Deliverable D3.4
5G-PPP Security Enablers Documentation (v1.0)
Enabler PulSAR: Proactive Security Analysis and Remediation

Project name	5G Enablers for Network and System Security and Resilience	
Short name	5G-ENSURE	
Grant agreement	671562	
Call	H2020-ICT-2014-2	
Delivery date	30.09.2016	
Dissemination Level:	Public	
Lead beneficiary	NEC	Felix Klaedtke, felix.klaedtke@neclab.eu
Authors	TS : Theo Combe	

Contents

1 Introduction

The PulSAR enabler (ex-CyberCAPTOR) provides comprehensive risk analysis on a network through attack graph generation in 5G networks. It relies on topological information of the network (firewall rules, flow matrix, routing tables, virtual machines placement, etc) and vulnerability scan information from physical and virtual machines (Nessus [1], OpenVAS [2], Docker-scanner [3]) to enumerate all attack paths from any machine to any other machine. These attack paths are then scored according to their likelihood and difficulty (using metrics such as their length and the CVSS difficulty of the exploited vulnerabilities) and presented to the user through a REST API. At the time of writing this document, a remediation capability is being developed, to suggest possible actions to take to thwart the most critical attack graphs.

The typical use-case of this enabler is a network in which physical machines and placement / routing are mastered and virtual machine images are known (the vulnerability scan can be performed on cold images, for instance in a local image repository).

The enabler is composed of different programs that call each other:

- A Tomcat webserver that exposes the REST API.
- A java program that computes attack paths, scoring and remediation, and acts as the main program of the enabler. It runs as a Tomcat plugin.
- The MulVAL attack graph engine, using the XSB prolog engine. Its input and output are `Datalog` files.
- A python script that generates an input file for PulSAR from the CSV topology files and vulnerability scans. This script is embedded in the container to load the example topology, but should be called standalone in order to post a new topology to the API.
- A SQLITE database containing the vulnerabilities and scoring (CVE, CVSS, CWE) extracted from the NIST database [4].

2 Installation and Administration Guide

2.1 System Requirements

The PulSAR enabler is packaged in a Docker container. It has been tested for small network instances on an Ubuntu 16.04 machine with Docker 1.10.3 (using the `devicemapper` storage backend). The container embeds all dependencies, so it is expected to run out-of-the-box.

The amount of RAM and hard disk needed for PulSAR can be high, according to the network topology. 8GB of RAM and 5GB of hard disk dedicated to the application should be enough for small to medium systems (which is what was used for testing purposes at development time). For medium to big information systems, 32GB of RAM and 30GB of hard disk dedicated to the application may be needed.

2.2 Enabler Configuration

PulSAR's configuration is located in the `config.properties` file in the container, at `/root/.remediation/config.properties`. It can be overridden at container launch adding the switch: `-v /path/to/new/file:/root/.remediation/config.properties`

Here is the `config.properties` embedded in the container:

```
#Mandatory parameters
xsb-path=/opt/XSB/bin
output-path=/root/.remediation/configuration-files/tmp
mulval-path=/opt/mulval
mulval-rules-path=/root/.remediation/configuration-files/rules-with-topology.P
cost-parameters-path=/root/.remediation/cost-parameters
database-path=/opt/cybercaptor/vulnerability-remediation-database.db
python-path=/usr/bin/python
mulval-input-script-folder=/root/.remediation/cyber-data-extract/
host-interfaces-path=/root/.remediation/configuration-files/inputs/hosts-
interfaces.csv
vlans-path=/root/.remediation/configuration-files/inputs/vlans.csv
routing-path=/root/.remediation/configuration-files/inputs/routing.csv
flow-matrix-path=/root/.remediation/configuration-files/inputs/flow-matrix.csv
placement-path=/root/.remediation/configuration-files/inputs/hosts-vms.csv
controllers-path=/root/.remediation/configuration-files/inputs/controllers-
hosts.csv
#vulnerability-scan-path=/root/.remediation/configuration-
files/inputs/scan.nessus
generic-scan-path=/root/.remediation/configuration-files/inputs/scan-
generic.json
mulval-input=/root/.remediation/configuration-files/tmp/mulval-input-generated.P
topology-path=/root/.remediation/configuration-files/tmp/topology-generated.xml
remediations-history-path=/root/.remediation/configuration-
files/tmp/remediations-history.bin
alerts-temporary-path=/root/.remediation/configuration-files/inputs/tmp/alerts-
temp.binalerts-temporary-path=/root/.remediation/cybercaptor-
server/configuration-files/inputs/tmp/alerts-temp.bin
```

This file contains 2 types of parameters:

- Parameters that reference resources the PulSAR server must access inside the container (mulval-path, output-path, etc): these parameters should not be modified.
- Parameters that reference input files that will make the network topology: these filenames start by `/root/.remediation/configuration-files/inputs`: these files may be overridden at container launch by another topology (see the user manual for more information), but the recommended way is to upload the topology through the API once the server is running.

Among these parameters, the vulnerability scan files (`vulnerability-scan-path`, `openvas-scan-path` and `generic-scan-path`) are not mandatory, but at least one of them should be provided to generate the attack graphs.

2.3 Enabler Installation

The container is provided as a `.bz2` archive to be extracted and imported by Docker.

