**Assignment Directions**

This assignment aims to train the students in the activities involved in the configuration and testing of firewalls using a network emulator/simulator, which name is IMUNES.

To realize the assignment, the students need to create a group of five participants.

Only one student submits this assignment's answer and any complement files (do not submit VM images!!).

**Identification Form**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **E-mail** |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  |  |
| **5** |  |  |

**PART I – ENVIRONMENT SETUP AND BASIC CONCEPTS**

IMUNES is a realistic network topology emulation/simulation framework based on the FreeBSD and Linux operating system kernel partitioned into multiple lightweight virtual nodes, which can be interconnected via kernel-level links to form arbitrarily complex network topologies.

IMUNES provides an Integrated Multiprotocol Network Emulator / Simulator of IP based networks, which its virtual nodes are multiple network stack instances that are formed through special FreeBSD kernel modifications and Docker containters on Linux. Virtual nodes can be linked either with other virtual nodes or with the physical network interface through simulated links. All virtual nodes share a single place for their application binaries and libraries. The main strengths of this tool are high scalability, performance and fidelity.

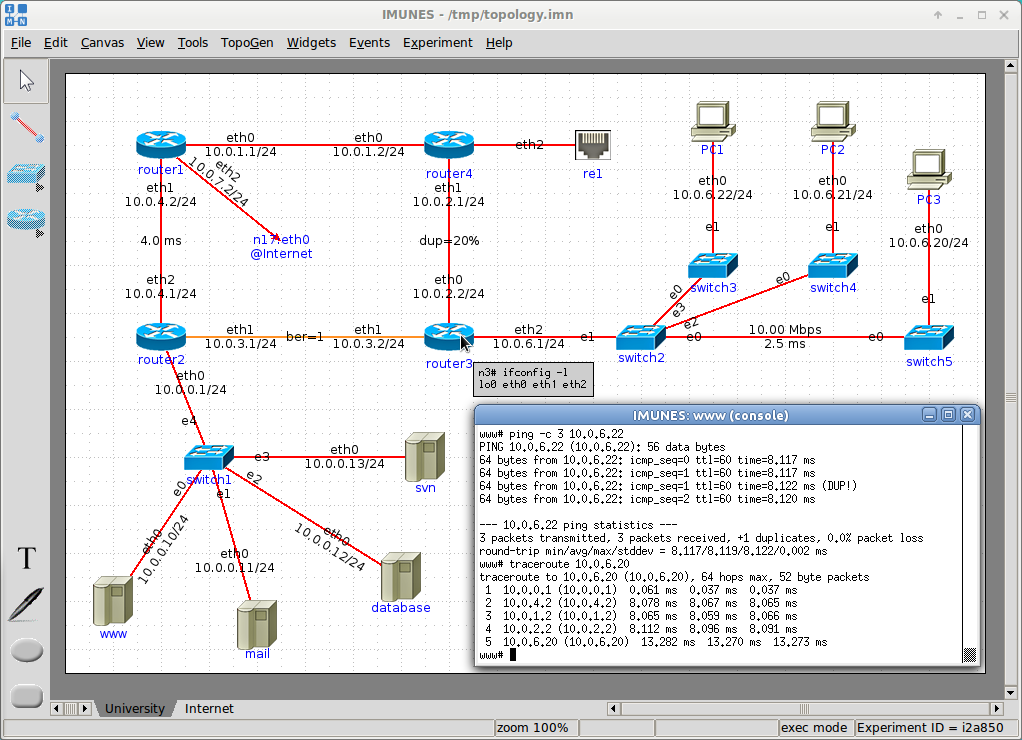


Figure 1 - IMUNES Simulator / Emulator.

IMUNES main caracteristics:

* Real time IP network topology emulation / simulation at Gigabit speeds;
* 100s to 1000s of virtual nodes on one physical machine, each node capable of running unmodified UNIX applications;
* Scalable architecture for real-time large scale experiments;
* GUI: management plane, automated node and link creation and configuration;
* Lightweight, portable experiments which can be quickly and easily instantiated;
* FreeBSD kernel-level network stack virtualization technology developed at the University of Zagreb; and
* Open source and free.

