CSCI 7000-013

SDN Virtualization and Orchestration

Lab 8

TensorFlow

University of Colorado Boulder

Department of Computer Science

Professor Levi Perigo, Ph.D.

# Summary

TensorFlow is a software framework for building and deploying machine learning models. Machine learning is a different approach than traditional programming. With traditional programming, you write the program that tells the computer exactly what to do to complete the task. With machine learning, you don't explicitly tell the computer how to do something. Instead you show a training data and the machine learning algorithm uses the training data that come up with its own rules to complete the task. TensorFlow gives you the basic building blocks that you need to design, train, and deploy machine learning models.

While it's flexible enough to be used for several different types of machine learning types of algorithms, it's typically used to build deep neural networks. Deep neural networks built with TensorFlow are used in many different areas like image recognition, where you recognize what objects appear in a picture. Speech recognition where you turn speech into text.  But everything you learn about building and deploying with TensorFlow works the same way no matter what kind of application you are using it for.

TensorFlow is a low level toolkit. It can take quite a few lines of code to build a machine learning model in TensorFlow. Because of this, there are wrappers for TensorFlow that simplify common operations. The most popular wrapper for TensorFlow is Keras. Keras is a high level programming toolkit that makes it easy to build many different types of neural networks with only a few lines of code.

Keras still uses TensorFlow behind the scenes to do the processing. It's a great choice if you don't need the low level flexibility of TensorFlow. But once you're familiar with TensorFlow, it's great to also learn how to use Keras. It makes many common tasks much easier and it can be a great time saver.

# Objective 1 – Building a Simple Model in TensorFlow

In this objective, you will build a TensorFlow computational graph that adds two numbers together. Figure 1 shows what the computational graph will look like. The model only has three nodes X, Y, and addition. The graph has two inputs, X and Y. Those are the two numbers to be added together. The graph has one operation called 'addition'. That node simply adds together the tensors passed into it. Once this graph is defined, you can use it by creating a new TensorFlow session. Then you'll feed in values for X and Y, and you'll call the session object to execute the addition node.

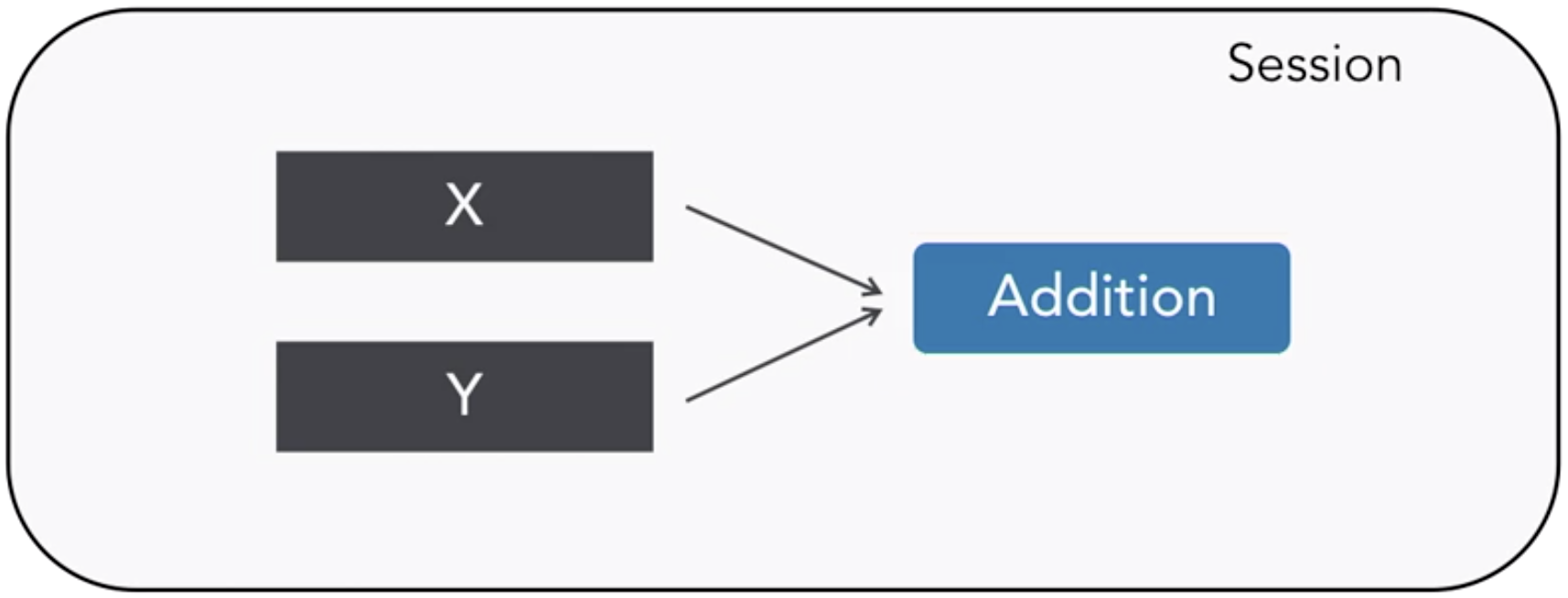


Fig. 1: TensorFlow addition operation graph.