Archive extraction :

Type the command:

```
bzip2 –d pulsar.tar.bz2
```

This will create the file `pulsar.tar`

Container import :

Type the command:

```
docker load –i pulsar.tar
```

This will load the container image. You can check that it was successfully loaded by typing

```
docker images
```

The image is called `pulsar-xxx`.

Container launch :

If you want to run the server in foreground, launch the following command:

```
docker run –ti –p 8080:8080 pulsar-xxx
```

It will redirect the port 8080 of the local machine to the port 8080 of the container, on which listens the PulSAR API server. The local machine port can be changed. If an orchestrator is used, the configured exposed port must be 8080.

If you want to run the server in background, launch the following command:

```
docker run –d –p 8080:8080 pulsar-xxx
```

2.4 Troubleshooting

PulSAR errors appear either in error messages in API call responses, or in exceptions thrown and logged by Tomcat. The main logs of the application are stored in the files:

- `/var/log/tomcat7/catalina.out` (the tomcat log file)
- `/root/.remediation/configuration-files/tmp/xsb_log.txt` (the MulVAL log file)
- `/root/.remediation/configuration-files/tmp/input-generation.log` (the cyber-data-extract log file)

These files are in the container. They can be accessed from the host with the docker exec command:

- `docker exec pulsar-xxx tail -n 50 -f /var/log/tomcat7/catalina.out`
- `docker exec pulsar-xxx tail -f /root/.remediation/configuration-files/tmp/xsb_log.txt`

- `docker exec pulsar-xxx tail -f /root/.remediation/configuration-files/tmp/input-generation.log`

They can also be replaced by host files with the –v switch, to get logs directly on the host.

3 User and Programmer Guide

3.1 User Guide

3.1.1 Topological inputs file specifications

Inputs can be passed to the enabler by 3 different ways:

- Override the topological and scan files referenced in the `config.properties` file at container launch. This requires a container restart.
- Call the `cyber-data-extract` preprocessor to generate the XML topology file, then upload it to the PulSAR API.
- **Manually generate the XML topology file**, according to the specification in section 3.1.6, and upload it to the API. **This allows using any vulnerability scanner.**

`cyber-data-extract` can take several types of inputs to generate the XML topology file. We describe in this section the format of the inputs that are currently taken by the script. `cyber-data-extract` may be extended to take into account new types of inputs.

Note that all CSV files use a semi-colon ';' as separator, as it is done by default with Microsoft Excel.

3.1.2 Topological files

In the rest of this documentation, we use the test topology embedded in the container. This topology contains 3 physical hosts, 4 virtual machines and 2 VNFs (running as simple VMs). There are 2 VMs per host and the orchestrator (running on 'machine4') controls the whole network. Machine1, machine2 and machine3 expose a vulnerable service enabling remote code execution on the network, while host2 and host3 have vulnerability in the hypervisor allowing a malicious VM to execute code on the host.

