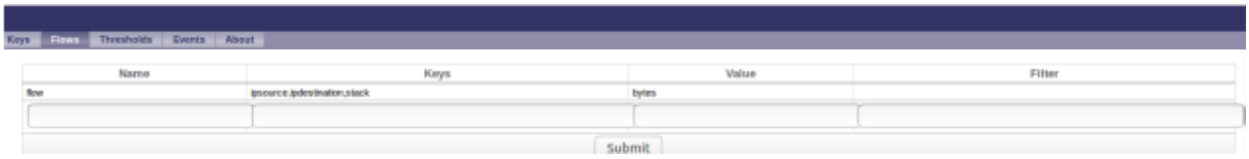

Figure 8: definition of a flow .

Figure 9: definition of a flow .

Figure 10: ICMP traffic in a normal situation.

Figure 11: ICMP traffic during a DoS attack.

the flow through the web-ui.

The second way to represent the traffic concerned by this flow is to go in the metrics tab then select it. This will show us the following graph

On figure 10, we can see the number of bytes concerned on the network at the time. This is very useful to notice if the traffic is abnormal for example with very high peaks that would represent a huge number of bytes of data circulating on the network in a very short amount of time, that could be the indicator of an ongoing DDoS attack like in this case with a simple DoS attack based on a flood of ping between two hosts on the network like on figure 11.

On it, we can see that the number of bytes is very important and drop sharply toward the end which corresponds to the end of the attack..

3.3.4 Wireshark

Wireshark is a network protocol analyzer really useful to see and understand what is happening on the network. It is currently the most widely used

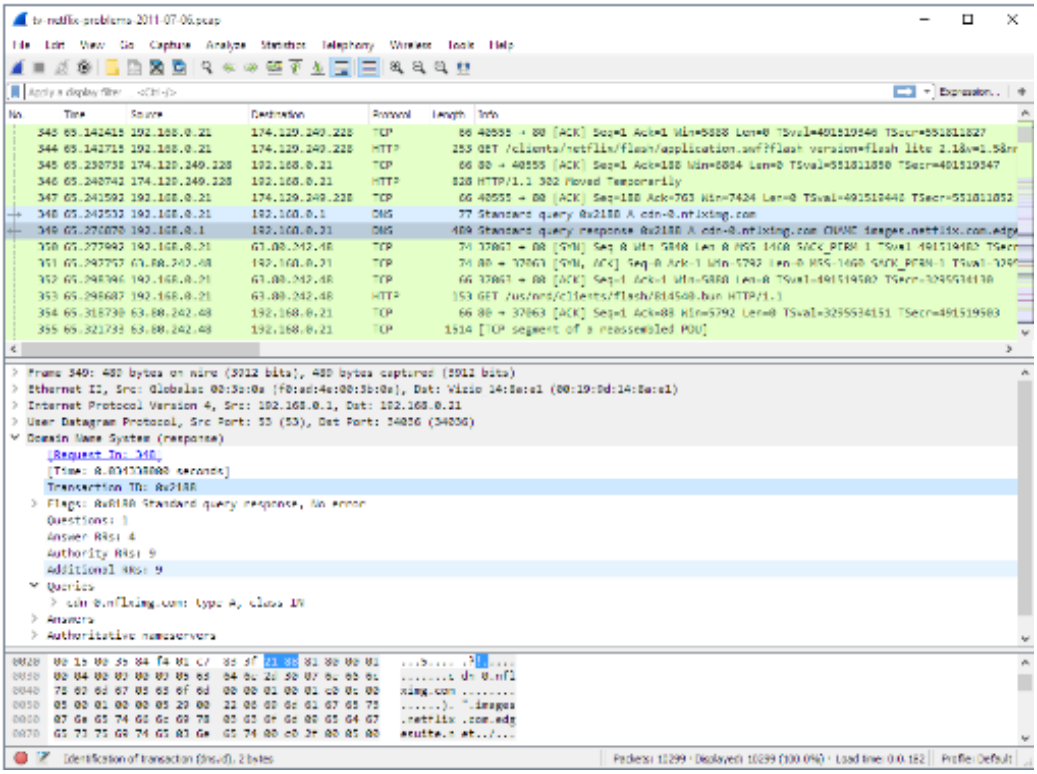

Figure 12: Example of a wireshark windows.

solution to analyze what's happening on the network. We can choose on which interface we want to listen and even apply a filter to see only what we are monitoring.

This option is really to check on what is happening on the network at any given time. It also shine when we need to find the nature of an error.The result are shown like on the prior image with a list of the packet and the option to see the detail of one packet in particular.

3.3.5 Scapy

Scapy is a packet crafting tool that gives a lot of freedom. In most networking tool, the information available to the user is only the information that the creator of the tool needed at the time. Scapy is functioning in a different way. It is a packet crafting tool that let the user create networking packet in the way he wants. With this tool, it is possible to stack networking protocols

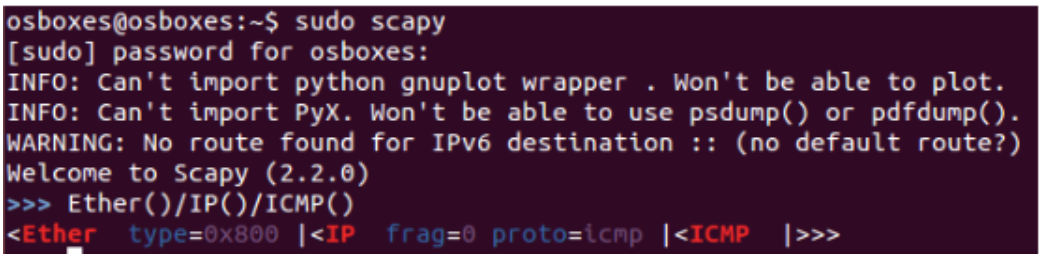

Figure 13: example of a stacking of protocol with Scapy.

in the order that we want.

It also differs from other networking tools in that it gives us the raw information about what is happening with the packet we send on the network. There is no processing by the computer to give a more readable output to the information. The feedback is complete so that no information end up being overlooked and lost for the user. However, the consequence of this way of doing things is that it makes things more difficult for the user as it is not really friendly towards beginners.

This solution is available for Windows and Linux and was used to craft specific packet for the attack on the controller later on in the project.

On figure 13, we can see the interactive interface of Scapy and the output given when we just stack the Ethernet, IP and ICMP protocols.

3.4 Security problems in SDN

Software-Defined Networking (SDN) is an approach to networking that separates the control plane from the forwarding plane to support virtualization. SDN is a new paradigm for network virtualization. Most SDN architecture models have three layers: a lower layer of SDN-capable network devices, a middle layer of SDN controller(s), and a higher layer that includes the applications and services that request or configure the SDN. Even though many SDN systems are relatively new and SDN is still in the early phase of development, once it is widely deployed it will be the target of a lot of attacks. We can anticipate several attack vectors on SDN systems. The more common SDN security concerns include attacks at the various SDN architecture layers.

3.4.1 Security of the Southbound

The first part at risk the network itself o more precisely the elements inside of it. An attacker could theoretically gain unauthorized physical or virtual access to the network or compromise a host that is already connected to the SDN and then try to perform attacks to destabilize the network elements. This could be a type of Denial of Service (DoS) attack or it could be a type of fuzzing attack to try to attack the network elements.

There are numerous southbound APIs and protocols used for the controller to communicate with the network elements. These SDN southbound communications could use OpenFlow (OF), Open vSwitch Database Management Protocol (OVSDB), Path Computation Element Communication Protocol (PCEP), Interface to the Routing System (I2RS), BGP-LS, OpenStack Neutron, Open Management Infrastructure (OMI), Puppet, Chef, Diameter, Radius, NETCONF, Extensible Messaging and Presence Protocol (XMPP), Locator/ID Separation Protocol (LISP), Simple Network Management Protocol (SNMP), CLI, Embedded Event Manager (EEM), Cisco onePK, Application Centric Infrastructure (ACI), Opflex, among others. Each of these protocols has their own methods of securing the communications to network elements. However, many of these protocols are very new and implementers may not have set them up in the most secure way possible. The most used protocols for Southbound API right now are mainly OpenFlow and Open-vSwitch Database Management Protocol that are developed by the Open Networking Foundation.

Another possibility for an attacker is to exploit those protocols and attempt to instantiate new flows into the device's flow-table. The attacker would want to try to spoof new flows to permit specific types of traffic that should be disallowed across the network. If an attacker could create a flow that bypasses the traffic steering that guides traffic through a firewall the attacker would have a decided advantage. If the attacker can steer traffic in their direction, they may try to leverage that capability to sniff traffic and perform a Man in the Middle (MITM) attack.