The result you get back from executing the addition node will be the answer. Following are the high-level steps needed to build a TensorFlow application:

1. Define the model
2. Create a session
3. Pass in data
4. Send it off to the TensorFlow execution engine
5. Result

Define the X and Y input nodes. When you create a node in TensorFlow, you must choose what kind of node to create. The X and Y nodes will be placeholder nodes that get assigned a new value every time you make a calculation, so use “tf.placeholder” function to define the node.

When you create a placeholder node, you must pass on the data type. Since you'll be adding numbers here, use a floating point data-type. You also need to give this node the name. The name will show up when you look at graphical visualizations of our model. You can name this node X by passing in the parameter called 'name' with a value of X. Similarly, define Y.

Now you can define the node that does the addition operation. In TensorFlow, you can do that by creating a “tf.add” node. Then you'll pass on the X and Y nodes to the addition node, that tells TensorFlow to link those nodes on the computational graph.

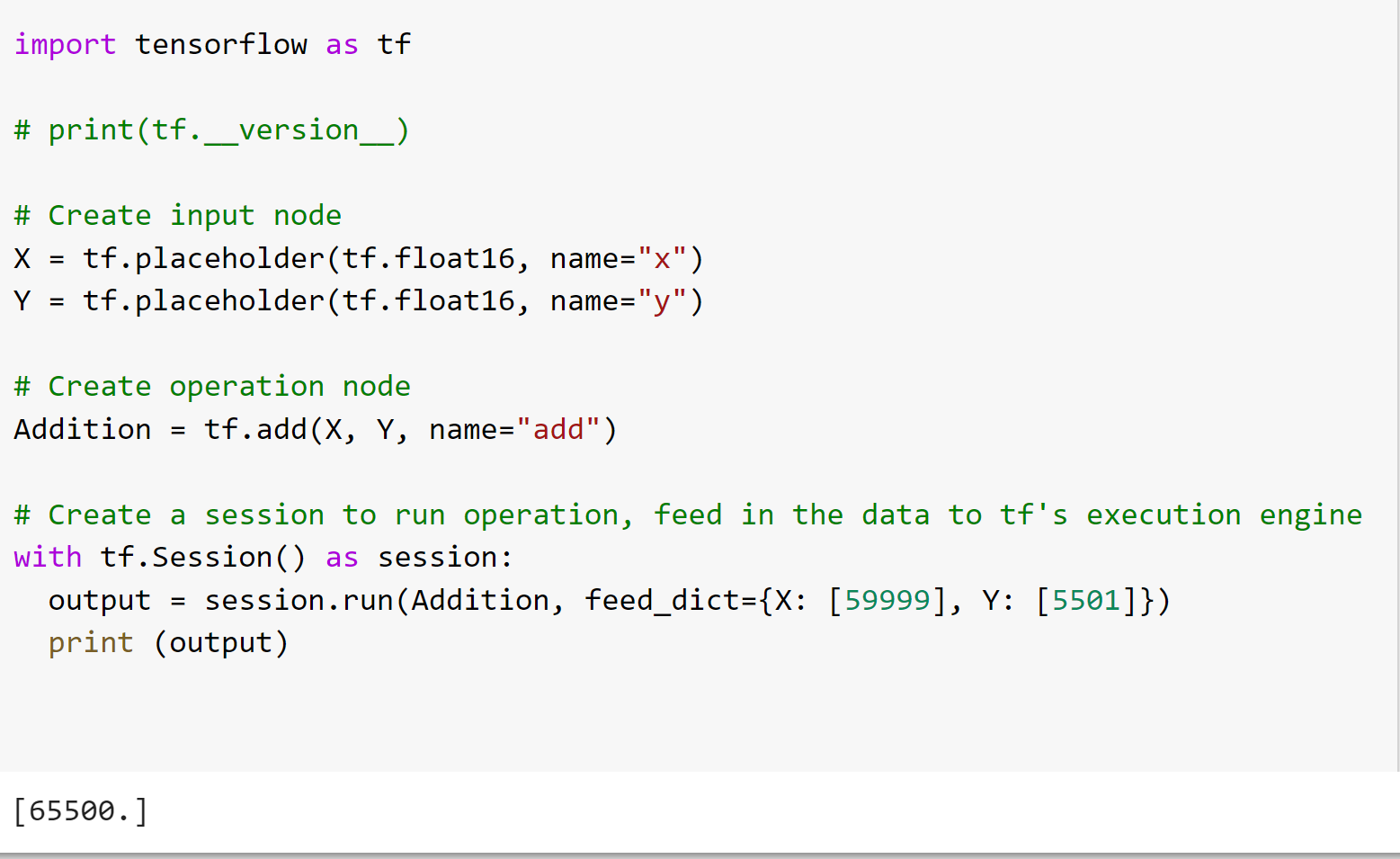
You may name your Addition node. And that's the entire definition for our simple computational graph.