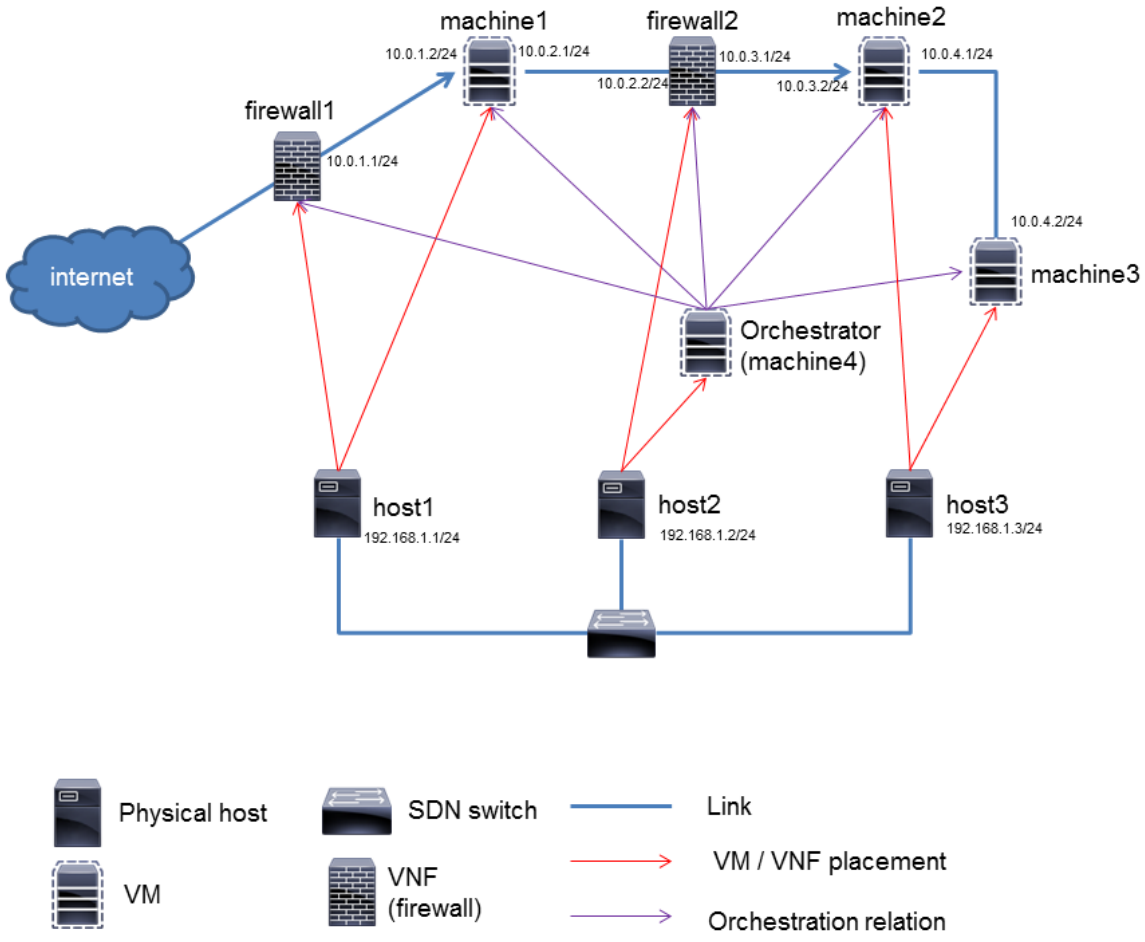

Figure 1 : The example topology

3.1.2.1 Host-interfaces file

This CSV file describes the hosts of the topology, with their network interface.

3.1.2.1.1 Columns explanations

Hostname	Interface Name	IP address	Connected to WAN	Metric
The name of the host (without spaces)	The name of the interface	The IP address of the interface	Whether or not this network interface is connected to WAN (Internet)	A Metric describing the importance of the services running on this IP address.

3.1.2.1.2 Example

```
Hostname;ifName;ifAddr;connectedToInternet;metric
firewall1;eth0;42.42.42.42;True;1
firewall1;eth1;10.0.1.1;False;1
machine1;eth0;10.0.1.2;False;1
machine1;eth1;10.0.2.1;False;1
firewall2;eth0;10.0.2.2;False;1
firewall2;eth1;10.0.3.1;False;1
machine2;eth0;10.0.3.2;False;1
```

```
machine2;eth1;10.0.4.1;False;1
machine3;eth0;10.0.4.2;False;1
host1;eth0;192.168.1.1;False;1
host2;eth0;192.168.1.2;False;1
host3;eth0;192.168.1.3;False;1
machine4;eth0;10.0.5.1;False;1
```

3.1.2.2 Vlans file

This CSV file describes the subnets/VLANS of the network topology.

3.1.2.2.1 Columns explanations

VLAN Name	VLAN Address	VLAN Netmask	VLAN default Gateway
The name of the VLAN	The IP address of the network	The subnet mask CIDR	The IP address of the VLAN default gateway

3.1.2.2.2 Example

```
name;address;netmask;gateway
vlan0;192.168.1.0;24;192.168.1.254
vlan1;10.0.1.0;24;10.0.1.254
vlan2;10.0.2.0;24;10.0.2.254
vlan3;10.0.3.0;24;10.0.3.254
vlan4;10.0.4.0;24;10.0.4.254
vlan5;10.0.5.0;24;10.0.5.254
```

3.1.2.3 Flow Matrix File

This CSV file describes the authorized accesses in the network topology. Note that all accesses that are not specified are supposed unauthorized.

3.1.2.3.1 Columns explanations

Source	Destination	Source port	Destination port	Protocol
The source network (`IP/mask`) or internet	The destination network (`IP/mask`) or internet	The source port or `any`	the destination port or `any`	the protocol or `any`.

Each line describes an authorized access.

3.1.2.3.2 Example

```
source;destination;source_port;destination_port;protocol
internet;10.0.1.2;any;443;TCP
10.0.1.2;internet;80;any;TCP
10.0.2.0/24;10.0.3.0/24;any;5432;TCP
10.0.3.0/24;10.0.2.0/24;5432;any;TCP
```

3.1.2.4 Routing file

This file describes the routes of the hosts that have routes, others than the default gateways of the interfaces' VLAN.

3.1.2.4.1 Columns explanations

Host	Destination	Mask	Gateway	Interface
The name of the host for which this route is specified	The destination network of this route	the network mask of this route	The gateway IP address for this route	The outgoing interface of the route.

3.1.2.4.2 Example

```
host;destination;mask;gateway;interface
router;10.15.10.1;255.255.255.0;10.15.10.1;eth1
router;192.168.1.1;255.255.255.0;192.168.1.1;eth0
router;0.0.0.0;0.0.0.0;1.1.1.1;eth2
```

3.1.2.5 Hosts-vms file

This file describes the placement of virtual machines on physical hosts.

3.1.2.5.1 Columns explanations

Vm	Host	Software	User
The name of the VM	The name of the physical host	The hypervisor software (to be referenced by the vulnerability scan)	The user running the hypervisor software on the host.

3.1.2.5.2 Example

```
vm;host;software;user
firewall1;host1;kvm;root
firewall2;host2;kvm;root
machine1;host1;kvm;root
machine2;host3;kvm;root
machine3;host3;kvm;root
machine4;host2;kvm;root
```

3.1.2.6 Controllers-hosts file

This file describes the control relationships between controllers / orchestrators and hosts (physical or virtual). This relationship expresses the fact that a controller / orchestrator can execute code on the corresponding machine (for instance through VM image modification and restart).

3.1.2.6.1 Columns explanations

Host	controller_global_name
The	The global name of the

name of the machine	controller / orchestrator service, referenced in the services section of its host.