**Task 1: Install the IMUNES**

1. **Requirement Tools**:
   * Oracle VM Virtual Box: [www.virtualbox.org](http://www.virtualbox.org)
   * Ubuntu 18.04 LTS - Desktop (or newest): <https://ubuntu.com>
2. Install the support libraries (into Ubuntu):
   * *apt-get update*
   * *apt-get upgrade*
   * *apt install openvswitch-switch docker.io xterm wireshark make imagemagick tk tcllib util-linux git*
   * *apt-get install build-essential*
   * *reboot*
3. Install IMUNES:
   * *git clone* [*https://github.com/imunes/imunes.git*](https://github.com/imunes/imunes.git)
   * *cd imunes*
   * *sudo su*
   * *# make install*
   * *# imunes -p*
   * *# imunes* (to run the apps and see that the installation’s process worked).
4. Install locally the examples models:
   * *git clone* [*https://github.com/imunes/imunes-examples.git*](https://github.com/imunes/imunes-examples.git)

**Task 2: Read and realize the tutorial: “IMUNES network simulator test drive”**

1. **Support Material**:
   1. <https://www.brianlinkletter.com/imunes-network-simulator-test-drive/>
   2. <http://imunes.net/dl/guide/>
   3. <http://imunes.net/dl/imunes_user_guide.pdf>
2. **Time**: 20 mins.

**Task 3: Configure the basic environment**

* + <https://github.com/imunes/imunes-examples/tree/master/DNS%2BMail%2BWEB>

1. Perform these tasks:
   * *Enter in the setup environment folder: cd imune/imunes-examples/DNS+Mail+WEB*
   * run the follow command to start the imune: *# imunes NETWORK.imn &*
   * Using the IMUNES GUI, starts the simulation:
     1. *Experiment -> Execute*
     2. *Cleanup the picture: select View -> Show ->* 
        1. *de-select "Interface Names" and de-select "Link Lables"*
   * open a diiferent console to run this command:
     1. *# ./start\_dns*
     2. *# ./start\_mail*
2. Check if the configuration works as you planned.
   * Inside the ZPM, on node ‘pc’, open the terminal:
     1. *ping the IP 20.0.0.4*
     2. *ping* [*www.tel.fer.hr*](http://www.tel.fer.hr)
   * Inside the ZZT, on node ‘mn’, open the terminal:
     1. *ping the IP 30.0.0.3*
     2. *ping* [*www.zpm.fer.hr*](http://www.zpm.fer.hr)

**PART II – FIREWALL BASIC CONCEPTS**

In Part I of the assignment, you prepared the virtual environment required to realize the main features of this exercise. Figure 2 presents the final topology of the exercise. The second part of the assignment aims that the students to learn how to create rules in the routers to filter the traffic flow between the zones existent. The firewall application that we will use in this part of the assignment is the iptables.