Furthermore, if an attacker gains access to a switch on the network, he could eavesdrop on flows to see what flows are in use and what traffic is being permitted across the network. From there it is very easy for the attacker to listen to the communication on the southbound interfaces between the switch and the controller. This information can later be used to perpetrate another attack or simply for reconnaissance purposes.

Many SDN systems are deployed within data centers and data centers are more frequently using Data Center Interconnect (DCI) protocols such as Network Virtualization using Generic Routing Encapsulation (NVGRE), Stateless Transport Tunneling (STT), Virtual Extensible LAN (VXLAN), Cisco Overlay Transport Virtualization (OTV), Layer 2 Multi-Path (L2MP), TRILL-based protocols (Cisco FabricPath, Juniper QFabric, Brocade VCS Fabric), Shortest Path Bridging (SPB), among others. These protocols may lack authentication and any form of encryption to secure the packet contents. These new protocols could possess vulnerabilities due to an aspect of the protocol design or the way the vendor or customer has implemented the protocol. An attacker could be motivated to create spoofed traffic in such a way that it traverses the DCI links or to create a DoS attack of the DCI connections.

3.4.2 Security of the Controller

It is obvious that the SDN controller is an attack target. An attacker would try to target the SDN controller for several purposes. The attacker would want to instantiate new flows by either spoofing northbound API messages or spoofing southbound messages toward the network devices. If an attacker can successfully spoof flows from the controller, it basically means that he got a clear view of the network and can bypass any security measure put in place. This is one of the biggest problems with the SDN architecture.

An attacker might try to perform a DoS of the controller or use another method to cause the controller to fail. The attacker might try to attempt some form of resource consumption attack on the controller to bog it down and cause it to respond extremely slowly to Packet-In events and make it slow to send Packet-Out messages.

Most of the time, SDN controllers are run on some form of Linux operating system. If the SDN controller runs on a general-purpose operating system, then the vulnerabilities of that OS become vulnerabilities for the controller. Usually, the controllers are deployed into production with only the default password and no other security settings configured. The fear of creating a malfunction after finally getting it to a working state end up being a problem because a lot of security exploits are left unattended.

Lastly, it would be bad if an attacker created their own controller and got network elements to believe flows from the “rogue” controller. The attacker could then create entries in the flow tables of the network elements and the

network administrator would not have any visibility on those flows from the perspective of the real SDN controller. In this case, the attacker would have complete control of the network.

3.4.3 Security of the Northbound

Attacking the security of the northbound protocol would also be a likely vector. There are many northbound APIs that can be used by SDN controllers. Northbound APIs can use Python, Java, C, REST, XML, JSON, among others. If the attacker is able to leverage the vulnerable northbound API, then the attacker would have access to the controller and through the controller, he'll have access to the whole network. If the controller lacked any form of security for the northbound API, then the attacker might be able to create their own SDN policies and thus gain control of the SDN environment.

Once again, most of the times the password set for the Northbound API is not very secure and easy to figure out after some time. If an SDN deployment don't change this default password and the attacker can create packets via the controller's management interface, then he can also query the configuration of the SDN environment and put in his own configuration.

For my project, I decided to work on Dos attack to the controller from the Southbound to see how well prepared it was against it and how the security on that particular point could be improved upon. However, as the subject is SDN, I tried to find a way to do a DoS attack that would be specific to an SDN environment. The idea that came with this restriction was to find a way to force the switch to send a lot of packet-in to the controller.

To put it simply, in a normal functionment when the network is set up, the controller sends all the flows to the switches. The switch then treats the packets on the network according to the flows that were transmitted by the controller and that form his flow table. However, it is possible that at some point the switch receives a packet that doesn't fit any of the rules of the flow in the flow table. When faced with such a packet, the switch will inform the controller by sending a packet that is the packet-in. When the controller receives the packet-in, he can decide to do two different things. The first option is to create a new flow rule that fit the packet and that will serve as a reference for all future packet fitting those criteria. The other option is to decide to drop the packet. No matter the solution chosen, in both cases, the controller will then send a packet-out to answer the query of the switch.

In this case, the idea is to generate a lot of packets that would not appear

suspicious but that cannot be handled by standard flow rules passed down by the controller. Indeed, generating a large number of such packet will result in a great number of packet-in to be sent to the controller. This will, in turn, reduce the resources available for the controller to deal with other problem on the network and the traffic generated will also impact the fluidity of the network as a whole. Another interesting point of such an attack is that it really impact directly the controller. Since what actually reach the controller are packet-in send by the switch, it will reach it even if some secure protocols like TLS are configured for the exchange between the switch and the controller.

After the base idea was found, it was necessary to find ways to simulate the network I needed and how do the attack on the controller. Speaking about the controller, I also had to choose one to work with from the many available.

3.5 Creation of the testbed

To do my test I first had to create a proper testbed. A testbed (or test bed) is basically an isolated environment in which we can run a number of tests as we like. For this test bed, a few things were required:

- a software to use virtual machines
- a way to emulate a whole network that would use OpenFlow protocol
- a controller
- a solution to enact the attack in various precise ways

For the software used to virtualize machines, I will not develop more but it suffices to say that I used Virtualbox, one of the two major solutions used currently with Vmware Player.

3.5.1 The network simulation

The first real solution I had to find was the solution to emulate the SDN network. As it was the core part of the project it was important for that problem to be solved as quickly as possible. In this situation, two candidates emerged as possible solutions, namely ns3 and Mininet.

Since I already had some previous experience with ns3 with some lab at the IUT, I decided to use Mininet to take that chance to learn a maximum of new things.

Like ns3, Mininet is a network simulator but compared to ns3 it is more designed to emulate networks with OpenFlow switches but they give the same possibilities in general.

Since it was a new solution for me, before creating the actual test environment for the rest of the project, I had to experiment a bit with Mininet to facilitate the rest of the work. During this phase of experimentation, I was able to notice that Mininet is highly flexible since it is very easy to go from one configuration to another by destroying and building a new one in a matter of second.

Another good point of Mininet is that by default the configuration that will be created will be an SDN network and while it is totally possible to define the controller manually if nothing is explicitly specified, Mininet will emulate a local controller on the Virtual Machine (VM) that host the network. This feature was very useful in the beginning because it gave me the possibility to start working right away.

3.5.2 The SDN controller

During this phase of experimentation, I also got to get more familiar with different SDN controllers before electing one to continue the experience. The first one I used was the POX controller.

POX is an open source development platform for Python-based software-defined networking (SDN) control applications, such as OpenFlow SDN controllers. POX, with NOX is one of the first SDN controllers that was developed and happen to be the controller that is emulated by default by Mininet.

In an effort to get more familiar with the POX controller and the diverse possibility of Mininet, I also started to learn a bit about the Python language to be able to manually modify the configuration however I wanted.

However, after finally being a bit familiar with the functionment of the POX controller, I sadly realized that to do the test that I wanted, it was necessary to host the controller on another VM that would later be linked to the Mininet VM. I also found out that most of the resources I was finding regarding security configuration for SDN were done on other, more recent SDN controller.

This situation made me lose some time and forced me to change the controller I was using and I ended up choosing the Java-based Floodlight controller . The Floodlight Open SDN Controller is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller. It is supported by a community of developers that is still very active. It is an interesting solution for my project because it can handle a mix of OpenFlow and non-OpenFlow networks, meaning it is a hybrid controller that will be the next step in the deployment of SDN solution in companies. Furthermore, it is described by the developing team as a controller designed for high performances.

It is also very easy to deploy and compatible with both physical and virtual machines.

OpenFlow is an open standard managed by Open Networking Foundation. It specifies a protocol through which a remote controller can modify the behavior of networking devices through a well-defined "forwarding instruction set". Floodlight is designed to work with the growing number of switches, routers, virtual switches, and access points that support the OpenFlow standard.

For all the above reasons, I chose to switch to a Floodlight controller. Once that choice was made, I had to relearn how to use it and how I could tweak the configuration of the controller if needed. On the bright side, I quickly discovered that the floodlight controller was clearly more user friendly compared to the Pox controller in that that it had a web-ui (User Interface) available that was very quickly downloaded and installed and that gave an overview of the information that the controller had about the network and presented them in an easy to understand way. However, the "high performance" of the controller came at a cost and appeared to be very demanding on resources and every VM I tried to use to emulate it ended up suffering from random slow peak where the VM would nearly not respond because most of the resources were used by the controller.