To execute operations in the graph, you have to create a session by calling “tf.session”. Now that you’ve have a session, you can ask the session to run operations on their computational graph by calling “session.run”. When calling “session.run” you need to pass on the operation you want to run, in this case addition. When the addition operation runs, it's going to see that it needs to grab the values of the X and Y nodes, so you also need to feed in values for X and Y. To do so you can supply a parameter called 'feed dict'.

Notice that you when you pass in the arrays for the values of X and Y, the result you get back from TensorFlow is also an array. That's because TensorFlow always works with tensors, which are multi-dimensional arrays. It's expecting you to feed in the arrays and it will return arrays, that means you can feed in multiple numbers at once for X and Y. Now feed in some more numbers for X and Y and run the code again.

You could even pass in multi-dimensional rays for X and Y and it would still work the same way because you aren't just adding together two numbers, you're actually adding together two tensors. TensorFlow's real value is when you are working with large data sets and computationally intensive operations.

The below line shows the result of executing that code-



Deliverable:

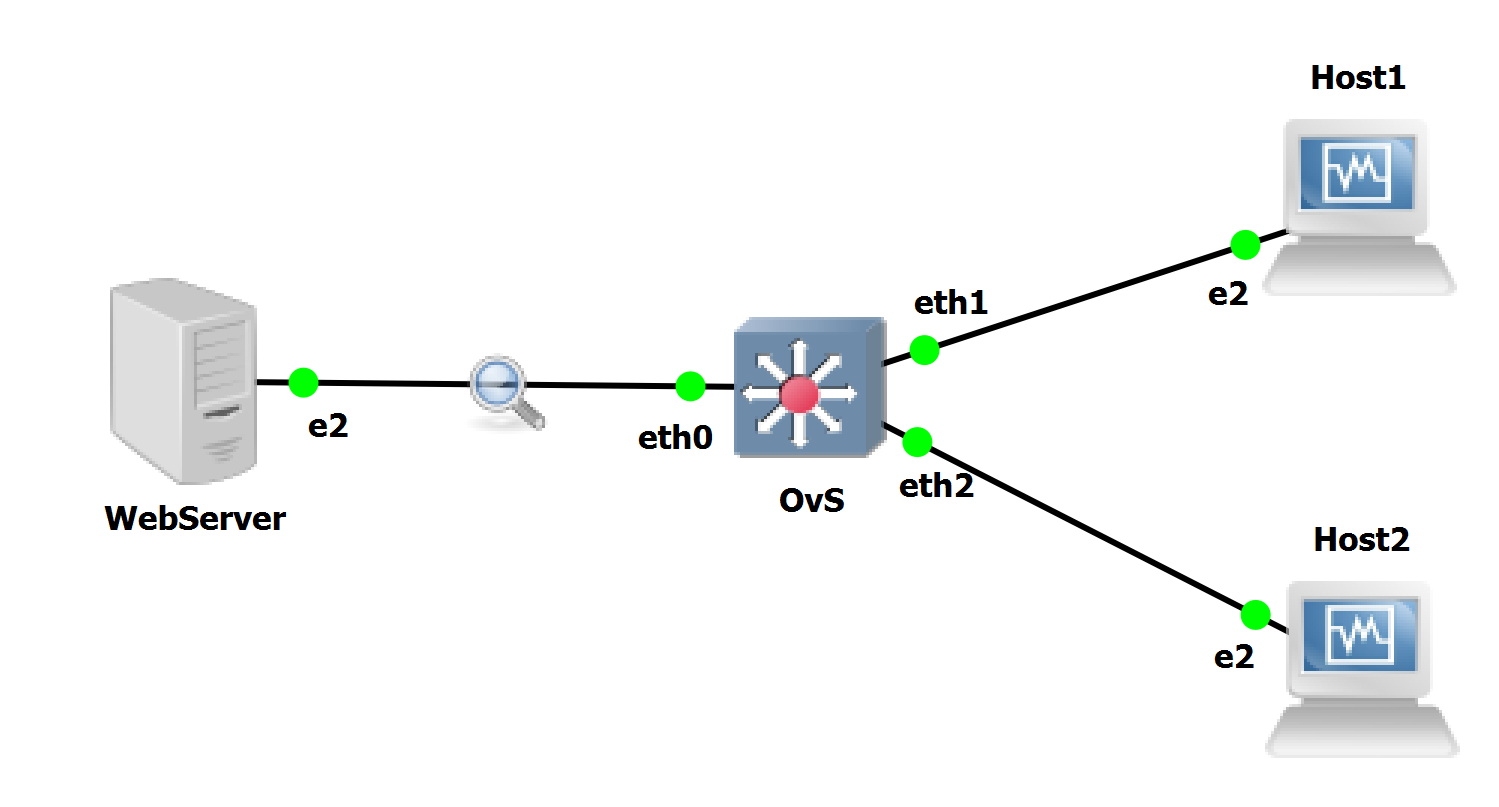
1. Using the fundamentals of the TensorFlow model that you created above, write a simple machine learning networking application of your choice.
2. Attach your python files along with the lab and explain the application you developed in detail.