3.1.2.6.2 Example

```
host;controller_global_name
machine1;orchestrateur_global
machine2;orchestrateur_global
machine3;orchestrateur_global
firewall1;orchestrateur_global
firewall2;orchestrateur_global
host1;orchestrateur_global
host2;orchestrateur_global
host3;orchestrateur_global
```

3.1.3 Vulnerability scanner files

Currently, 3 vulnerability scanners can be used: Nessus, OpenVAS and a custom JSON format. For our test topology we used the custom JSON scan format.

3.1.3.1 Nessus scanner files

The outputs of the vulnerability scanner Nessus are stored in a .nessus file, which is an XML file.

The only outputs that are used in this file are:

```
<Report>
<ReportHost name="host ip address or hostname">
<ReportItem port="service port" svc_name="service name" protocol="service proto
col">
<cve>CVE-2015-1234</cve>
<cve>CVE-2015-2345</cve>
</ReportItem>
<ReportItem>
...
</ReportItem>
```

3.1.3.2 OpenVAS files

The outputs of the vulnerability scanner OpenVAS are stored in an XML file.

3.1.3.3 Generic scan file

Other scanner files can be added, provided they are converted to the generic scan file format, stored as a JSON file. This file contains the hosts, services and vulnerabilities information, as follows:

```
{
    "date" : "2016-08-09 11:02:00",
    "hosts" : [
        {
            "name" : "machine1",
            "firstIP" : "10.0.1.2",
            "services" : [
                {
                    "serviceName" : "apache2",
                    "serviceVersion" : "2.2.4-rc10",
                    "serviceProto" : "TCP",
                    "servicePort" : 443,
                    "vulnerabilities" : [
                        {
                            "name" : "CVE-2013-2249"
                        }
                    ]
                }
            ]
        }
    ]
}
```

The “firstIP” field corresponds to the IP of the interface with the default route.

3.1.4 PulSAR Topology XML Input File description

The XML topological file defined here is the main input which is used globally for PulSAR. It unifies all topological data used PulSAR to compute the attack graphs and do the risk analysis.

The main goal of the XML topology file is to describe the network topology, all hosts and their network configuration. Each host can have several network interfaces which can be in different VLAN. The routing and filtering information attached to each host allow computing the network topology (packet route in the network, filtered packets, position of firewalls…). This file can be generated automatically thanks to the cyber-data-extract script.

3.1.5 Description of all fields (xml tags)

3.1.5.1 Machine

Each Machine tag is related to a specific host. This machine tag is used by the Remediation. By way the algorithm is developed, this information is necessary to compute the solutions proposed by Remediation tool.

3.1.5.1.1 Name

- Type : String
- Usage : Contains the name of a host

3.1.5.1.2 Security Requirement (Optional)

- Type : String : NEGLIGEABLE/MINOR/MEDIUM/SEVERE/CATASTROPHIC
- Usage : A value describing a security requirement related to this host.

3.1.5.1.3 Physical host

These XML tags contain attributes related to the physical machine on which the current machine is running. It the current machine is a physical host, this section must be omitted.

3.1.5.1.3.1 Hostname

- Type : String
- Usage : Contains the name of the physical host

3.1.5.1.3.2 Hypervisor

- Type : String
- Usage : Contains the name of the hypervisor program on the host

3.1.5.1.3.3 User

- Type : String
- Usage : Contains the user running the hypervisor program on the host

3.1.5.1.4 Controllers

These XML tags contain the name of the controllers that control the current machine. These names must be global_name properties of services running on controllers.

3.1.5.1.4.1 Controller

- Type : String
- Usage : The name of a controller

3.1.5.1.5 Interfaces - Interface

These XML tags contain all the attributes related to an interface of a machine. Each tag is related to a specific network interface.

3.1.5.1.5.1 Name

- Type : String
- Usage : Contains the name of this interface

3.1.5.1.5.2 VLAN - Name (Optional)

- Type : String

- Usage : Contains the name of the VLAN attached to this interface

3.1.5.1.5.3 VLAN – Label (Optional)

- Type : String
- Usage : Contains the label of the VLAN attached to this interface

3.1.5.1.5.4 IPaddress

- Type : IP address (string)
- Usage : Contains the IP address of this interface

3.1.5.1.6 Services - Service

The description of the network services or applications running on this machine.

3.1.5.1.6.1 Name

- Type : String
- Usage : The name of the service

3.1.5.1.6.2 IPaddress (Optional)

- Type : IP address (string)
- Usage : The IP address on which the service is listenning (if applicable).

3.1.5.1.6.3 Protocol (Optional)

- Type : TCP/UDP/ICMP/ANY (string)
- Usage : The protocol on which the service is listenning (if applicable).

3.1.5.1.6.4 Port (Optional)

- Type : Integer
- Usage : The port on which the service is listenning (if applicable).

3.1.5.1.6.5 Global name (Optional)

- Type : String
- Usage : The global name to identify the service on the network. Is used for instance to identify a controller service in the Controllers section of slave machines.

3.1.5.1.6.6 Vulnerabilities - Vulnerability (Optional)

The vulnerabilities of this service, if applicable.

3.1.5.1.6.6.1 Type

- Type : remoteExploit/localExploit
- Usage : The type of vulnerability (cf CVSS).

3.1.5.1.6.6.2 CVE

- Type : String (CVE-YEAR-1234)
- Usage : The CVE identifier of the vulnerability.

3.1.5.1.6.6.3 Goal

- Type : String
- Usage : The goal of the vulnerability

3.1.5.1.6.6.4 CVSS

- Type : Double
- Usage : The CVSS score of the vulnerability.