3.5.3 The attack solution

After choosing the software to use to emulate the network and the controller that was to be used, it was necessary to find a way to enact the attack on the controller to go through with the entirety of the project.

As the goal of the attack was to be able to pass the packets as normal to generate a lot of packet-in between the switch and the controller, the tool used had to be very specific. After some research, the conclusion that was

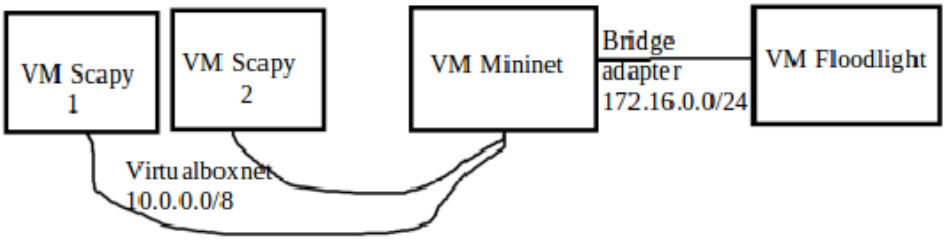

Figure 14: scheme of the configuration.

reached was that the best option would be to resort to packet crafting to be able to specify the type of packet I wanted for the attack. Once, the decision was made, I very quickly started for a packet crafting tool that dealt with the ARP protocol. However, one of the problems that came up was the fact that most of the tools that fitted my requirement were not free. Nonetheless, after further research, I found a solution with Scapy.

Scapy was presented earlier in this paper so I will not dwell too long on it but in short, it is a packet crafting tool that gives a lot of liberty to the user. Not only does it handle the protocols I needed to craft my packet but also a lot of other, which would have been very interesting for further testing if I had more time during the internship.

3.5.4 Putting together the Testbed

Now that the different tools were found, another problem was to make so that they could properly interact with each other. Indeed, for the attack to go smoothly, it was necessary to separate properly each part of the project. As such, the Mininet network, the Floodlight controller and the Hosts with Scapy installed are all on separate VM. This not only served to ensure a clear view of the network but also proved to be an interesting task to do since it also meant that modification had to be made to the Mininet configuration to integrate the other machine in the emulated network. To do that, it was necessary to modify a bit the network configuration of the VM according to the scheme on figure 14 .

To do that, it was also necessary to modify the configuration of Mininet to link it to an interface on the hosting VM. For that, and to simplify the

rest of the project for whenever it was needed to turn off and on the VMs, I prepared the script for the Mininet configuration (figure 15).

With this script, we tell Mininet to use a "remote controller" that corresponds to the controller hosted on another VM. We also fill the IP address of the controller and we can see that it differ from the IP range used in the emulated network (controller: 172.16.0.23 Mininet network: 10.0.0.0/8). Another point of notice is that we added a "hardware interface" which correspond to one of the VM interfaces instead of an actual physical interface.

When we run this script we obtain the figure 16:

Once the script is running on the Mininet VM, we have to prepare the attack on the Scapy VM. The work on this VM was the most difficult out of all the project because I had trouble crafting a packet correctly. This led to a loss of time not planned but turned out to be a very good experience for me. Once a correct packet was crafted, it was necessary to find a way to send a large number of them in a short time. To solve this problem I created another script. This time the job of this script was to create then send a forged packet in the network to generate packet-in for the controller.

After a few days of work, I ended up with figure 17.

If we look more in details at this script, we can see that I installed a loop to create and send ARP packet on the network. It is also important to note that each packet uses a different MAC address as the source. Similarly, they all use a different IP address. These pieces of information are important. Indeed, if all the packet were to use the same information they would quickly be recognized by the controller and only a few packets would be accepted before the controller tell the switch to drop the packet. In such a scenario, no packet-in would be created and the attack would end up as a failure.

Now that the script is created, we only need to run it.In a normal functioning, running the script end up with something like figure 18:

note: it is absolutely necessary to use sudo with this command because some processes used in it need root permission to run properly, without it you'll get an error message.

This is only a part of it but we can still see that packets are sent repeatedly.

To make sure that the packets get the reaction we wanted from the controller it is necessary to use Wireshark on the receiving interface on the controller side. Indeed, if used anywhere else on the network, we will not be able to see the ARP packet as they pass through the Mininet Network and are protected by the Transport Layer Security that makes it impossible to

```
#!/usr/bin/python
"""
Custom topology example
One directly connected switch plus three host attached to the switch with a controller
(c0) and an additional external interface connected to s1
"""

from mininet.net import Mininet
from mininet.link import Intf
from mininet.node import Controller, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.topo import SingleSwitchTopo
# OpenDayLight controller
FL_CONTROLLER_IP='172.16.0.23'
FL_CONTROLLER_PORT=6653
# Define remote OpenDaylight Controller
info( 'Floodlight IP Addr:', FL_CONTROLLER_IP, '\n' )
info( 'Floodlight Port:', FL_CONTROLLER_PORT, '\n' )
def customNet():
    "Create a customNet and add devices to it."
    net = Mininet( topo=None, build=False )
    # Add controller
    info( 'Adding controller\n' )
    net.addController( 'c0',
                       controller=RemoteController,
                       ip=FL_CONTROLLER_IP,
                       port=FL_CONTROLLER_PORT
                     )
    # Add physical interface
    info( 'Defining physical interface\n' )
    intfName = 'enp0s8'

    # Add hosts
    info( 'Adding hosts\n' )
    h1 = net.addHost( 'h1' )
    h2 = net.addHost( 'h2' )
    h3 = net.addHost( 'h3' )
    # Add switches
    info( 'Adding switches\n' )
    s1 = net.addSwitch( 's1' )
    # Add links
    info( 'Adding host links\n' )
    net.addLink( h1, s1 )
    net.addLink( h2, s1 )
    net.addLink( h3, s1 )
    info( 'Adding hardware interface', intfName, 'to switch', s1.name, '\n' )
    _intf = Intf( intfName, node=s1 )

    info( '*** Starting network ***\n')
    net.start()
    info( '*** Running CLI ***\n' )
    CLI( net )
    info( '*** Stopping network ***' )
    net.stop()
if __name__ == '__main__':
    setLogLevel( 'info' )
    customNet()
```

Figure 15: Mininet configuration.

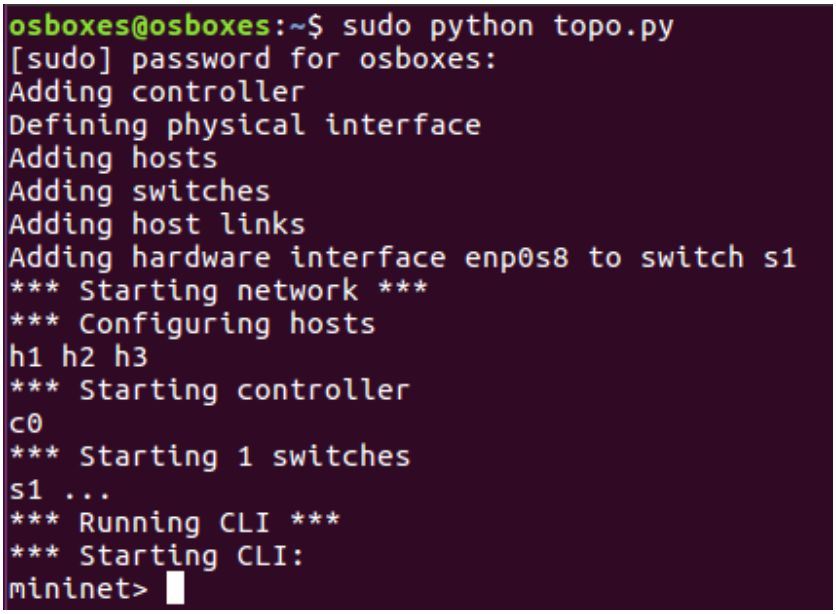

Figure 16: Mininet configuration.

have details about the packet in the middle of the transmission. Of course, we could also use Wireshark on the interface of the attacking VM but it is pointless as the ARP packet are not what is of interest for us right now. Indeed, what we have to check when the attack is running is that the controller does receive packet-in and answer with packet-out packets.

Unless there is an unforeseen problem, at this part of the attack, when the script is running what we see on the Wireshark of the controller is figure 19. On this figure, we can see that there is a strove of packet-in and packet-out circulating at the controller. We can also ensure that those packets involve the ARP protocol with the filter in the upper left corner of the screenshot.