ML identifies DDoS Traffic from normal traffic-

Below is the network topology used to detect DDoS traffic in the network using ML script. The network has two hosts which can access web page hosted by the web server. These hosts and server are connected using an OvS.

The ML script is hosted on the webserver which monitors the traffic coming in using Wireshark. The script is trained with various legitimate and attack traffics and can detect a potential DDoS attack.

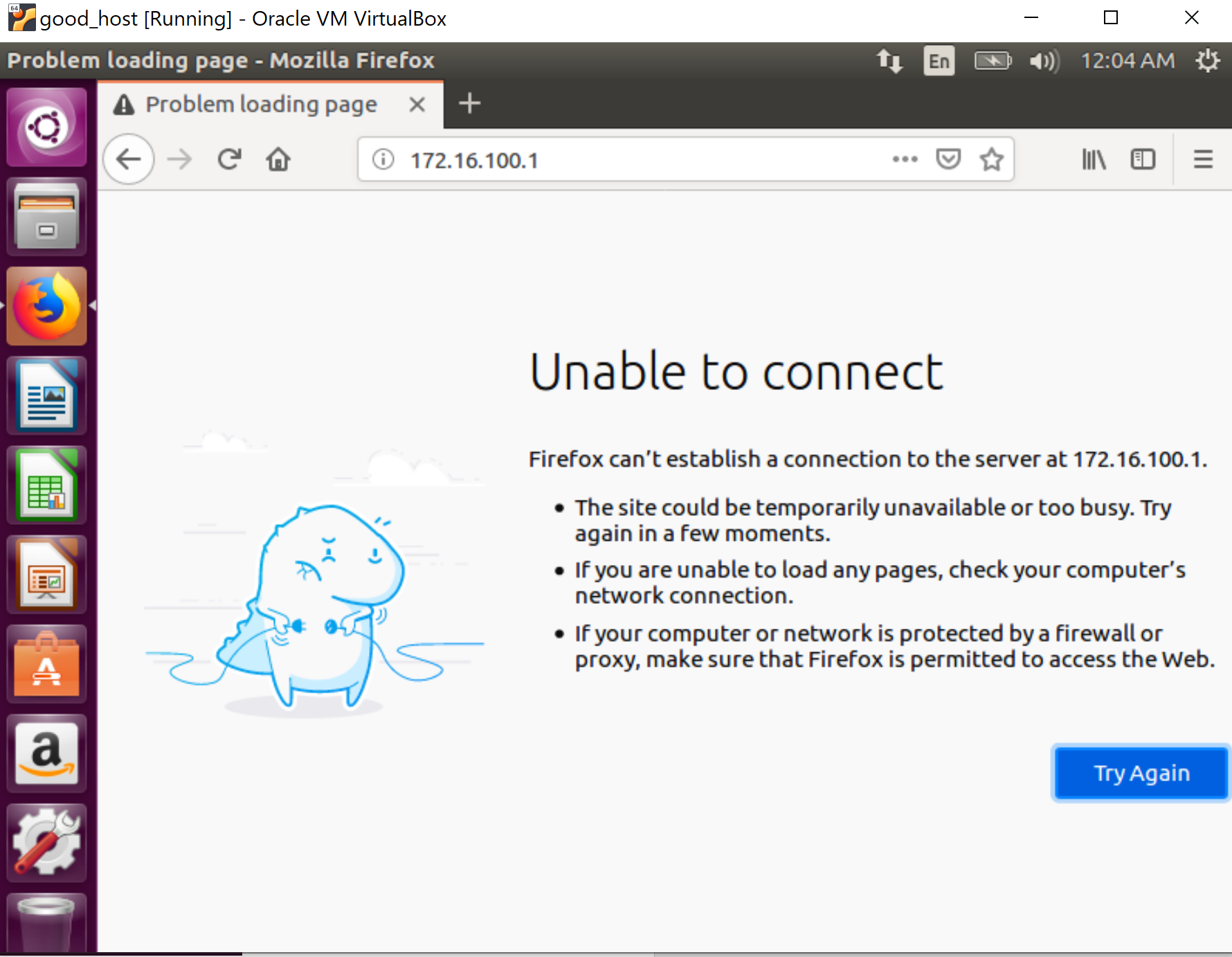
For testing purpose, host1 will carry DDoS Syn Flood attack on the webserver, making the website inaccessible to any other device in the network. Host2 will have legitimate traffic which is accessing the webpage.