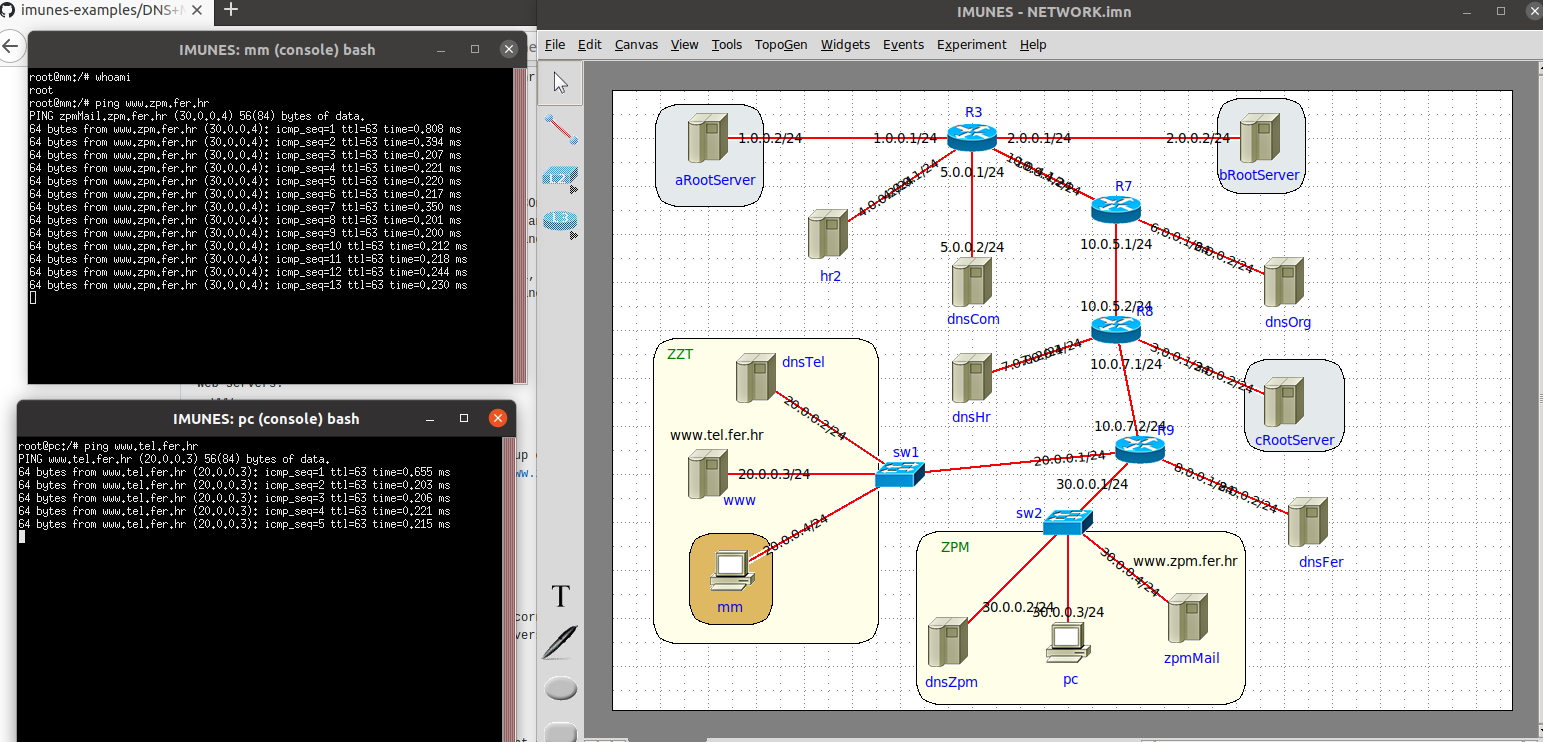


Figure 2 - Assigment Topology

iptables is a command-line firewall utility that uses policy chains to allow or block traffic. When a connection tries to establish itself on your system, iptables looks for a rule in its list to match it to. If it doesn’t find one, it resorts to the default action. iptables almost always comes pre-installed on any Linux distribution.

Iptables has four default tables: Filter, NAT, RAW, and Mangle.

The default table is **FILTER**. It uses three different chains: **input, forward, and output**.

* **Input:** This chain is used to control the behavior for incoming connections. For example, if a user attempts to SSH into your PC/server, iptables will attempt to match the IP address and port to a rule in the input chain.
* **Forward:** This chain is used for incoming connections that aren’t actually being delivered locally. Think of a router – data is always being sent to it but rarely actually destined for the router itself; the data is just forwarded to its target. Unless you’re doing some kind of routing, NATing, or something else on your system that requires forwarding, you won’t even use this chain.
* **Output:** This chain is used for outgoing connections. For example, if you try to ping howtogeek.com, iptables will check its output chain to see what the rules are regarding ping and howtogeek.com before making a decision to allow or deny the connection attempt.

It is possible to check the default policy behavior of iptables, using this sintaxe: *iptables -L*. The default behavior is ACCEPT to input, output and forward. It means that, by defaulting, all package will be accept when enter in the interface. If you would rather deny all connections and manually specify which ones you want to allow to connect, you should change the default policy of your chains to drop. Doing this would probably only be useful for servers that contain sensitive information and only ever have the same IP addresses connect to them.

Following, the next table is **NAT**. This table is consulted when a packet tries to create a new connection. It has following chains:

* **PREROUTING:** This chain rule alters a packet as soon as it’s received
* **POSTROUTING:** This chain rule alters a packet as it is about to go out.
* **OUTPUT:** This chain rule alerts locally generated traffic.

Next, iptables has **RAW**. The Raw table is used to exempt packets from connection tracking. This table consists of two chains:

* **OUTPUT:** To alter locally generated packets
* **PREROUTING:** To alter incoming connections

Finnaly, you have MANGLE table. The Mangle table adjusts the IP header properties of packets. The table has all the following chains we described above:

* **INPUT:** for incoming packets
* **OUTPUT:** To alter locally generated packets
* **FORWARD:** for packets routed through the linux box
* **PREROUTING:** To alter incoming connections
* **POSTROUTING:** To alert outgoing connections

When a packet matches a rule and it is given a target, which can be another chain or one of these special options:

* **REJECT:** server receives the packet and rejects that packet and also send the acknowledgement.
* **DROP:** server receives the packet and drop the request without sending any acknowledgement.
* **ACCEPT:** server receives the packet and server allows that request.
* **RETURN:** this rule sends the packet back to the originating chain so you can match it against other rules.

