|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | | | | |
| Icon  Description automatically generated | Icon  Description automatically generated | | | Icon  Description automatically generated | Icon  Description automatically generated |
| Answer Sheet | | | | | |
| Assignment – A01 | | | | | |
| Introduction to Google Cloud Platform, Packet Tracer, Wireshark, and CLI Networking Tools | | | | | |
| Name | | : | Alvaro Austin | | |
| Student ID | | : | 2106752180 | | |

# A01a – Introduction To Google Cloud Platform

## [10 points] SSH Connection Establishment Proof

|  |
| --- |
| **VM1 uname -a Execution** |
|  |
| **VM2 uname -a Execution** |
|  |

## [10 points] Application Installation

|  |
| --- |
| **VM1 which lynx Execution** |
|  |
| **VM2 which lynx Execution** |
|  |

## [30 points] Firewall Rules Creation Proof

|  |
| --- |
| **Firewall Rules Page** |
|  |

## [50 points] Virtual Machine Creation Proof

No answer is required for this component in this document. Please see the assignment document for explanation regarding the submission artifacts for this component.

**Submitted the proof through SCELE.**

# A01b – Introduction to Packet Tracer

## Create Topology

|  |
| --- |
| **Screenshot of the Topology** |
|  |

x

## Configuration

|  |  |
| --- | --- |
| **PC0-wireless** | **PC1-wireless** |
|  |  |
| **PC0-wired** | **PC1-wired** |
|  |  |

## Connectivity Test

|  |  |
| --- | --- |
| **Command Prompt – PING** | **Simple PDU** |
| **From PC0-wireless to PC1-wired** | **From PC0-wireless to PC1-wired** |
|  |  |
| **From PC0-wireless to PC1-wireless** | **From PC0-wireless to PC1-wireless** |
|  |  |
| **From PC0-wired to PC0-wireless** | **From PC0-wired to PC0-wireless** |
|  |  |
| **From PC0-wired to PC1-wired** | **From PC0-wired to PC1-wired** |
|  |  |

## Connectivity Test with Simulation

### Simulation 1

|  |
| --- |
| **PING from PC0 to PC1**    **PDU from PC0 to PC1**    Simulation 1 has the correct configuration. As you can see from PDU and PING test, the first time we tried to do connectivity test it failed but the second time we tried to do connectivity test it came out successful. From what I can infer, this is because network needs time to send data. Hence, the network came out fail but successful (need to time forward for this to work).  This is why simulation 1 has the correct configuration because the connectivity test ended up successful. |

### Simulation 2

|  |
| --- |
| Simulation 2 ended up failing. This can be seen from PDU done in the example below. When we use simulation mode to find out about the problem, you can see that destination IP address of 192.166.0.2 is not in the same subnet and is not the broadcast address. This is the reason why the packet isn’t successfully delivered. |

# A01c – Introduction to Wireshark

## Simple Packet Capture

|  |
| --- |
| **Screenshotted website:** |

## Simple Packet Filtering

|  |
| --- |
| Before Applying Filter |
|  |
| After Applying Filter |
|  |

## Simple Packet Analysis

|  |
| --- |
| IPv4 Address Screenshot |
|  |
| Packet Detail Screenshots |
| Detail of packets: |
| Brief and Concise Explanation |
| I will explain what happened in the packet list step by step. Do note that <http://gaia.cs.umass.edu/> has an IP address of 128.119.245.12 and my connection from Wireless LAN (ui.ac.id) is 10.5.89.213. Now that everything has been mentioned, I will explain:   1. My current connection (source = 10.5.89.213, IPV4 from Wireless LAN) request data from <http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html> (with IP address of 128.119.245.12). The data that were requested is the meta data of the website. In this case it is text/html. 2. Then <http://gaia.cs.umass.edu/> return back a response of data that were requested to my connection (10.5.89.213) in the form of text/html (html data). 3. The next data that were requested (from my connection: 10.5.89.213) is favicon.ico which is the logo that is generated while opening the browser page (to <http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html> with IP of 128.119.245.12). 4. If we open <http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html> (with IP of 128.119.245.12), you could see that the website has no favicon.ico (seen as logo) hence it returned 404 error to us (10.5.89.213) means the requested data was not found. |