The syn flood initiated by host1. The attack is directed towards the webserver IP 172.16.100.1:80 using 100 threads at the same time.



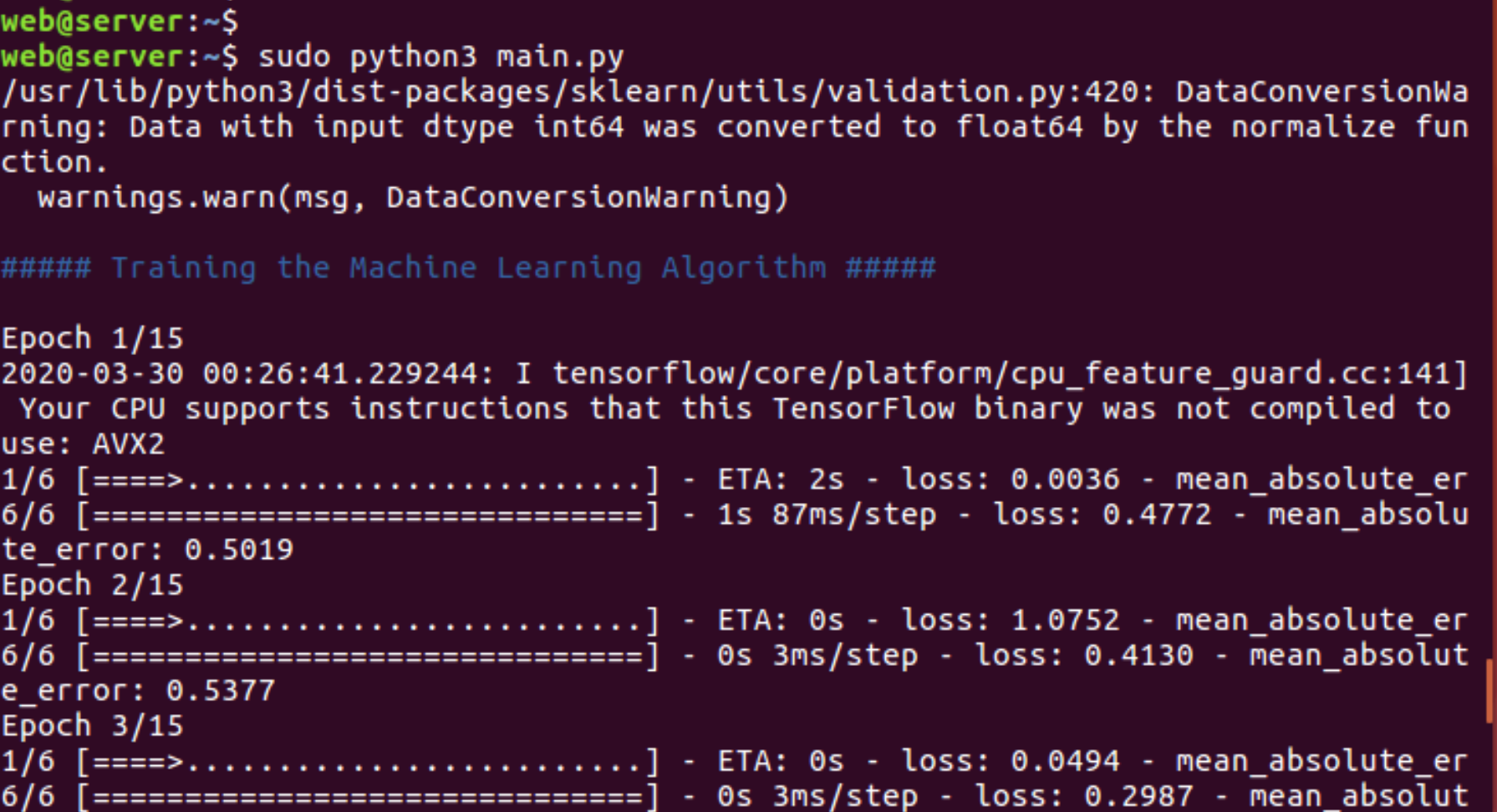
When the host2 tries to reach the website during the malicious host1 attack, the website does denial of service.