3.1.5.1.7 Routes - Route

These XML tags contain the routing table attached to each host. Each tag contains a route of the routing table. Each host needs at least a route containing its default gateway (0.0.0.0/0.0.0.0).

3.1.5.1.7.1 Destination

- Type : IP address (string)
- Usage : Contains the destination network address of the route

3.1.5.1.7.2 Mask

- Type : IP address (string)
- Usage : Contains the network mask of the destination network

3.1.5.1.7.3 Gateway

- Type : String
- Usage : Contains the IP address of the gateway to take for this route (next hop)

3.1.5.1.7.4 Interface

- Type : IP address (string)
- Usage : Contains the interface of the host to use for this route

3.1.5.1.8 Input-Firewall

This XML tag contains the input firewall table attached to each host.

3.1.5.1.8.1 Default-policy

- Type : ALLOW/DENY
- Usage: Contains the default policy of the input firewall table, selected if no firewall line match.

3.1.5.1.8.2 Firewall rule (Optional)

This XML tag contains one line of the input firewall table.

3.1.5.1.8.2.1 Protocol

- Type : String : TCP/UDP/ANY
- Usage : Contains the network flow protocol to match for this firewall line.

3.1.5.1.8.2.2 Source IP

- Type : IP address (string)
- Usage : Contains the source network address to match for this firewall line

3.1.5.1.8.2.3 Source Mask

- Type : IP address (string)
- Usage : Contains the source network mask to match for this firewall line

3.1.5.1.8.2.4 Source Port

- Type : integer or ANY
- Usage : Contains the source port to match for this firewall line

3.1.5.1.8.2.5 Destination IP

- Type : IP address (string)
- Usage : Contains the destination network address to match for this firewall line

3.1.5.1.8.2.6 Destination Mask

- Type : IP address (string)
- Usage : Contains the destination network mask to match for this firewall line

3.1.5.1.8.2.7 Destination Port

- Type : integer or ANY
- Usage : Contains the destination port to match for this firewall line

3.1.5.1.8.2.8 Action

- Type : ACCEPT / DROP
- Usage : Contains the action to do if a packet match this firewall line

3.1.5.1.9 Output-Firewall

This XML tag contains the output firewall table attached to each host.

3.1.5.1.9.1 Default-policy

- Type : ALLOW/DENY
- Usage : Contains the default policy of the output firewall table, selected if no firewall line match.

3.1.5.1.9.2 Firewall rule (Optional)

This XML tag contains one line of the output firewall table.

3.1.5.1.9.2.1 Protocol

- Type : String : TCP/UDP/ANY
- Usage : Contains the network flow protocol to match for this firewall line.

3.1.5.1.9.2.2 Source IP

- Type : IP address (string)
- Usage : Contains the source network address to match for this firewall line

3.1.5.1.9.2.3 Source Mask

- Type : IP address (string)
- Usage : Contains the source network mask to match for this firewall line

3.1.5.1.9.2.4 Source Port

- Type : integer or ANY
- Usage : Contains the source port to match for this firewall line

3.1.5.1.9.2.5 Destination IP

- Type : IP address (string)
- Usage : Contains the destination network address to match for this firewall line

3.1.5.1.9.2.6 Destination Mask

- Type : IP address (string)
- Usage : Contains the destination network mask to match for this firewall line

3.1.5.1.9.2.7 Destination Port

- Type : integer or ANY
- Usage : Contains the destination port to match for this firewall line

3.1.5.1.9.2.8 Action

- Type : ACCEPT / DROP
- Usage : Contains the action to do if a packet match this firewall line

3.1.5.2 Flow-matrix - Flow-matrix-line

This contain all the lines of the flow matrix in this network (all authorized accesses)

