|  |
| --- |
|  |
| Capture and Replay Tool |
| Documentation |

|  |
| --- |
|  |

Table of Contents

[CHAPTER 1 2](#_Toc458119794)

[INTRODUCTION 2](#_Toc458119795)

[1.1 Objective 2](#_Toc458119796)

[CHAPTER 2 3](#_Toc458119797)

[REQUIREMENTS 3](#_Toc458119798)

[CHAPTER 3 4](#_Toc458119799)

[DESIGN 4](#_Toc458119800)

[CHAPTER 4 7](#_Toc458119801)

[CONSTRAINTS AND CHALLENGES 7](#_Toc458119802)

[CHAPTER 5 8](#_Toc458119803)

[CODING 8](#_Toc458119804)

[CHAPTER 6 9](#_Toc458119805)

[INSTALLATION PROCEDURE 9](#_Toc458119806)

[CHAPTER 7 10](#_Toc458119807)

[RESULTS 10](#_Toc458119808)

[APPENDIX-A 14](#_Toc458119809)

# CHAPTER 1

## INTRODUCTION

A packet generator or packet builder is a type of software that generates random packets or allows the user to construct detailed custom packets. Depending on the network medium and operating system, packet generators utilize raw sockets, NDIS function calls, or direct access to the network adapter kernel-mode driver.

### 1.1 Objective

Given a transaction from the packet capture, this tool should be able to regenerate the transaction sequence. There should be a way to modify any of the parameters in the package (Example IP, Port, Data) . The tool should automatically manage the packet sequence/ack numbers in-case of replaying the TCP flow through a proxy device.

# CHAPTER 2

## REQUIREMENTS

The following important requirements were identified for the capture and replay tool:

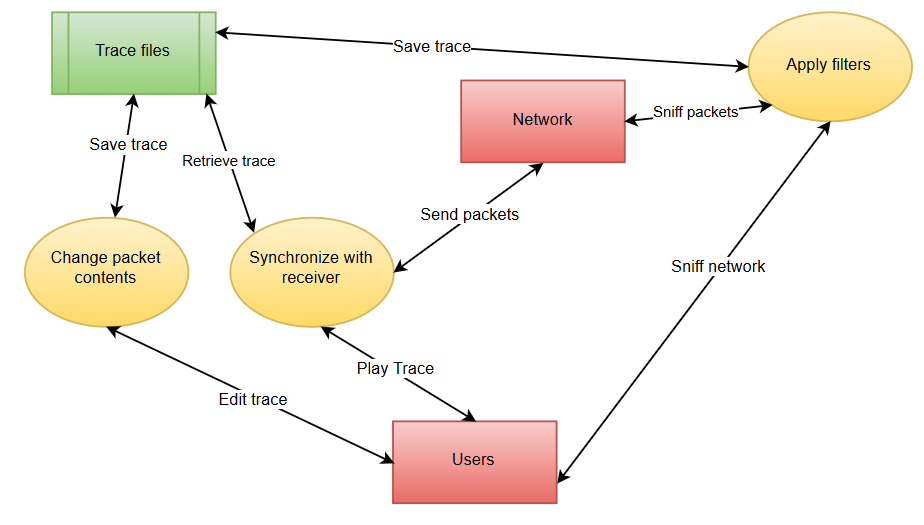
* The tool will have backend and GUI.
* The tool will be developed for TCP initially and later other protocols like HTTP and FTP will be integrated.
* GUI should have option to select trace file and select transaction in the trace file.
* Summary of the capture uploaded to be displayed in the GUI.
* Edit the transaction if required and replay.
* GUI should have options to change Source IP, Destination IP, port numbers and some fields in the TCP Header.
* The output will be in the form of trace file or data structure
* The tool will have 2 parts, the client and the server. Both will share the same capture file and replay the packets in synchronization with each other.
* The acknowledgement packets of TCP have to be handled correctly.

# CHAPTER 3

## DESIGN

The requirements mentioned in previously in chapter 2 are mapped to the required functionality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Steps** | **Description** | **Comments** |
| **1** | Application | The tool will contain a fully functional GUI with the ability to capture packets, replay a selected trace file and also to edit and save as a trace file. | Application will be designed with minimal overhead to avoid delays in packet transmission and capture. |
| **2** | Target platform | The given tool is intended for usage over Linux based machines. | The compatibility will be maintained for ubuntu 14.04 LTS |
| **3** | Languages used | The development of GUI and the entire functionality will be implemented using python. | Python 2.7 is expected to be used for development |
| **4** | Functionalities | 1. The tool will be able to upload a trace file and display the contents of each packet. Summary of each portion of the trace will be displayed by the tool.  2. User is provided with the ability to configure any detail of the packet individually or as a group based on certain filters. There will be an option to save this file.  3. User can replay this edited trace file by specifying the method of transmission. (e.g. Ethernet)  4. Trace file will be present on both the sender and receiver side which will be used for handling both sides of the transaction.  5. Tool will contain an option to sniff/capture network packets giving either a specific time interval/number of packets to be sniffed. This can be further customized by giving the specific details of packets to be sniffed(like source IP). 6. Sniffed packets can be stored as a separate trace file over the host system or represented as data structure for further usage. | Synchronization between the sender and receiver is managed based on packet. |
| **5** | Protocols | This application will be designed to operate on TCP/IP packets. This will be extended further based on the industrial requirement/trace file content. | Protocols like HTTP and FTP will be included as per requirement |
| **6** | Dynamic Adaptability | Synchronization between nodes will be automatically managed by this tool based on the type of transaction. | Tool will be present on sender and receiver sides. |