If we look at the details on figure 20 we can see the details of one of those packet-in. We can note the reason for this packet-in is that there is "no match". This means that the switch does not know what to do with this packet so he is asking the controller to choose in his place. In our situation, it's an indicator that the attack is doing exactly what we expected.

In the Data part, we have information about the packet that caused this packet-in from the switch. We can see that it's an ARP packet, more precisely a gratuitous ARP.

A gratuitous ARP packet is an ARP response that is prompted without

```
#!/usr/bin/python

from scapy.all import *
import string
import random

def MAC_Generator(size=2, chars= 'a' + 'b' + 'c' + 'd' + 'e' + 'f' + string.digits):

#Generate random MAC address to use for the attack

   return "".join(random.choice(chars) for _ in range(size))

def IP_Generator():

#Generate random IP address to use for the attack
    part1 = random.randint(1, 255)
    part1 = str(part1)
    part2 = random.randint(0, 255)
    part2 = str(part2)
    part3 = random.randint(0, 255)
    part3 = str(part3)
    part4 = random.randint(0, 255)
    part4 = str(part4)
    addr = part1 + '.' + part2 + '.' + part3 + '.'  + part4
    return addr

def ARP_Packet_Attack():

    x= 0
#Creates the complete packet for the attack then send them

    while (x<=100000):
        #Define each bytes of the MAC address src
        a = MAC_Generator()
        b = MAC_Generator()
        c = MAC_Generator()
        d = MAC_Generator()
        e = MAC_Generator()
        f = MAC_Generator()
        #Define the MAC adresses
        MAC_src = a + ':' + b + ':' + c + ':' + d + ':' + e + ':' + f
        MAC_dst = 'ff:ff:ff:ff:ff:ff'
        #Define the IP address
        IP_addr = IP_Generator()

        #Create and send the packet
        L1 = Ether(src=MAC_src, dst=MAC_dst, type=0x806)
        L2 = ARP(hwdst=MAC_dst, ptype=0x0800, hwtype=1, psrc=IP_addr, hwlen=6, plen=4, pdst=IP_addr, hwsrc=MAC_src, op=1)
        Padd = Padding(load='\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00')

        sendp(L1/L2/Padd)

        x = x + 1

if __name__ == '__main__':
    ARP_Packet_Attack()
```

Figure 17: attack script configuration.

Figure 18: attack script running.

any prior request. It is always sent as a broadcast and usually serves as a way for a node to update his IP or MAC address mapping to the entire network. But it's not because a gratuitous ARP packet is unusual that it is entirely different from a normal ARP packet. Indeed, the Source MAC, type, and Padding still follow the same rule and work the exact same way as before. In the same way, the Hardware and Protocol type, as well as Hardware and Protocol size are the same as in traditional ARP.

We can see that the OP code is set to 2, this means that is to indicate that it's a reply even though this packet was not preceded by an ARP request. The sender MAC and IP contain the ARP mapping that is being sent to the network and are the most important parts of the packet. Finally, the Target MAC is set to broadcast but is actually not taken into account for this type of packet, and the target IP is set to be the same as the sender IP to confirm that this new ARP mapping concern this peculiar IP address.

3.5.5 Getting the controller down

Now that all of the different parts of the test bed are functional and linked together, it's time to start a real attack on the controller. Indeed, the first few tries showed that the current script for the attack was not sufficient to

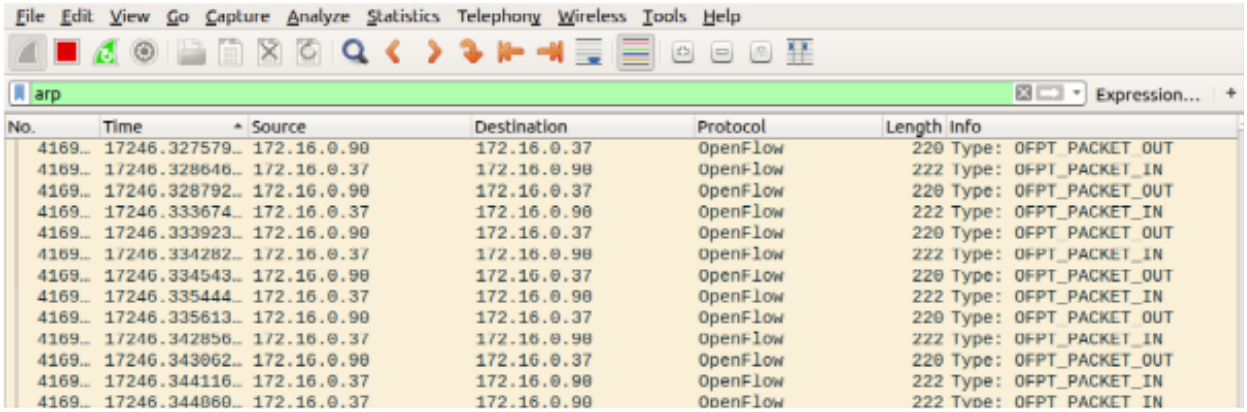

Figure 19: attack viewed from Wireshark on the controller.

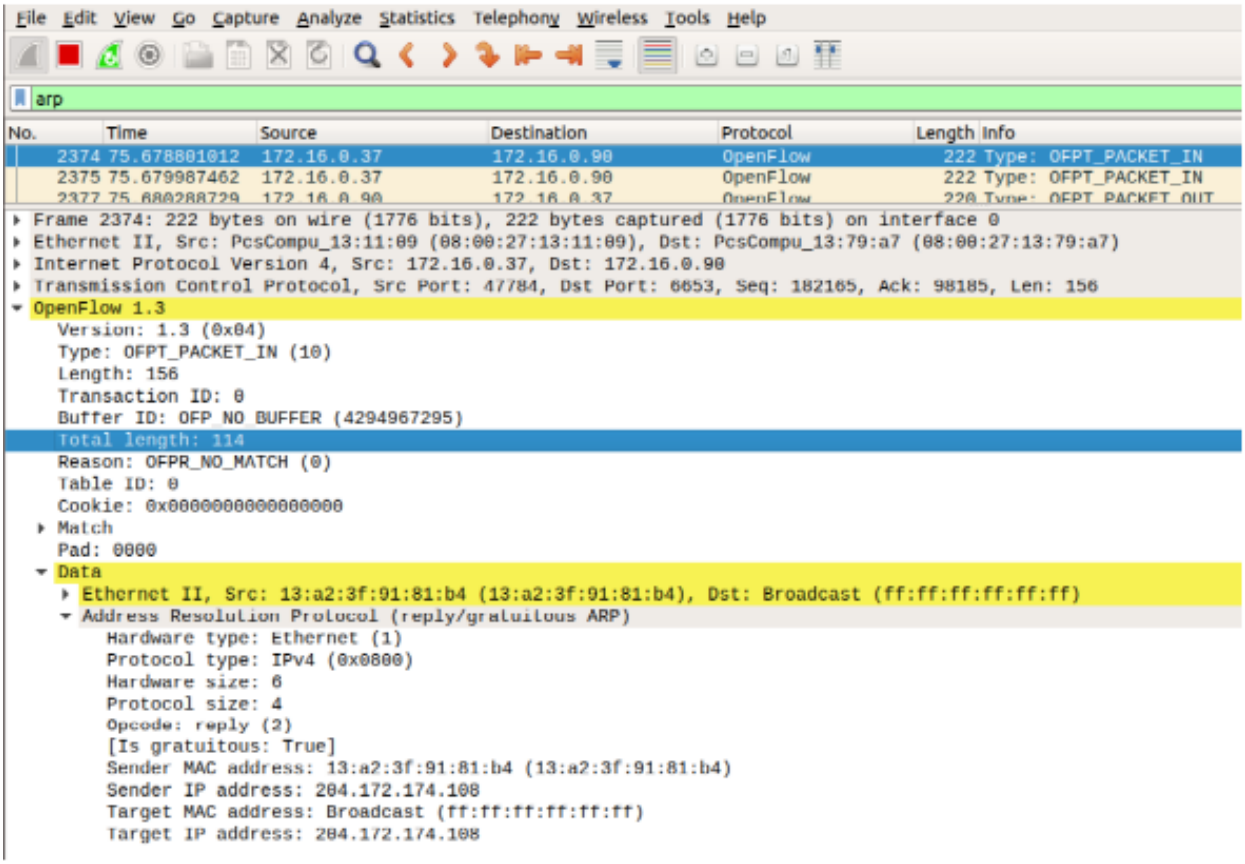

Figure 20: details on one packet-in.