The best way to show the difference between these three rules is to show what it looks like when a PC tries to ping a Linux machine with iptables configured for each one of these settings.



Figure 3 - Allowing the connectio.

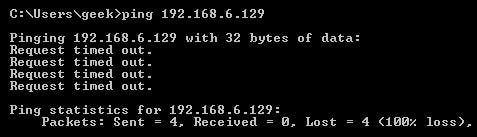


Figure 4 - Dropping the connection.

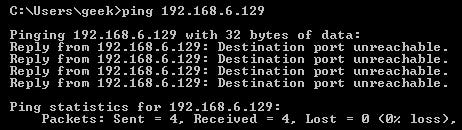


Figure 5 - Rejecting the connection.

The general process of how iptables uses the tables is presented in Figure 6.

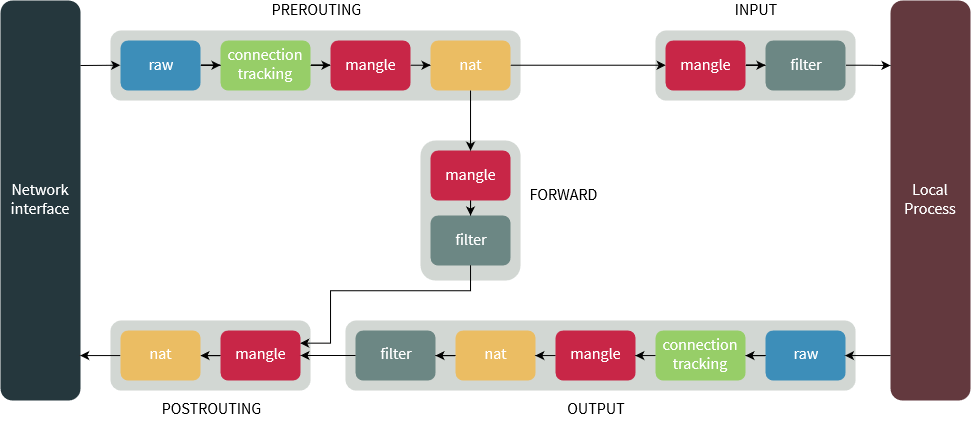


Figure 6 - iptables Process Workflow.

**Task 4: Read and realize the tutorial “An In-Depth Guide to iptables, the Linux Firewall”.**

1. **Support Material**:
   1. <https://www.booleanworld.com/depth-guide-iptables-linux-firewall/>
   2. <https://ipset.netfilter.org/iptables.man.html>
2. **Time**: 20 mins.

**PART III – EVALUATION**

In this section, the students perform the required tasks using the simulator/emulator environment (IMUNISIS) and the iptables.

Part III tasks are related to the following scenario: ‘imune/imunes-examples/DNS+Mail+WEB.’

The answers to the tasks require:

1. Write the commands applied to the firewall (remember the sequence of commands is very important).
2. Demonstrate that the rules solve the proposed problem (print the screen that demonstrates your approach's efficiency). Show the positive cases (where the connection is allowed) and negative cases (where the connections are forbidden).
3. Design a set of firewall rules in the route R9, that block any ICMP traffic (like ping) to the internal hosts inside the zone ZZT and ZPM. Pay attention, does not block any package from these zones, just only the input traffic. Save the firewall configuration persistently in the router[[1]](#footnote-1). (30 points)
4. In the server dnsHost, enable and configure the ssh server. Also, create a firewall rule in the route R7 that block any ssh request and channels to this server, except the packages from the mm host (zone ZZT)1. (30 points)
5. Change the configuration in the route R3 to it has this behavior (40 POINTS):
   * + - 1. Change the default policy to assure that all input traffic is denied by default
         2. Only DNS package is allowed to pass through the firewall to achieve these hosts: aRootServer, bRootServer, hr2, and dnsCom. Remember the rules requires to be bi-directional (to/from). (20 POINTS)
         3. Create a new service ftp in the hr2 server. Adjust the rules in the route R3 that allows that only the servers aRootServer, bRootServer, and dnsCom can send ftp messages to the server.

REMEMBER SAVE THE EXPERIMENT FILES TO BE POSSIBLE TO USE IN FUTURE ASSIGMENT.

1. Verify that the other packages works fine, like before you apply the rule. [↑](#footnote-ref-1)