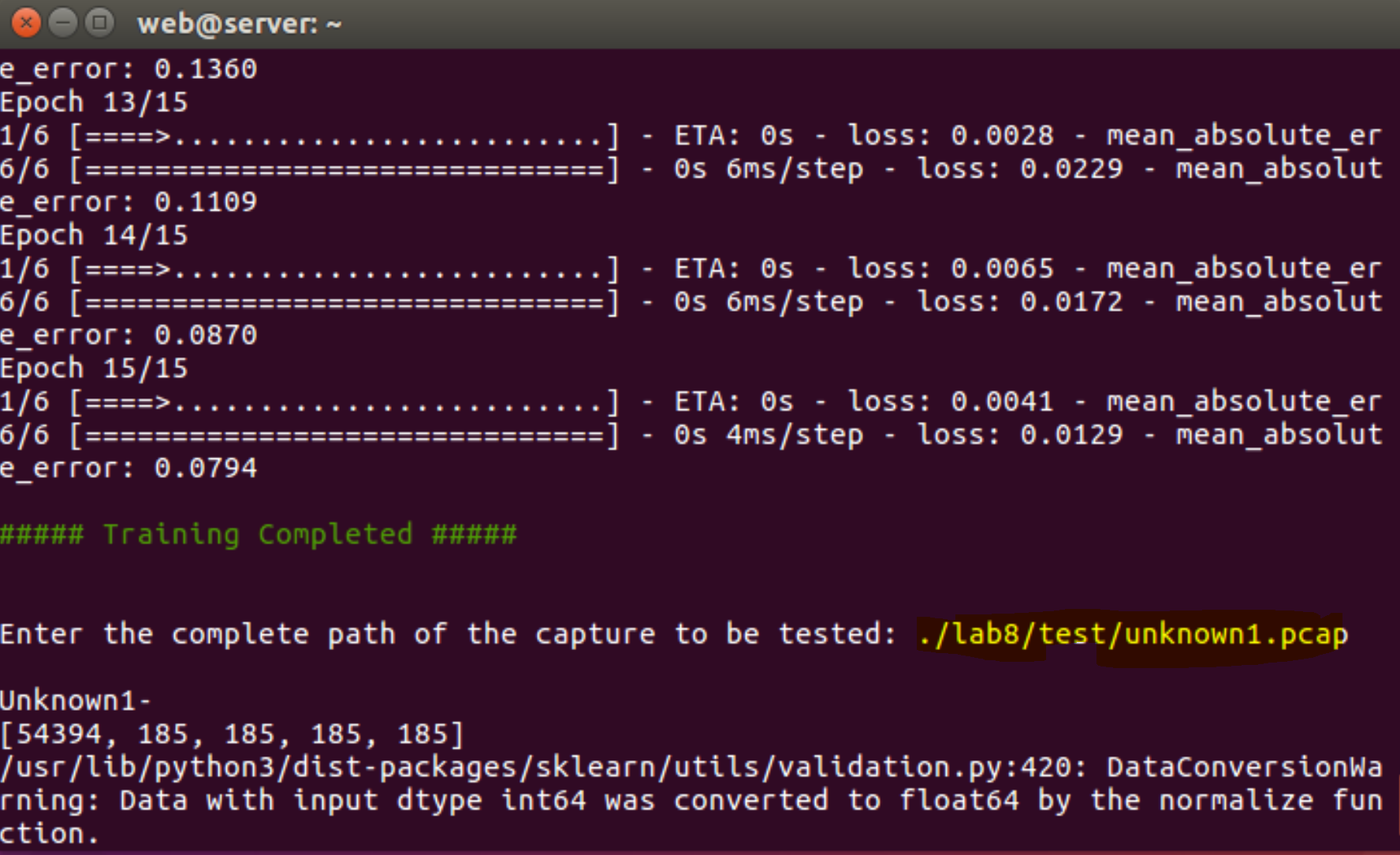
Webserver is running the script where we will train and test the DDoS traffic capture.