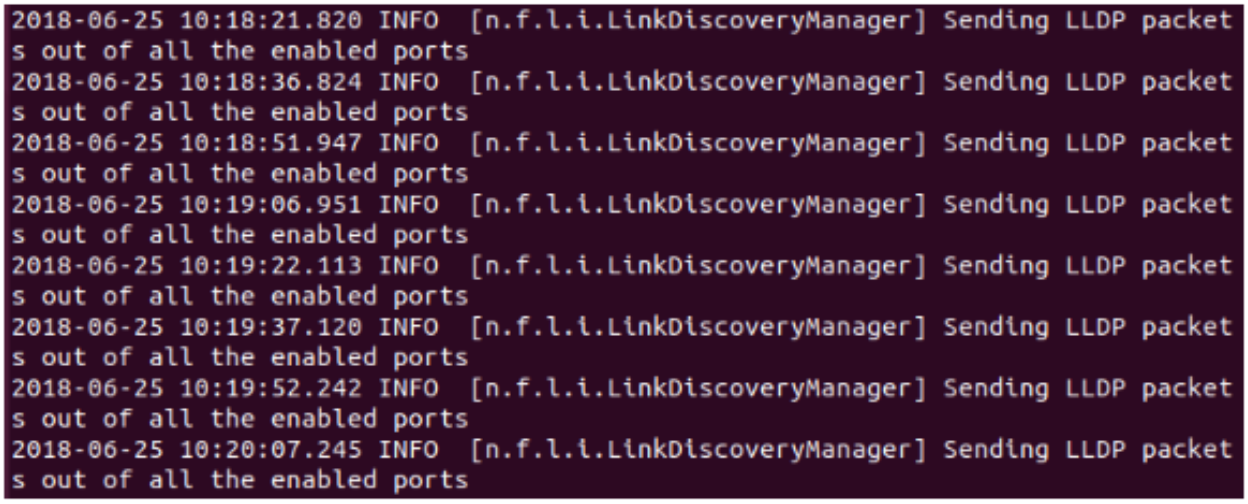

Figure 21: healthy controller.

really impact the Floodlight controller. In an attempt to solve that problem I cloned the attacking VM and launched an attack with two machines at once to see if there was any change.

For future reference, figure 21 represent the logs on a properly running floodlight controller. As we can see, the only notable activity are the regularly send LLDP Discovery packet

After launching the script once on each VM we still see no real consequences, the controller seems to be holding up properly. At this point in time, 2 000 000 of ARP packet were injected into the network, which generated for the controller 4 000 000 of packets (1 packet-out for 1 packet-in).

Since the limits of the controller are still not reached, I quickly restarted the attack script on one of the VM. After some time, we can see some change in the logs of the controller on figure 22. Here we can see that without any apparent reasons, the controller is renewing the connection with the switch. This could be due to an overload on the switch side or be the first signs of dysfunctions in the controller. This idea is reinforced by the following messages that showed up in the logs shortly after in figure 23.

If we look through each line, we can see that the controller is negotiating with the switch to downgrade the version of OpenFlow used. The hypothesis for this downgrading is that it's an attempt to save resources by using a simpler algorithm. However, this solution does not hold out as we can see

```
2018-06-22 06:02:04.757 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out o
f all the enabled ports
2018-06-22 06:02:25.539 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out o
f all the enabled ports
2018-06-22 06:02:46.766 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out o
f all the enabled ports
2018-06-22 06:02:55.575 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /17
2.16.0.37:60364
2018-06-22 06:02:56.568 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /17
2.16.0.37:60368
2018-06-22 06:02:57.592 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /17
2.16.0.37:60370
2018-06-22 06:02:57.594 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /17
2.16.0.37:60366
2018-06-22 06:02:58.602 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /17
2.16.0.37:60372
2018-06-22 06:03:08.279 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /17
2.16.0.37:60376
2018-06-22 06:02:59.662 INFO  [n.f.c.i.OFChannelHandler] New switch connection from /17
2.16.0.37:60374
```

Figure 22: first changes.

```
2018-06-22 04:59:55.38 INFO  [n.f.c.i.OFChannelHandler] Negotiated down to switc
h OpenFlow version of OF_13 for /172.16.0.37:56200 using lesser hello header alg
orithm.
2018-06-22 04:59:55.907 INFO  [n.f.c.i.OFChannelHandler] New switch connection f
rom /172.16.0.37:56202
2018-06-22 04:59:58.276 INFO  [n.f.c.i.OFChannelHandler] [[? from 172.16.0.37:56
200]] Disconnected connection
2018-06-22 04:59:57.447 INFO  [n.f.c.i.OFChannelHandler] [[? from 172.16.0.37:56
196]] Disconnected connection
2018-06-22 04:59:55.908 INFO  [n.f.c.i.OFChannelHandler] Negotiated down to swit
ch OpenFlow version of OF_13 for /172.16.0.37:56198 using lesser hello header al
gorithm.

Exception: java.lang.OutOfMemoryError thrown from the UncaughtExceptionHandler i
n thread "pool-7-thread-4"

Exception: java.lang.OutOfMemoryError thrown from the UncaughtExceptionHandler i
n thread "pool-7-thread-3"
Exception in thread "pool-7-thread-1" java.lang.OutOfMemoryError: Java heap spac
e
```

Figure 23: first errors.

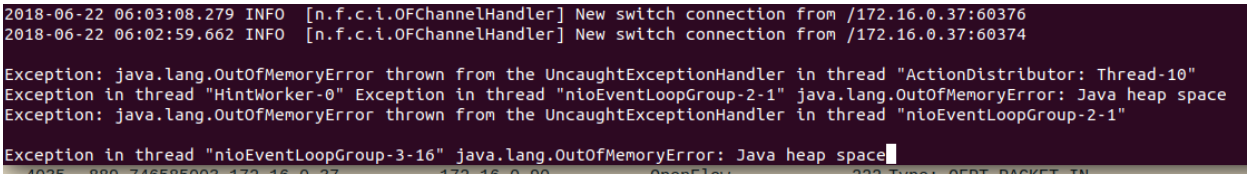

Figure 24: more and more errors.

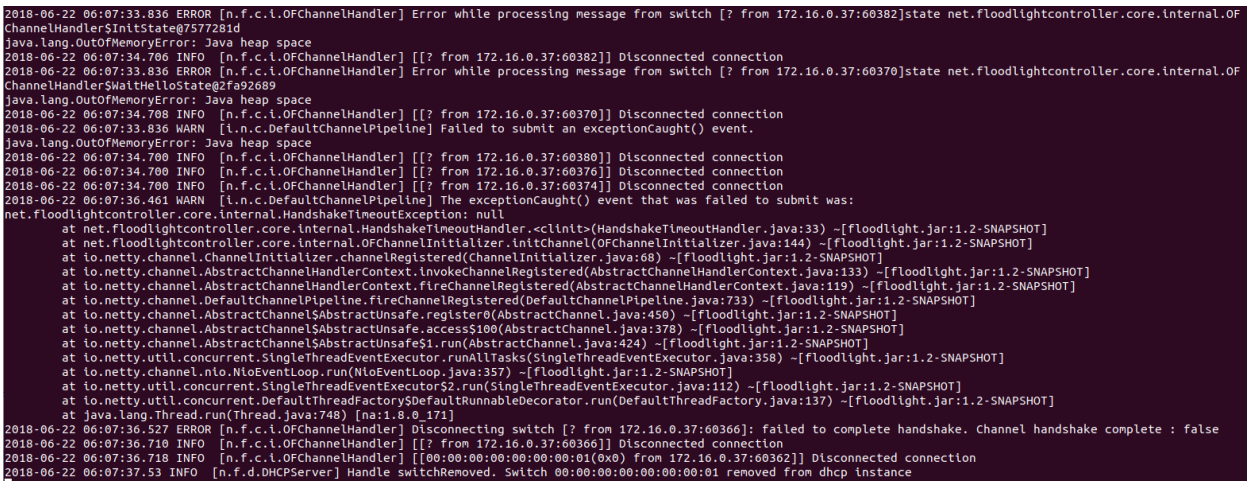

Figure 25: more and more errors.

with the error messages. If we read those error messages, we see that the controller is starting to be out of memory. This means the attack is finally having some consequences on it and at this point in time, it is actually very close to the breaking point. To reach sooner this breaking point, I ran the script once more on the second attacking VM. This very quickly ended up showing other error messages in the logs of the figure 24.

On this image, we can see that the controller tried to renew with the switch. We can also note that it consider the switch as an entirely new switch. However, the memory problem once more prevents it from doing anything.Finally, the controller breaks down and completely sever the connection in figure 25.