3.1.5.2.1 Source - Resource

- Type : String
- Usage : The name of the authorized source resource

3.1.5.2.2 Source - Type

- Type : VLAN/IP
- Usage : The type of the authorized source resource

3.1.5.2.3 Destination - Resource

- Type : String
- Usage : The name of the authorized destination resource

3.1.5.2.4 Destination - Type

- Type : VLAN/IP
- Usage : The type of the authorized destination resource

3.1.5.2.5 Source Port

- Type : Integer
- Usage : The authorized source port

3.1.5.2.6 Destination Port

- Type : Integer
- Usage : The authorized destination port

3.1.5.2.7 Protocol

- Type : TCP/UDP/ICMP/ANY
- Usage : The authorized protocol

3.1.6 APPENDIX: Example topology file

This topology file represents topology from Figure 1.

```
<topology>
  <machine>
    <name>firewall1</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host1</hostname>
      <hypervisor>kvm</hypervisor>
```

```
    <user>root</user>
  </physical_host>
  <controllers>
    <controller>orchestrateur_global</controller>
  </controllers>
  <interfaces>
    <interface>
      <name>eth1</name>
      <vlan>
        <name>vlan1</name>
        <label>vlan1</label>
      </vlan>
      <ipaddress>10.0.1.1</ipaddress>
      <directly-connected>
        <ipaddress>10.0.1.2</ipaddress>
      </directly-connected>
    </interface>
    <interface>
      <name>eth0</name>
      <vlan>
        <name />
        <label>6548</label>
      </vlan>
      <ipaddress>42.42.42.42</ipaddress>
      <directly-connected>
        <internet />
      </directly-connected>
    </interface>
  </interfaces>
  <services />
  <routes />
  <input-firewall>
    <default-policy>ACCEPT</default-policy>
  </input-firewall>
  <output-firewall>
    <default-policy>ACCEPT</default-policy>
  </output-firewall>
</machine>
<machine>
  <name>machine1</name>
  <cpe>cpe:/</cpe>
  <physical_host>
    <hostname>host1</hostname>
    <hypervisor>kvm</hypervisor>
```

```
  <user>root</user>
</physical_host>
<controllers>
  <controller>orchestrateur_global</controller>
</controllers>
<interfaces>
  <interface>
    <name>eth1</name>
    <vlan>
      <name>vlan2</name>
      <label>vlan2</label>
    </vlan>
    <ipaddress>10.0.2.1</ipaddress>
    <directly-connected>
      <ipaddress>10.0.2.2</ipaddress>
    </directly-connected>
  </interface>
  <interface>
    <name>eth0</name>
    <vlan>
      <name>vlan1</name>
      <label>vlan1</label>
    </vlan>
    <ipaddress>10.0.1.2</ipaddress>
    <directly-connected>
      <ipaddress>10.0.1.1</ipaddress>
    </directly-connected>
  </interface>
</interfaces>
<services>
  <service>
    <name>apache2</name>
    <global_name />
    <ipaddress>10.0.1.2</ipaddress>
    <protocol>TCP</protocol>
    <port>443</port>
    <CPE>cpe:/</CPE>
    <vulnerabilities>
      <vulnerability>
        <type>remoteExploit</type>
        <goal>privEscalation</goal>
        <cve>CVE-2013-2249</cve>
      </vulnerability>
    </vulnerabilities>
```

```
    </service>
  </services>
  <routes />
  <input-firewall>
    <default-policy>ACCEPT</default-policy>
  </input-firewall>
  <output-firewall>
    <default-policy>ACCEPT</default-policy>
  </output-firewall>
</machine>
<machine>
  <name>firewall2</name>
  <cpe>cpe:/</cpe>
  <physical_host>
    <hostname>host2</hostname>
    <hypervisor>kvm</hypervisor>
    <user>root</user>
  </physical_host>
  <controllers>
    <controller>orchestrateur_global</controller>
  </controllers>
  <interfaces>
    <interface>
      <name>eth1</name>
      <vlan>
        <name>vlan3</name>
        <label>vlan3</label>
      </vlan>
      <ipaddress>10.0.3.1</ipaddress>
      <directly-connected>
        <ipaddress>10.0.3.2</ipaddress>
      </directly-connected>
    </interface>
    <interface>
      <name>eth0</name>
      <vlan>
        <name>vlan2</name>
        <label>vlan2</label>
      </vlan>
      <ipaddress>10.0.2.2</ipaddress>
      <directly-connected>
        <ipaddress>10.0.2.1</ipaddress>
      </directly-connected>
    </interface>
```

```
    </interfaces>
    <services />
    <routes />
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>machine2</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host3</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth1</name>
        <vlan>
          <name>vlan4</name>
          <label>vlan4</label>
        </vlan>
        <ipaddress>10.0.4.1</ipaddress>
        <directly-connected>
          <ipaddress>10.0.4.2</ipaddress>
        </directly-connected>
      </interface>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan3</name>
          <label>vlan3</label>
        </vlan>
        <ipaddress>10.0.3.2</ipaddress>
        <directly-connected>
          <ipaddress>10.0.3.1</ipaddress>
        </directly-connected>
      </interface>
```

```
    </interfaces>
    <services>
      <service>
        <name>postgreSQL</name>
        <global_name />
        <ipaddress>10.0.3.2</ipaddress>
        <protocol>TCP</protocol>
        <port>5432</port>
        <CPE>cpe:/</CPE>
        <vulnerabilities>
          <vulnerability>
            <type>remoteExploit</type>
            <goal>privEscalation</goal>
            <cve>CVE-2013-1903</cve>
          </vulnerability>
        </vulnerabilities>
      </service>
    </services>
    <routes />
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>machine3</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host3</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <controllers>
      <controller>orchestrateur_global</controller>
    </controllers>
    <interfaces>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan4</name>
          <label>vlan4</label>
        </vlan>
```

```
      <ipaddress>10.0.4.2</ipaddress>
      <directly-connected>
        <ipaddress>10.0.4.1</ipaddress>
      </directly-connected>
    </interface>
  </interfaces>
  <services>
    <service>
      <name>apache2</name>
      <global_name />
      <ipaddress>10.0.4.2</ipaddress>
      <protocol>TCP</protocol>
      <port>80</port>
      <CPE>cpe:/</CPE>
      <vulnerabilities>
        <vulnerability>
          <type>remoteExploit</type>
          <goal>privEscalation</goal>
          <cve>CVE-2011-3192</cve>
        </vulnerability>
      </vulnerabilities>
    </service>
  </services>
  <routes>
    <route>