# A01d – Introduction to CLI Networking Tools

## IP Config

|  |
| --- |
| Step 1: Local Device |
|  |
| Step 2: GCP VM 1 |
|  |
| Step 2: GCP VM 2 |
|  |

## ARP

|  |
| --- |
| Step 1: Local Device |
| Command: **arp -av**            Primary Network Interface |
| Step 2: GCP VM 1 |
| Command: **ip neigh show** |
| Step 2: GCP VM 2 |
| Command: **ip neigh show** |

## DiG

|  |
| --- |
| Step 1: GCP VM 1 |
|  |
| Step 1: GCP VM 2 |
|  |
| Step 2: Explanation |
| Side by side:    The output between dig on both VM is (VM 1 to google.com and VM2 to gaia.cs.umass.edu:   * Answer Section (IP Address): 6 (VM 1) to 1 (VM 2) * Query time (in ms): 4 (VM1) to 83 (VM2) -> VM1 faster than VM2 * Message Size: 135 (VM1) to 62 (VM2)   The massive difference between 2 output is on the query time and the number of IP address assigned to google.com DNS. The reasoning between this difference based on what I could analyze is:   1. Google servers is all over the world. Meanwhile, gaia.cs.umass.edu is a server hosted in University of Massachusetts which located in eastern coast of US. In my VM1, the server is hosted on us west, the geographical difference could be the main difference of query time. 2. The number of IP address give some good reasoning behind the speed. When we try to access a website, we tried to translate domain name’s to IP address. Based on what I read on Internet, Google operates on it’s own DNS server that is highly optimized. That’s why there is 6 DNS record that **dig** found. This, most likely, increase the speed of DNS lookup. Meanwhile, gaia.cs.umass.edu might not operate on DNS server with the same speed or reliability as google.   Based on what I read, higher message size could means affect the speed and efficiency to be worse than smaller message size. Even though **dig google.com** gives higher message size, the speed that it delivers still much faster than **dig gaia.cs.umass.edu**. This means the factor above play a big role, such as number of IP address (affected the DNS lookup) and geographical location. |

## TCPDump

|  |
| --- |
| Step 3: TCPDump Terminal |
|  |
| Step 3: Lynx Terminal |
|  |
| Step 4: File Moved Verification |
|  |
| Step 6: Open Captured Data using TCPDump |
|  |
| Step 6: Open Captured Data using Wireshark |
|  |
| Step 7: Open Captured Data using TCPDump -n |
|  |
| Step 8: Comparison Explanation |
| On step 6, without using -n parameter, it returns uses the name resolution for my machine and DNS for aren.cs.ui.ac.id. From what I read and analyze, without using -n basically converting IP addresses between my machine and target (aren.cs.ui.ac.id) to their domain name. This is great for human readability.  On step 7, as I mentioned previously, using -n gives the real output, the one without conversion of IP addresses to domain name. This increases output speed because we don’t have to convert each time. |

## Netstat and ss

|  |
| --- |
| Step 1: Local Device |
| Command: **netstat -ano** |
| Step 2: GCP VM 1 |
| Command: **ss -aetu** |
| Step 2: GCP VM 2 |
| Command: **ss -aetu** |
| Step 3: Explanation |
| Using **netstat -ano** and **ss -aetu** gives all active TCP and UDP connection. Both also gives local and Peer addresses. Not only that they also gives Port number (PID) that’s really important if we want to use a port that is already been used by another process. Using those PID, we could kill the process corresponding to that port. |

## Ping and Tracert

|  |
| --- |
| Step 1: GCP VM 1 |
| **Ping VM1 to VM2(Internal IP = 10.138.0.2)**    **Traceroute VM 1 to VM 2 (Internal IP = 10.138.0.2)** |
| Step 1: GCP VM 2 |
| **Ping VM2 to VM1(Internal IP = 10.138.0.3)**    **Traceroute VM 2 to VM1 (Internal IP = 10.138.0.3)** |
| Step 2: Explanation |
| Based on what I could infer on the result, PING is used to test the reachability of a network host. It also counts the time it takes to react the network host. PING basically sends bytes of data then wait for response from the network host. If the data takes too long to be reached, then it will return time out. Meanwhile Traceroute is to trace path of the packets to reach the destination, in this case network hosts. |