On this figure, we can see that there are errors while trying to process packets coming from the switch while the connection is constantly lost times and times again. Once more a lot of Java functions are showing errors mainly due to a lack of resources available. At the end of the logs we can see that

the controller even removed the switch from its DHCP instance.

It is interesting to note that Wireshark was running on the controller during this attack and actually stopped showing new packets arriving when the first java exception showed up in the log.

At this point in time, 4 000 000 ARP packets have been injected into the network and twice that amount should have been generated at the controller. However, in actuality, it stopped receiving packet pas the 6 500 000 marks approximately.

To see the full extent of the damage, we then try to do some pings between the different hosts of the network. The result are stricking: every test end in failure. At this point in time evry communication between the machines of the network is absolutely impossible.From this, we can conclude that the attack on the controller impacted the whole network behind it as it seems that it is incapable to function without it.

Indeed, in the context of an SDN based network, the loss of the controller is critical. This is due to the fact that in SDN the routing tables of the switch(es) are replaced by the flow rules table. However, those tables are made to be regularly updated. The aim of this was to have dynamic gestion of the network. While this is, in theory, an excellent idea it also creates such flaws like this one. Since the controller is down due to an attack that used all of his resources the flow tables are not updated anymore and end up being empty. With empty tables and no controller to take control, the network is effectively completely paralyzed and no packet can circulate.

The more worrying part of this attack is that it was not even recognized as one and easily executable with only two virtual machines. In a real scenario, it is easy to imagine that a larger number of machines would gain access to the network then start the attack at once. The controller would be down in barely one or two minutes in such a scenario.

4 Security improvement

We have seen with the previous parts that there is a lot of security problems that are to be solved in the SDN technology. Moreover, my project proved that such problem was very easy to exploit to impact deeply a network. We will now present some possible solution to improve the security in an SDN context.

4.1 Improving the security of OpenFlow

As we have seen earlier, SDN is far from being completely secure and can still be improved on a lot of points. In this attack, we exploited a default in the way the Southbound works and particularly the way the OpenFlow protocol operates. We will now present the potential ways to improve the security of this protocol.

4.1.1 Coping with the single-point failure

In the legacy network, the network intelligence is distributed throughout the whole network between the different machines with the routing tables and forwarding tables of the routers. In an SDN environment, all those information are centralized at the controller. This creates a potential single-point failure where if the controller fails the whole network is compromised. This is exactly what happened in the attack that was done earlier. To deal with this problem it thus becomes mandatory to implement multiple controllers for the network to ensure some sort of redundancy.

4.1.2 Prevent disclosure of information

OpenFlow replaces the old routing and forwarding tables in the switch with flow tables. Those tables contain rules that are matched to flow defined by the controller. Compared to legacy network, the controller doesn't see the network packet by packet but flows on them. This is called flow aggregation and permit to have a globalized view of the network. The problem of this way of doing things is that it is much too open to a man in the middle attack. Indeed, if someone starts listening to the communication between a switch and a controller he could quickly figure out the different flows defined on the network. From this point, he would be able to gain insight on the various part of the network to then formulate an attack. Thankfully it is possible to make it more difficult for the listener by associating one flow with multiple flow rules. This makes it more difficult to interpret the flow and as such offer some protection to the machines generating it.

4.1.3 Mitigate DoS attack

As we have seen previously, the DoS attack in an SDN environment can have disastrous consequences for the network due to the centralization of

information. That's why it is very important to find solutions to deal with this type of attack. One of the ways to do it would be to limit the flow-rate of the packets to ensure that the controller is never flooded with packets. However, this would also impact the quality of services of the whole network as it would require to slow down the whole network. Other solutions that are researched right now are Flow-based IDS (Intrusion Detection System) that are made to work in an SDN environment and that would be able to detect the early sign of a DoS attack to take measure to prevent. In such a case, the limit on the flow rate could be implemented only on the flow concerned and only for the time of the attack. This could be a good compromise to ensure a better security without having a heavy impact on the QoS (Quality of Service)

4.1.4 Good practice

Currently, there is a lot of things that are considered common good practices related to the security of the network that are done in a Legacy network and not necessarily implemented in SDN. We can for examples note that currently, the communication between the controller and the switches are not always encrypted which make it vulnerable to a man-in-the-middle (MITM) attack. In the same veins, there is no authentication of the switches at the controller level. This, in a nutshell, means that any malevolent can impersonate a switch and collect information or poison the network without the controller being aware of it. Implementing authorization certificates and systematically encrypting the communication of the Southbound could help diminish the importance of this threat.

4.2 Improvement of the northbound API

We've covered the Southbound and presented a few solutions to the lack of security for that part. Now, we will see what can be done on another axis of the SDN concept, the Northbound.

The source of most security failures in the Northbound is the API used. Even though a lot of different API exists, they usually are subject to the same kind of security exploits. The first and most common of those exploits was mentioned earlier in the paper and it's the lack of a secured authentication on the API. This, in turn, leads to the attacker gaining access to the API and using it to implement the wrong flows on the controller which then transmit

them to the whole network. If this scenario comes to pass, the attacker can perfectly create flows that would allow all kind of nefarious packet on the network or even configure the network in a way that all packet has to transit through a specific machine that would belong to the attacker. From this point, the attacker can do whatever he wants on the network.

Fortunately, a few simple solutions can help mitigate the importance of this exploit. In a nutshell, it would be to apply all the solutions existing to secure a server that are available currently. Indeed, the crux of the problem is that the API can be hacked then used for malevolent purposes. As such, to solve the problem is to secure all access to the machine hosting the API and that is no different than what is done in a legacy network. Among those methods, we can mention the use of a strong password for authentication as well as certificates and a good configuration of the ports opened on the machine to ensure that only the necessary ones are usable.

4.3 Securing the Controller

Finally, it is time to talk about the most important part to secure in an SDN network, the controller. As we have seen in an early part of this report, everything transit by the controller at some point. As such, the controller is obviously going to be the target of a lot of attacks and needs to be secure. We saw the problem of the single point failure with attacks from the Southbound. The solution to that was simply to have multiple controllers in place on the network.

Another threat to the controller is that someone takes controls of it. This is a tricky as the attempt to take control can come from the Southbound or Northbound alike, either from a connexion attempt from the network monitored or from a corrupted API. However, the solution to such problems is actually the same. To prevent fraudulent connexion to the controller it is necessary to implement a strict authentication policy with certificates to make sure that both parts of the exchange are who they claim they are. Furthermore, like before, all the method used for OS hardening (improving the security of the operating system on a machine) be it closing unnecessary ports or strong passwords are relevant for the controller.

However, there is another problem that can also an impact on the security of the whole network that involves the controller. The short explanation of this problem would be: the controller currently is only able to implement the flows he receives from the Northbound API one after another. Furthermore,

the most recent one end up having the priority because the aim of SDN is to offer a dynamic monitoring of the network. If one were to only look at this way of doing things it would appear to be completely fine and perfectly aligned with the objectives of an SDN based network. In reality, this poses a huge problem on the security side of the network if this problem is not quickly addressed.

The reasoning is the following. The controllers and the API available right now are not capable of differentiating the priorities between the different protocol. This means that it is possible for a flow designed to handle one application to take precedence over an older flow meant for the security application used by the administrators. In the current situation, this is a huge problem and a potential point of entry for attackers.

Indeed, it means they could create flows that would allow a nefarious application that would block some process used by monitoring solution on the network. The difficulty in this situation is that the problem lies in the way that SDN was imagined. It is touching one of its core concept which is the dynamic gestion of the network. There is currently no real solution to this problem even if researches are done on this aspect.

4.4 Summary

As we have seen, there is still a lot of security problems in SDN and no doubt that there is still a lot of other exploits that will be found the further it is developed. However, at the same time we saw that solutions exist to cope with most of those problems and even if these solutions are not perfect they can still mitigate the threat to the network.

In the end, we can implement every secure solution possible but we can only try to anticipate what attackers would do and it is not impossible that they still find a way to break in. As said before the concept is new, the protocols used are new and there is room for a lot of improvement still.

5 Conclusion

We will now enter the final part of the report. This part will be a summary of what was done during the internship and my thought on it. This will also serve to talk about what I gained from this internship, be it on the technical or personal side.