1] Create a dataset for training the ML algorithm

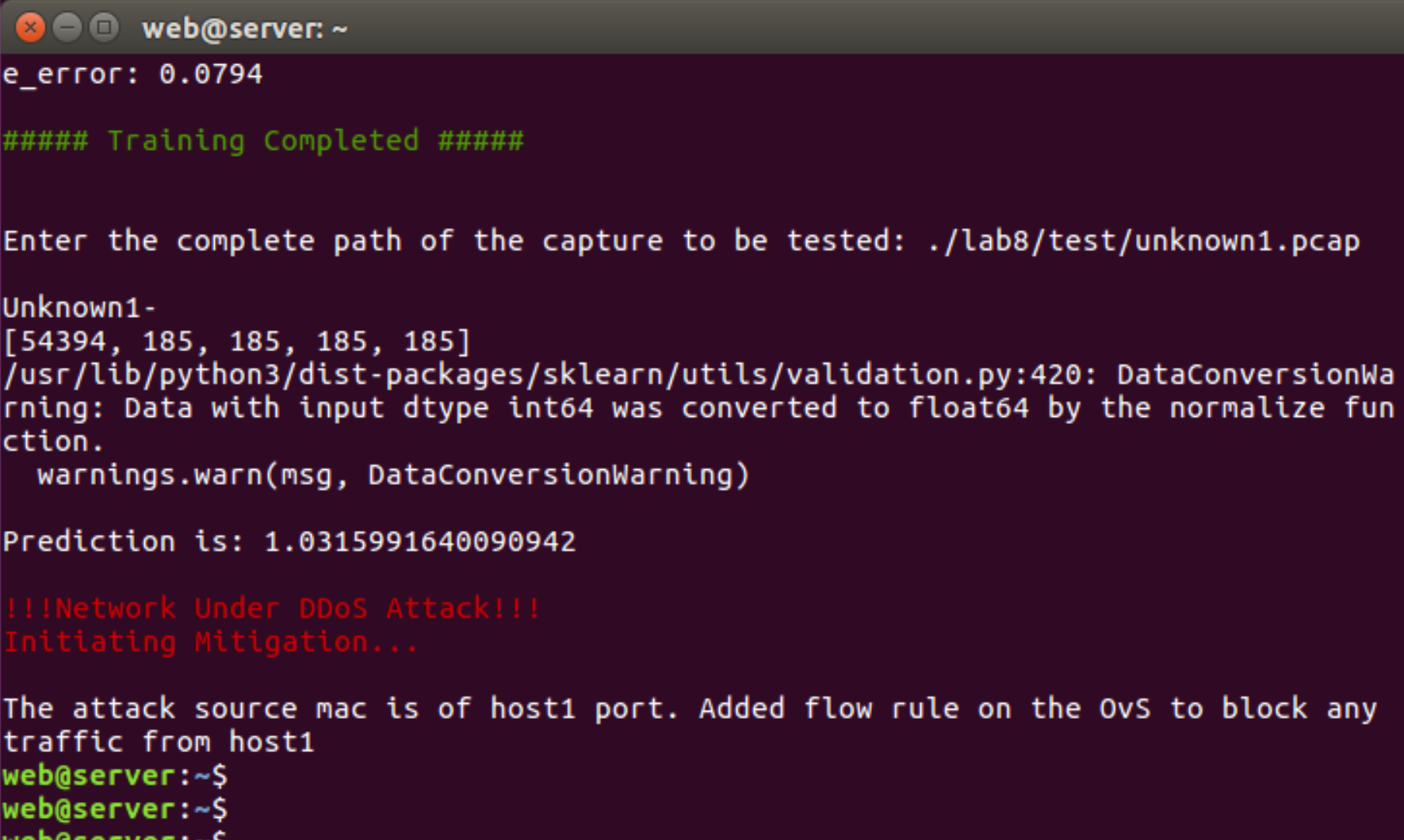
2] Start training the algorithm with 3 legitimate traffic sets and 3 attack traffic sets



3] Once the training is completed, we enter the test .pcap capture file to find out if the network is under DDoS attack.

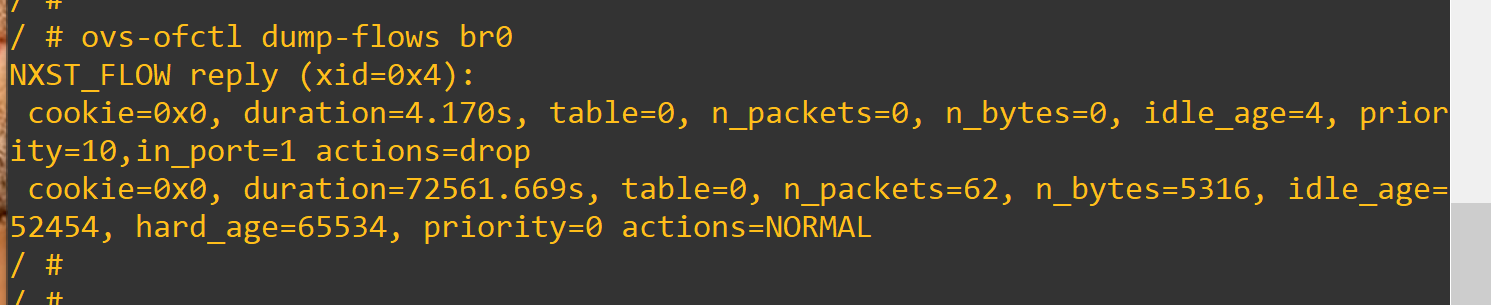


4] With the training the script predicted that the network was for sure under stack with prediction of 1.

