

MANAGEMENT PROJECT

DDoS attacks detection by using SVM on SDN networks

2019

**David Carrascal Adrián Guerrero**

**Pablo Collado Artem Strilets**

INDEX

**INTRODUCTION**3

**MININET AND CONFIGURATION SCENARIO**4

1. Vagrant file installation4

1.1. Troubleshooting problems regarding SSH 5

2. Native File Installation6

**GRAFANA CONFIGURATION**7

**INFLUXDB CONFIGURATION** 7

**TELEGRAF CONFIGURATION** 7

# INTRODUCTION

A network scenario is planned with Mininet. For this, we think of a simple triangle-shaped network in which 2 virtual machines will be used in which this scenario will be implemented, with "Test" and "Controller" being the chosen names.

Therefore, we will need a virtual environment to work. In this way, we will use Mininet at all times to carry out this project.

In addition, we will carry out a monitoring infrastructure that will be based on Grafana, together with InfluxDB and Telegraf, so that we can run the telegraph on the "Test" machine and InfluxDB together with Grafafa on the "Controller" machine. In this way, we will have total control over our network.

Also, we will choose two tools to recreate a DDoS attack (denial of service attack). Being a simple and virtual network, we will generate these attacks ourselves and verify through Grafana how this data is recorded. To do this, we will use "ping" and "hping3", which are two tools whose purpose is the analysis and assembly of TCP / IP packets

On the one hand, using the InfluxDB (Python API) interface, we will create a script that implements an artificial intelligence algorithm that determines whether or not we are under a DDoS attack or in a normal traffic situation through a classification.

On the other hand, we will see how we can import the output of the script that decides if a DDoS attack is being executed in the Grafana panel, to verify it and generate an alarm if it is in this situation.

We have worked using GitHub for the development of the project since we can have a version control and be able to work each without problems, each carrying out a part of the project and sharing it with the rest of the group.

**MININET AND CONFIGURATION SCENARIO**

For the correct operation of our project, we need a scenario that, to mount it, we have previously created a Vagrantfile through which all the scripts necessary for the installation and configuration of each of the machines are provisioned.

When working in a virtualized environment, we ensure that all machines have the same specifications and configuration to help us in the event of an error. Thus, it helps us to have the most localized tracking and correction of errors and we can more easily trace the traceability of their cause.

In addition, in case you do not want to use the Vagrant installation method as a provider, you can follow the native installation method that we present below

### Vagrant file installation

For the installation of Vagrant, we will need to have this program installed in our local machine in order to work. To do this, we will download the executable for Windows from the official website

https://www.vagrantup.com/downloads.html

Once downloaded and installed, we will proceed to install VirtualBox, version 6.0. Although we can currently have higher versions, as it is today 6.1, Vagrant is not implemented for this new version, so we will proceed to download an "old builds" from VirtualBox.

https://www.virtualbox.org/wiki/Download\_Old\_Builds\_6\_0

To have a simpler display, we can use either the console that Windows offers us or download a console emulator, such as Cmder.

After having the main files downloaded, for the configuration and installation of our project, we make a clone of our repository

git clone https://github.com/GAR-Project/project

cd project

In this way, we will have in our local machine the necessary files for the realization of the project.

We enter the “Project” folder and raise the virtual machine

vagrant up

What it does is download, install, configure and start the virtual machine. Once the command is finished, we will have the virtual machine started and ready to work. Now we only have access to these. To do this, we connect to virtual machines using ssh

vagrant ssh test

vagrant ssh controller

We should already have all the virtual machines configured with all the necessary tools to lift our network scenario using Mininet in the virtual machine "test" and Ryu in the virtual machine "controller".

### Troubleshooting problems regarding SSH

If the connection by SSH to the virtual machine is a problem, either "test" or "controller", it should be checked that the keys that are in the path .vagrant / machines / test / virtualbox / have the user as owner, and read-only permissions for the owner of the key. To do this, we simply change the permissions

cd .vagrant/machines/test/virtualbox/

chmod 400 private\_key

# We could also use this instead of "chmod 400" (u,g,o -> user, group,others)

# chmod u=r,go= private\_key

Instead of using the Vagrant administrator to make the SSH connection, we can choose to do it manually by passing the path to the private key using SSH

ssh -i .vagrant/machines/test/virtualbox/private\_key [vagrant@10.0.123.2](mailto:vagrant@10.0.123.2)

### Native File Installation

This method assumes that you already have all the virtual machines running, configured correctly and with the dependencies installed

Ideally you should have 2 virtual machines, where one of them will be the "controller", responsible for running Ryu and the other will be the emulated network of Mininet where we will have our network topology already created

To do this, we will make a clone of the repository:

git clone https://github.com/GAR-Project/project

cd project

For later, to be able to launch the scripts in each machine:

# To install Mininet and Mininet's dependencies. Run it on the "mininet" VM

sudo ./util/provisioning.sh

# To install Ryu. Run it on the "controller" VM

sudo ./util/ryu.sh

**GRAFANA CONFIGURATION**

Grafana is a tool that allows us to monitor, control and analyze any resource in our virtualized network. It is based on open source so we can work with it. In addition, it is a tool that can be installed on any platform, very useful if we want to work on Ubuntu or Windows as is our case.