5.1 end of the project

When I started this internship, one of the things I told myself was that I didn't want to spend four months without achieving any result. As I was given the chance to work on a very interesting subject with a lot of liberty on the way I wanted to do things I felt that it would be a shame to leave without any trace of what I have done. Since the security of networks is a subject I feel is both important and interesting I was tasked with doing some research on the security of SDN. While the subject is pretty vague it means I could approach it however I wanted. This resulted in the project that was presented in this paper. While I did get some result toward the end I still feel that this conclusion is a bit frustrating.

Indeed, I think that if I had more time and using the current results I could have done more tests and try to attack using other protocols to get a more definite conclusion on the security problem I exploited. I also lacked time to try to find, propose and test a solution to the problem I put in light and that is still a sore point for me and my biggest regret with this project. However, this project was still a very interesting experience that gave me a lot of things.

5.2 Technical gain

While as said before, in my mind I regret ending this internship so soon, I still learned a lot during those four months. First of all, I am a lot more familiar with the SDN technology now than I ever was in February. Without declaring myself as an expert on the subject I can confidently say that I have some real knowledge and familiarity with the subject and a lot of different concepts or technology surrounding it. This will no doubt prove useful in the future as I stay rooted in my belief that SDN is the future of the network and will soon be widely deployed.

During the project I also learned how to use a variety of tools that can be used either in an SDN or Legacy network. These new pieces of knowledge came to reinforce my practical skills for network related matter. Furthermore, while gaining new skills, this internship also enabled me to be more familiar with what we learned in class, especially with some aspects of the network security and the packet analysis as it was absolutely necessary to formulate and implement the attack. It was also the occasion to familiarize myself with the scripting language Python that I used to automate some task for

practicality.

Finally, those four months were a very good experience to improve my project planning skills as well as my autonomy. Indeed, as said before I was given a lot of liberty and had to handle the planning of the task and find an angle of work for my project.

5.3 Personal gain

This internship in Spain was a very enriching experience for me. It helped me improved my technical skills as said before but also let me gain in maturity. Indeed, it was a whole other thing to start living in another country compared to living on my own in France. I also had the occasion to meet some very interesting person while I was there.

Furthermore, this gave me the chance to experience a different culture as well as improve my level in Spanish even if just a bit. Finally, if I had to sum up what I gained from this internship abroad I would say that it broaden my views and if I could I would redo it without any hesitation.

6 REFERENCES

- State of the Art and Recent Research Advances in Software Defined Networking, Hindawi, Wireless Communication and Mobile Computing, 2017
- A Comprehensive Security Architecture for SDN, Zhiyuan HU Mingwen WANG Xueqiang YAN Yueming YIN Zhigang LUO, Alcatel-Lucent Shangai Bell Co., Ltd. Shanghai China, 2015
- SDN Security: A Survey, Sandra Scott-Hayward, Gemma O'Callaghan and Sakir Sezer, Centre for Secure Information Technology (CSIT), Queen's University Belfast, Belfast, BT3 9DT, Northern Ireland
- Enhancing Security in OpenFlow, Capstone Research Project Proposal, April 22 2016
- https://www.networkworld.com/article/2840273/sdn/sdn-security-attack-vectors-and-sdn-hardening.html

A Complementqry technologies for SDN

Network Function Virtualization

In a legacy network, it is often needed to chain different dedicated hardware appliance to provide added-value services. This is called Service Function Chaining (SFC) and pose multiple problems. The resources of the different hardware are not used efficiently, the services are disrupted everytime a change is made in the chain and it renders detailed configuration of each hardware complicated.

A way to simplify this chain is to connect every element of the chain together through an OpenFlow switch. After that, it is possible to direct the traffic of each user to the proper set of service. This permit different users to have different service chains. However, difficulties still exist due to the different specific hardware needed for different network functions. The Network Functions Virtualization (NFV) was thought of as a solution to the problem created by the existence of the large variety of proprietary nodes and hardware appliances which causes problems during the launch of new services (too complex, too long). It is composed of four main components:

- Fast standard hardware
- Software-based network functions: Network functions that were running on the dedicated machine are now software image and can run on any standard server.
- Computing virtualization: include every advantage of virtualization(scalability, mobility, quick installation etc...)
- Standard API

NFV enables us to run any network function on a standard hardware, thus resolving the problem of dedicated hardware. It can also be coupled with computing virtualization to ensure an efficient use of the resources.

To sum up, the advantages of NFV are the virtualization which permits us to use the resources no matter where they are physically located, it makes it easier to manage a lot of devices, the configuration can be changed to suit different needs and it can be automated too. It also facilitates the monitoring of the resources being used and help to optimize the utilization of the various network device.

Even though a point-to-point Ethernet can be enough in some case, as NFV requires a flexible gestion of the traffic in the network it shows a great compatibility with SDN even if the two can be used separately.

Software Defined Radio

Software Defined Radio is a different technology than SDN but they share the concept of a "software-defined" feature which translates as their huge capability of adaptation and reconfiguration.

The aim of SDR is to overcome the disadvantages of radio hardware: expensive, too many components, compatibility problem between different manufacturers and more and more lack of efficiency in front of the growing demands of customers. The main difference between SDR and hardware radio is that it scans available frequency bands from the user end and it integrates a few connection technologies into one interface. This enables the user to receive signal using different technologies with only one device. From then it suffices that the norm for those technologies are embedded in the chip and it will be able to use the available frequency. That way, we can save some costs and space. SDR also allow more flexible configuration as part of the physical layer are covered and treated as software. Finally, it allows us to better monitor the spectrum.

A possibility that is being developed currently is to use an SDR layer in an SDN architecture for the 5G. The SDR layer would be in charge of ñonitoring the spectrum the frequency spectrum condition and serve as an indicator for the SDN layer to help him decide when some policies have to be changed.

B Summary of the state of the art in 5G

B.1 Introduction

With the growing number of connected devices in the world, the 4G and LTE will slowly become incapable of providing a quality of the services sufficient. The solution to this problem is the 5G. The promises of 5G are the following: it will be faster than 4G (up to 20 times faster), will use higher frequencies and will be able to accommodate a lot more of devices compared to 4G. It will also be better at managing the resources used on the network to save energy and in general, will have a better QoS at any time.

B.2 Challenges for 5G

As said before, the aim of 5G is to handle the growing number of connected devices. This also means that it has the ambition of taking care of a lot of different services on the same network: medical services, connected car, classic internet etc.

This huge quantity of services in itself will prove to be a challenge. Indeed, a huge number of services means a complicated set of rules for the traffic on the network.

Another problem is the spectrum harmonization. 5G is a new technology that will use both low and high frequencies. Some planning is needed to know how the spectrum will be divided. Furthermore, with the use of higher frequencies who have weaker propagation capabilities and the need for growing density in the network to meet the requirements for the quality of service, 5G will have to create a network with increasingly small cells.

B.3 Solution

To face those challenges, solutions exist. Of course, as 5G is still a technology in development, a lot of different models exist and each tries to meet the requirements for those challenges.

B.3.1 The User-centric Ultra Dense Network (UUDN)

The aims of UDN is to provide a very high data rate for users at indoors and hotspot area. The main difference between UDN and the traditional

cellular network is that the density of AP or BS is much more important in UDN (thousands of AP in 1 square kilometer compared to 3 to 5 BS before). Another keypoint is that user equipment (UE) can be used as AP in UDN.

UUDN is a wireless UDN where the density of AP is comparable to the density of users. The network will then organize the AP in dynamic AP group (APG) to serve each user seamlessly. There are four main features in UUDN:

- An intelligent network who knows the user: The network will be able to automatically detect the terminal capability, the user requirement, and the radio environment.
- A moving network who follows the user: The APG will adjust dynamically to the movement of the user
- A dynamic network at the service of the user: The member of an APG will be adjusted to fit his user's service requirement. This helo enhances the spectrum efficiency and the user experiences a secure network for the user: The network will provide security guarantees with AP authentification when an AP join an APG and UE to network authentification. The result of those authentifications can then be shared and inherited by the members of the APG.

B.3.2 Massive MIMO

MIMO is a system which relies on multiple antennas for the receiver and the transmitter both. With multiple antennas, a greater freedom can be achieved in wireless channels to accommodate more information data. This result in performance improvement in reliability, spectral efficiency, and energy efficiency. In massive MIMO, the transmitter and receiver are equipped with a larger number of antennas (tens to hundreds). The enormous number of receiving antennas can be possessed by only one device or distributed to many. This system obviously inherits the benefit of the MIMO system and further improve the spectral and energy efficiencies. Additionally, in massive MIMO, the effects of noise and fast fading vanish, and intracell interference can be partly solved with linear precoding and some detection methods. Finally, by properly using multi-user MIMO (MU-MIMO) in massive MIMO the medium access control (MAC) layer design can be simplified and the BS can send separate signals to each user by using the same time-frequency

resource. It's all those advantages that make massive MIMO an interesting candidate for future wireless 5G networks.