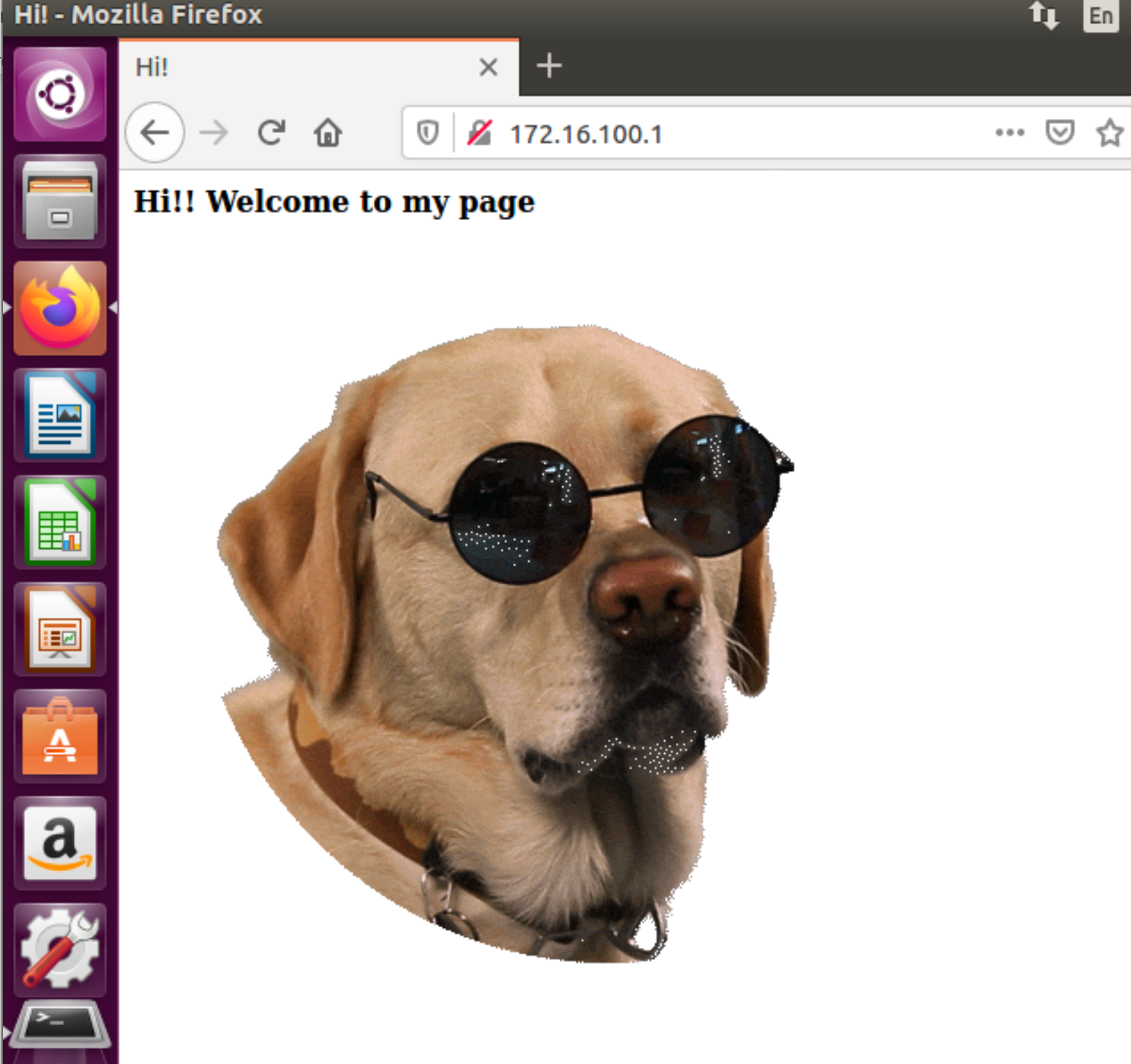


5] Script then initiates DDoS mitigation by finding out which host is sending legitimate web page access traffic and which host is being compromised.

6] The script identifies that the attack is initiated by host1 and blocks any traffic coming from that host’s port. Below is the screenshot of the OvS flow entry to block host1 traffic.



7] Host2 can access the webpage again after the flow entry to mitigate DDoS attack is added.



All the training pcap dataset, unknown attack traffic captures, codes and webpage are submitted along with the lab.