      <destination>0.0.0.0</destination>
      <mask>0.0.0.0</mask>
      <gateway>10.0.4.254</gateway>
      <interface>eth0</interface>
    </route>
  </routes>
  <input-firewall>
    <default-policy>ACCEPT</default-policy>
  </input-firewall>
  <output-firewall>
    <default-policy>ACCEPT</default-policy>
  </output-firewall>
</machine>
<machine>
  <name>host1</name>
  <cpe>cpe:/</cpe>
  <controllers>
    <controller>orchestrateur_global</controller>
  </controllers>
```

```
  <interfaces>
    <interface>
      <name>eth0</name>
      <vlan>
        <name>vlan0</name>
        <label>vlan0</label>
      </vlan>
      <ipaddress>192.168.1.1</ipaddress>
      <directly-connected>
        <ipaddress>192.168.1.2</ipaddress>
        <ipaddress>192.168.1.3</ipaddress>
      </directly-connected>
    </interface>
  </interfaces>
  <services>
    <service>
      <name>kvm</name>
      <global_name>host1</global_name>
      <ipaddress>192.168.1.1</ipaddress>
      <protocol>ANY</protocol>
      <CPE>cpe:/</CPE>
    </service>
  </services>
  <routes>
    <route>
      <destination>0.0.0.0</destination>
      <mask>0.0.0.0</mask>
      <gateway>192.168.1.254</gateway>
      <interface>eth0</interface>
    </route>
  </routes>
  <input-firewall>
    <default-policy>ACCEPT</default-policy>
  </input-firewall>
  <output-firewall>
    <default-policy>ACCEPT</default-policy>
  </output-firewall>
</machine>
<machine>
  <name>host2</name>
  <cpe>cpe:/</cpe>
  <controllers>
    <controller>orchestrateur_global</controller>
  </controllers>
```

```
<interfaces>
  <interface>
    <name>eth0</name>
    <vlan>
      <name>vlan0</name>
      <label>vlan0</label>
    </vlan>
    <ipaddress>192.168.1.2</ipaddress>
    <directly-connected>
      <ipaddress>192.168.1.1</ipaddress>
      <ipaddress>192.168.1.3</ipaddress>
    </directly-connected>
  </interface>
</interfaces>
<services>
  <service>
    <name>kvm</name>
    <global_name>host2</global_name>
    <ipaddress>192.168.1.2</ipaddress>
    <protocol>ANY</protocol>
    <CPE>cpe:/</CPE>
    <vulnerabilities>
      <vulnerability>
        <type>localExploit</type>
        <goal>privEscalation</goal>
        <cve>CVE-2011-4622</cve>
      </vulnerability>
    </vulnerabilities>
  </service>
</services>
<routes>
  <route>
    <destination>0.0.0.0</destination>
    <mask>0.0.0.0</mask>
    <gateway>192.168.1.254</gateway>
    <interface>eth0</interface>
  </route>
</routes>
<input-firewall>
  <default-policy>ACCEPT</default-policy>
</input-firewall>
<output-firewall>
  <default-policy>ACCEPT</default-policy>
</output-firewall>
```

```
</machine>
<machine>
  <name>host3</name>
  <cpe>cpe:/</cpe>
  <controllers>
    <controller>orchestrateur_global</controller>
  </controllers>
  <interfaces>
    <interface>
      <name>eth0</name>
      <vlan>
        <name>vlan0</name>
        <label>vlan0</label>
      </vlan>
      <ipaddress>192.168.1.3</ipaddress>
      <directly-connected>
        <ipaddress>192.168.1.1</ipaddress>
        <ipaddress>192.168.1.2</ipaddress>
      </directly-connected>
    </interface>
  </interfaces>
  <services>
    <service>
      <name>kvm</name>
      <global_name>host3</global_name>
      <ipaddress>192.168.1.3</ipaddress>
      <protocol>ANY</protocol>
      <CPE>cpe:/</CPE>
      <vulnerabilities>
        <vulnerability>
          <type>localExploit</type>
          <goal>privEscalation</goal>
          <cve>CVE-2011-4622</cve>
        </vulnerability>
      </vulnerabilities>
    </service>
  </services>
  <routes>
    <route>
      <destination>0.0.0.0</destination>
      <mask>0.0.0.0</mask>
      <gateway>192.168.1.254</gateway>
      <interface>eth0</interface>
    </route>
```

```
    </routes>
    <input-firewall>
      <default-policy>ACCEPT</default-policy>
    </input-firewall>
    <output-firewall>
      <default-policy>ACCEPT</default-policy>
    </output-firewall>
  </machine>
  <machine>
    <name>machine4</name>
    <cpe>cpe:/</cpe>
    <physical_host>
      <hostname>host2</hostname>
      <hypervisor>kvm</hypervisor>
      <user>root</user>
    </physical_host>
    <interfaces>
      <interface>
        <name>eth0</name>
        <vlan>
          <name>vlan5</name>
          <label>vlan5</label>
        </vlan>
        <ipaddress>10.0.5.1</ipaddress>
        <directly-connected />
      </interface>
    </interfaces>
    <services>
      <service>
        <name>orchestratord</name>
        <global_name>orchestrateur_global</global_name>
        <ipaddress>10.0.5.1</ipaddress>
        <protocol>ANY</protocol>
        <CPE>cpe:/</CPE>
      </service>
    </services>
    <routes>
      <route>
        <destination>0.0.0.0</destination>
        <mask>0.0.0.0</mask>
        <gateway>10.0.5.254</gateway>
        <interface>eth0</interface>
      </route>
    </routes>
```

```
  <input-firewall>
    <default-policy>ACCEPT</default-policy>
  </input-firewall>
  <output-firewall>
    <default-policy>ACCEPT</default-policy>
  </output-firewall>
</machine>
<machine>
  <name>internet_host</name>
  <cpe>cpe:/</cpe>
  <interfaces />
  <services />
  <routes />
  <input-firewall>
    <default-policy>ACCEPT</default-policy>
  </input-firewall>
  <output-firewall>
    <default-policy>ACCEPT</default-policy>
  </output-firewall>
</machine>
<flow-matrix>
  <flow-matrix-line>
    <source type="INTERNET" />
    <destination type="IP" resource="10.0.1.2" />
    <source_port>any</source_port>
    <destination_port>443</destination_port>
    <protocol>TCP</protocol>
  </flow-matrix-line>
  <flow-matrix-line>
    <source type="IP" resource="10.0.1.2" />
    <destination type="INTERNET" />
    <source_port>80</source_port>
    <destination_port>any</destination_port>
    <protocol>TCP</protocol>
  </flow-matrix-line>
  <flow-matrix-line>
    <source type="VLAN" resource="vlan2" />
    <destination type="VLAN" resource="vlan3" />
    <source_port>any</source_port>
    <destination_port>5432</destination_port>
    <protocol>TCP</protocol>
  </flow-matrix-line>
  <flow-matrix-line>
    <source type="VLAN" resource="vlan3" />
```

```
      <destination type="VLAN" resource="vlan2" />
      <source_port>5432</source_port>
      <destination_port>any</destination_port>
      <protocol>TCP</protocol>
    </flow-matrix-line>
  </flow-matrix>
</topology>
```