Thanks to this tool, we can customize the panels and graphics to monitor our network traffic and detect if it is normal traffic or on the contrary it can be a DDoS attack as we will see later and be able to generate alarms to be able to act in case of being The latter case.

In our case, we will work with Grafana together with InfluxDB (which we will talk about later) in the virtual machine “Controller”

For this, what we will do is connect via SSH to our machine "Controller"

But in our case it would not be necessary since the installation will occur during the provisioning phase of the machine which will execute a bash script (in which we write all the commands that we want to execute for the graphane installation)

vagrant ssh controller

Once verified that the service works, we only have to authenticate in the system with the IP of the machine “Controller” through web interface

<https://10.0.123.3:3000>

As soon as we enter the interface for the first time, the system itself proposes that we modify the default password grafana, being User: admin. Password: admin.

After making this change and having logged in, we can now work with grafana

In this image we can visualize graffiti home, in which we can interact with the different icons such as: add a data source from which to collect metrics, create dashboards to monitor them, create alerts for when they exceed a certain threshold, create administrative roles etc.

Next, we will proceed to explain the basic steps for the configuration from scratch to have it functional:

In this configuration we would link to the influexDB database that in our case we have called GAR Project and has an IP address of 10.0.123.3 because it corresponds to the ip of our controller machine in which grafana and influexDB is running with the port 8086 default influexDB

For the realization of this project we have used the database influexDB.

Once the data source is ready, we will proceed to expose our metrics visually using varieties of graphs in which we can make many queries, the one that best suits

our needs

This is an import of our dashboard:

The steps discussed above would be through the graphical interface although it can also be configured through provisioning of files so that once we open for the first time, the database and the dashboard with the metric panels that we want previously exported in format are already configured. .json, so we save time and have a more professional touch.

To configure the data source you would have to copy the file in the path / etc / grafana / provisioning / datasources in .yaml format

After this, we would only have to place the dashboard .yaml file in the path / etc / grafana / provisioning / dashboards and in the path / var / lib / grafana / dashboards its respective .json which will import the panels and queries previously created for the correct study and operation of our Project

**INFLUXDB CONFIGURATION**

Once we have our server set up and ready for monitoring that, in our case we will use the Grafana tool, it will be necessary to have a database (BBDD) in the system to be able to start receiving parameters and host the server values or items that are suitable for monitoring

As far as Grafana is concerned, the most used and well-known database is InfluxDB, since being an open source tool allows us to work with it without problems.

However, in order to use the InfluxDB database, we start from the base that Grafana has it installed in the virtual machine “controller” of the previous section.

Once the database is installed, we proceed to create a database with which we will work on our Project

At this point, we already have in our machine Grafana and InfluxDB together with a database created. Next, we proceed to create a “DataSource” of Grafana that points to this new database to be able to subsequently monitor the parameters we want and host them in this newly created database.

This can be achieved thanks to Grafana's own web interface, from the section

Configuración > DataSources > Add data source

And we add the data we need for proper operation, such as the name, URL (http: // localhost: 8086 since in our case, InfluxDB is on the same server as Grafana) and we write the database we have created previously

After finishing this data, we simply click on “Save & Test” and verify that it has been done correctly using the message “Data source is working”

To finish this part and check that it has been added correctly, we will

Configuración > DataSources

And we check that our Data Source is now available with the name of the database we have created. In this way, the database is correctly configured and ready to receive the elements that we want to monitor within our network

In the next section, we will see how to install and configure Telegraf on servers so that you can send parameters to the monitoring system as well as the creation of dashboards in Grafana to parameterize the views to be monitored

To verify that the database has been created correctly, we enter influex and enter the command show databases

**TELEGRAF CONFIGURATION**

We now start from the base of having both Grafana and the InfluxDB database installed on the “Controller” machine correctly installed and configured.

Next, we will configure the Telegraf agent. For this, what we will do is connect via SSH to our “test” machine

vagrant ssh test

In this way, we will have our server sending information directly to Grafana with InfluxDB database

In the manner discussed above it would be the way to do it command by command but we have done it by means of a bash script (in which we write all the commands we want to execute for the installation) which will be executed during the provisioning phase of the test machine .

After having the machine ready we will proceed to verify that you are running the telegraf service.