B.3.3 Spatial Modulation

Spatial modulation (SM) is a new way MIMO technique that has been proposed for low- complexity implementation of MIMO. SM encodes part of the data to be transmitted onto the

spatial position of each antenna in the antenna array instead of simultaneously transmitting multiple data streams to every antenna available. By doing this, the antenna can act as a second constellation diagram and/or increase the data rate via spatial multiplexing. The spatial modulation deal with three major problems n conventional MIMO: inter-channel interference, inter-antenna synchronization and, multiple RF chains. Moreover, low complexity SM receiver can be configured for any transmit or receive antenna in any MIMO system. Right now most studies focus on the case of single-user SM. Multi-user SM could be a new direction for research regarding the 5G network.

B.3.4 Cognitive Network

The cognitive radio network (CR) is an innovative software-defined radio technique with a lot of promises to deal with the congested RF spectrum. The reason to adopt CR is that currently, a large portion of the radio spectrum is underutilized most of the time. In CR, a secondary system can share the spectrum bands with the primary licensed system, on an interference-free basis or on an interference-tolerant basis. The goal is to be aware of the surrounding radio environment to regulate its transmission.

In interference free CR networks, the user can borrow spectrum resources only if licensed users don't use them. The key point in this is to have an effective method to detect the white space that spread out in the wideband frequency spectrum. The receiver first monitors and allocate unused spectrum via spectrum sensing then send this information to the CR transmitter.

In interference-tolerance CR networks, the CR users can share the resources with licensed users if the interference stay below a threshold.- if compared with the interference-free network, the interference-tolerance CR networks shows a better spectrum utilization and a better spectral and energy efficiency. However, it is also more sensitive to changes in the network,

making it less stable.

B.3.5 Mobile Femtocell

The Mfemtocell is a new concept that could be a candidate for the next generation intelligent transportation system. It uses the concept of moving network with the technology of femtocell. As such, a Mfemtocell is a small cell that can move and dynamically change it's connection to operator's network. Deployment of Mfemtocell can potentially be beneficial for the cellular network. Indeed, studies have shown that increasing the percentage of users that communicate with the BS through Mfemtocells leads to a better spectral efficiency on the entire network. Moreover, Mfemtocells can also reduce signaling overhead on the network by, for example performing handover for a group of users within the Mfemtocell, which reduces the handover activities in the cell.

B.3.6 HetNet

The HetNet creates a multi-tier topology where the nodes are deployed with different characteristic such as transmit powers, coverage areas and, radio access technologies. As such, it is a completely different approach compared to the conventional single-tier wireless network and creates a tendency to reduce the cost of the wireless connection in the future. Properly exploited, the opportunities offered by a multi-tier topology could become a core component of the physical layer security. We will discuss this in more details later on in the following sections.

B.3.7 MmWave

MmWave is an innovative solution created to try to meet the requirements of the 5G. It uses a huge swath of spectrum, from 30 GHz to 300 GHz, to shift wireless transmission away from the nearly fully occupied spectral band for current wireless networks. This solution has already been standardized for short-range transmission and deployed for small cell backhaul. However secure mmWave transmission is still a completely new and promising domain to research and will be talked about in more details in a later section.

B.4 Security Issues

Since 5G is still in its development phase, it is obvious that it is far from secure right now. However, at the same time, it also means that a lot of security issues have not been discovered yet. In this part, we will see in more details some security aspect of some of the technologies mentioned before.

B.4.1 The Physical security in HetNet

The HetNet is a network densification architecture showing a lot of promises for the 5G. Its aims are to be a spectrum and energy efficient solution that will satisfy the growth in data demands of the future wireless network. In the HetNet, nodes with different transmit powers, coverage areas, and radio access technologies are deployed to form a multi-tier hierarchical architecture. More specifically, high-power nodes, those with a big coverage area are placed in macro cell tiers while the low-power nodes with smaller coverage area are placed in small cell tiers. The small cell includes pico and femto cells that are to be deployed under a macro cell umbrella to improve the coverage indoor in highly populated areas. In addition to tho those cell, HetNet also has a device tier for device-to-device (D2D) communications. The D2D communications allows two devices who are close to each other to communicate directly with each other with a very low-latency. Among multiple layers, HetNet adapt to its need different radio technologies such as WCDMA, LTE, WiMAX and WLAN to provide his communication services. Since it is a completely different way of conceiving wireless networks its security will require some innovation, in term of spatial modeling, mobile association and device connection to secure those multi-tier communications.

B.4.2 The physical layer security in Massive MIMO

The benefit of Massive MIMO are realized by using a very great number of antennas at the transceiver and/or the receiver and this tendency is on the rise. In future cellular network relying on massive MIMO, the number of antenna at the BS will only increase to answer the need of the users. Since massive MIMO will probably be a key technology in 5G, it is necessary to design the physical layer security based on this technology.

In massive MIMO system, one way to increase security according to studies would be to adopt a low power consumption. Indeed, by reducing the power consumption it is possible to enhance secrecy performance. Since the

transmit power is cut the received SNR at the eavesdropper is highly reduced. This lead to a decrease of the eavesdropper channel capacity.

Another problem that has to be tackled is operation mode. A classic MIMO system generally operates in FDD mode. For the massive MIMO, it is a TDD mode that is recommended. This recommendation is due to the fact that in FDD there are severe limits on the number of antennas. In the TDD mode, as the training burden is independent of the number of BS and the channel reciprocity is used, those limitations are lifted. It also renders the wiretapping of communication more difficult. More specifically, there is no downlink required from the BS to the users in TDD. The user sends uplink channel state information (CSI) via uplinks pilot signals to the BS who then obtain the downlink information by using the reciprocity between the two channel. This makes it more difficult for the eavesdropper to get all the information about the CSI between them and the BS and the CSI between the user and the BS. Of course, this also means that it is necessary to design a secure transmission in case an eavesdropper still manage to get the CSI or part of it. Moreover, there is a need for reciprocity calibration between the users and the BS. As of now, further studies are needed to determine the impact of improper calibration on secrecy performance.

One more measure that was taken to ensure security in conventional MIMO system is the use of Artificial Noise to deteriorate the quality of the eavesdropper channel. In massive MIMO, due to the greater number of antennas, the current use of Artificial Noise could end up being not practical and the artificial noise could also end up being averaged out by the great number of antennas. It is necessary to develop new transmission schemes for Artificial Noises that are adapted to a massive MIMO system.

B.4.3 Physical layer security in Millimeter Wave communication

Current mobile communications system restrict their operation to a spectrum in the range of 300MHz – 3GHz. However, this spectral band is nearly fully occupied. With the millimeter wave communication system, the 5G will be able to operate in the frequency range of 30- 300 GHz. Obviously, security issues need to be addressed to implement this change. The physical layer security is the first step in that direction. Moreover, studies tend to the conclusion that securing the physical layer will be a very rewarding action, and that is due to the following factors:

- Large bandwidth: with the new frequency range available, the secrecy

outage probability in a passive eavesdropping scenario is significantly reduced if the transmitter set a lower transmit secrecy rate

- Short-range transmission: compared to the current communication system, mmWave signal in higher frequencies, which means a severe increase in free-space loss. Therefore, only geographically close eavesdropper are able to pick up the transmission while any remote user is unable to capture the data transmitted.

- Directionality: in mmWave, highly directional communication is employed to suppress the interference from neighbors. This result in a low received SNR which makes it very hard for an eavesdropper to recover any information from overhead messages.

Based on those factors, the aim of the physical layer design is to fully exploit the potential of these factors. Of course, to accomplish that a few challenging problems need to be fixed and a battery of test needs to be done to collect more data on this technology. Finally, it will be necessary to develop new secure transmission schemes.

B.5 Conclusion

We have seen in this paper that the 5G cellular network is still in a phase of development but hold a lot of promise. To meet the requirements a lot of technologies are being developed. If we can properly exploit the potential of those technologies and find an efficient way to use them together 5G network will be much closer to completion. However, new technologies also mean a lot of things are still unknown, including concerning the security of this new network. Indeed, even if some of the new technologies rely on existing one, as they are to be used in a totally different context the current security schemes are not necessarily still relevant. In a nutshell, the 5G currently has a lot of potential but still need a lot of studies before a stable and secure version meeting the requirement is fully developed.