3.2 Programmer guide

API usage :

In the following, we assume the container is running and port 8080 of the host redirects to port 8080 of the container, and that all commands are issued on the host.

Initialization calls:

Before using the API to manipulate the attack graph, the attack paths, and the remediations, the PulSAR server must be initialized with the topology and vulnerability scans. The attack graph is then generated calling MulVAL, and all attack paths are computed on initialization. There are 2 ways to initialize the server:

- Initialization through input files inside the container. The corresponding API call is: `curl -c /tmp/curl.cookie http://localhost:8080/cybercaptor-server/rest/json/initialize`
  This will fetch the `.csv` files inside the container containing the topology and the `.xml / .json` file containing the vulnerability scans referenced in the `config.properties` file. By default these file contain a test topology, and can be overridden with the –v switch on the `docker run` command. The detail of these files (located at `/root/.remediation/cybercaptor-server/configuration-files/inputs`) is given in the "input detail" section.
- Initialization through posting an XML file containing all the topology to the API. This file is uploaded with the API call:
  `curl -c /tmp/curl.cookie -X POST -H "Content-Type: multipart/form-data" -F "file=@./topology.xml" http://localhost:8080/cybercaptor-server/rest/json/initialize`
  The `topology.xml` must be generated by the cyber-data-extract program, as a preprocessor. This program takes as input the `.csv` files containing the topology and the `.xml / .json` files containing the vulnerability scans.

Attack graph, attack paths and remediation calls :

Then, the calls to get the attack paths, attack graph or remediations can be used:

Get the number of attack paths:

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-server/rest/json/attack_path/number
```

Note the `-b /tmp/curl.cookie` option of curl, to load the previously saved session cookie.

Get the attack path 0:

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/attack_path/0
```

Get the attack graph:

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/attack_graph
```

Get the remediations for attack path 0:

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/attack_path/0/remediations
```

Get the XML network topology (useful for backups, same format as the XML input file):

```
curl -b /tmp/curl.cookie http://localhost:8080/cybercaptor-
server/rest/json/topology
```

These API calls can be tested with the input files embedded in the container, located at:

```
/data/build/cybercaptor-server/configuration-files/inputs
```

4 Unit Tests

4.1 Information about Tests

These tests describe basic procedures to check if PulSAR is correctly installed and running. They check that all components of PulSAR (Tomcat, MulVAL, cyber-data-extract, and the java core) are installed by making some calls to the API.

4.2 Unit Test 1

This test aims at verifying that the Tomcat server and the PulSAR module are correctly installed and running. This first call to test that the server is available is:

```
curl http://localhost:8080/cybercaptor-server/rest/version/detailed
```

which should return something like :

```
{"version":"4.4"}
```

Otherwise, log files described in section 2.4 should contain relevant information.

4.3 Unit Test 2

A basic test to make sure the enabler is properly installed is calling the ‘initialize’ API call. This test aims at verifying that the PulSAR server can call cyber-data-extract to generate the topology, then MulVAL and generate attack graphs, attack paths and remediations. The test relies on the example data provided in the container.

If the port 8080 of the host was redirected to the API server, the command:

`curl http://localhost:8080/cybercaptor-server/rest/json/initialize`

on the host should return `{"status":"Loaded"}`.

Otherwise, relevant information should be in the log files.

5 Acknowledgements

We would like to thank the FiWare-CyberCAPTOR development team for their work.

6 Abbreviations

5G-PPP	5G Infrastructure Public Private Partnership

7 References

[1] "NESSUS," [Online]. Available: http://www.tenable.com/products/nessus-vulnerability-scanner. [Accessed 22 09 2016].

[2] "OpenVAS," 2016. [Online]. Available: http://www.openvas.org/. [Accessed 22 09 2016].

[3] "CoreOS - CLAIR," 2016. [Online]. Available: https://github.com/coreos/clair. [Accessed 22 09 2016].

[4] "NIST NVD vulnerability feed," [Online]. Available: https://nvd.nist.gov/download.cfm. [Accessed 22 09 2016].