**Figure 1: Overview of the Capture and Replay Tool**

Scapy was chosen as the framework for packet manipulation. Some of the key advantages of using scapy for this project are:

* Since it is written in python, it helps development of tool with ease.
* Supports all the features required for the development of this tool.
* Supports all current operating systems.
* Has large user database offering technical solutions instantly.
* Beautifully syncs with Django which will be used for front end of the tool.

Based on the requirements the tool has been designed on the following architecture. A web based interface has been developed for user interaction. The stack runs on Django, a python framework for web apps. MySQL is used for storing packet data contents for editing.

The web app has the following features:

* A web based interface to upload any number of trace files.
* Interface to open a trace file and display its contents. The trace file contents are displayed in a tabular manner.
* The interface allows editing of fields of packets like IP, Port number etc.
* The interface allows editing an existing .pcap file (trace file) and saving a modified trace file.
* Functionality to replay the packets between two hosts.
* The interface displays the packets that are sent and received one by one in a tabular form.

# CHAPTER 4

## CONSTRAINTS AND CHALLENGES

Scapy, the python packet manipulation library used for implementing the core functionalities of the capture and replay tool has some limitations:

* Link layer not well managed yet.
* Does not give the right source IP for routes that use interface aliases (/proc/net/route reports only master interface)
* DNS packets not reassembled exactly as the original (no compression used)
* May miss packets under heavy load
* BPF filters do not work on PPP interfaces

# CHAPTER 5

## CODING

The project is divided into two phases:

* Phase 1:

Implementation of a script for client and server that replays the transaction sequence between client and server with synchronization. Given a trace in client and server machines, they should be able to replicate exactly the same. Trace file can be uploaded and started manually in the client and the server this phase.

* Phase 2:

Development of a web based GUI.

The following steps were adopted to achieve synchronization of replay of packets between client and server:

1. Load the trace file into the GUI. E.g. test.pcap
2. Split the loaded trace file (test.pcap) into multiple files based on sessions
3. Select the session that has to be replayed.
4. Load the selected session pcap file into Host 1 and Host 2
5. Configure IP1 of session in host 1 and IP2 of session in Host 2
6. Send packet 1 from host 1 to host 2 from the loaded pcap file
7. Host 2 wait (sniff) for packet 1 and replays packet 2 from loaded pcap file
8. Host 1 sniff packet from Host 2 and replay next packet.

# CHAPTER 6

## INSTALLATION PROCEDURE

The following softwares are required for the tool to run properly:

* Django – 1.9.1
* Python – 2.7
* Scapy
* MySQL-python
* MySQL-server
* MySQL-client
* Any web browser

The first four packages can be installed using the Python package installer (PIP).

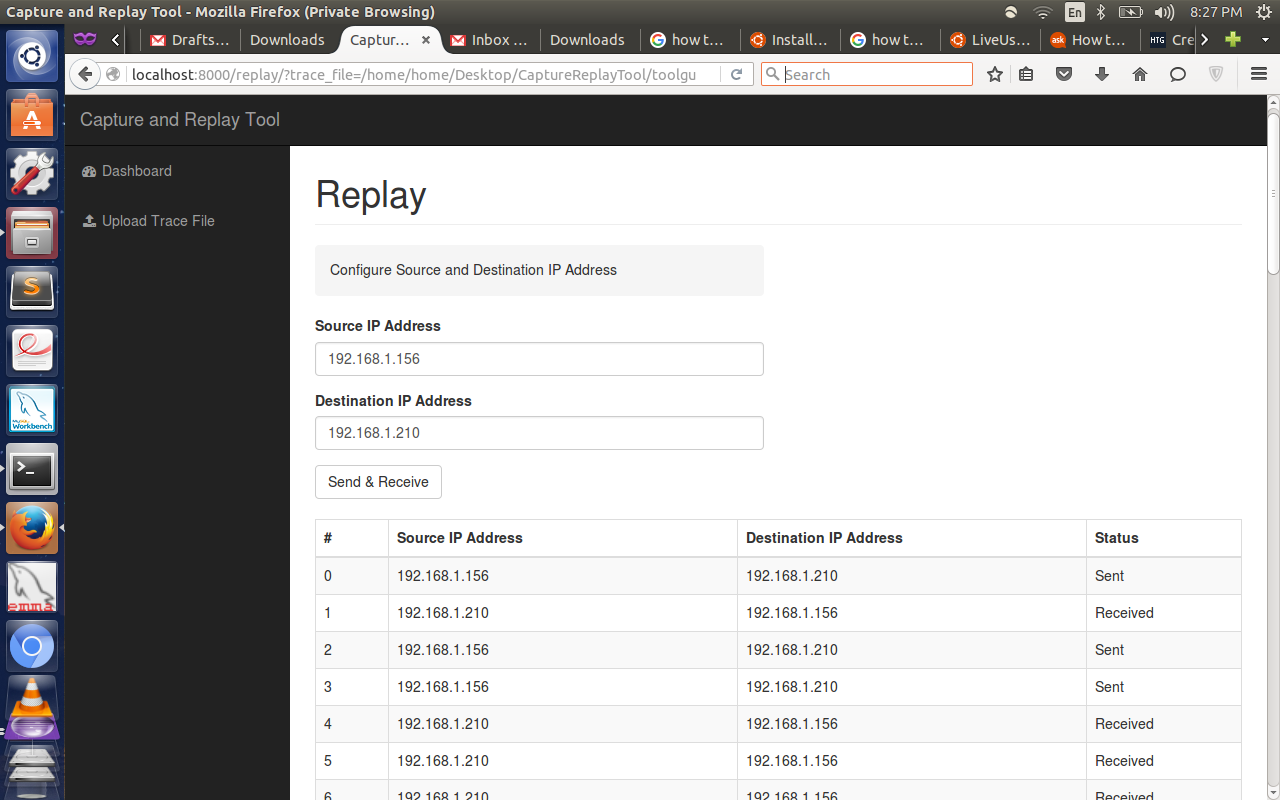
Steps to run the tool:

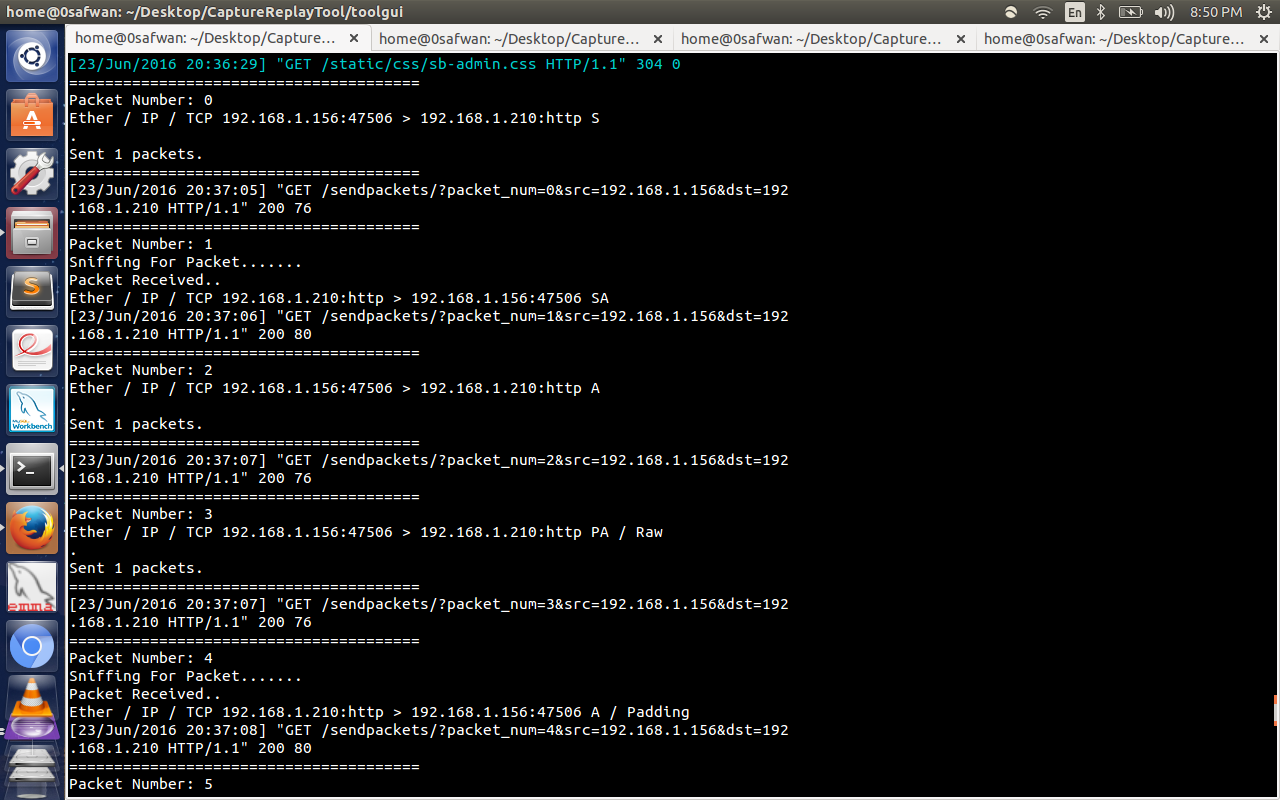
* Open the project directory.
* Find the main module (toolgui) of the capture and replay tool, and open the directory.
* Run the file manage.py using sudo privileges or as a superuser.
* Open a web browser and type in the url: localhost:8000/home.
* The dashboard of the tool opens.

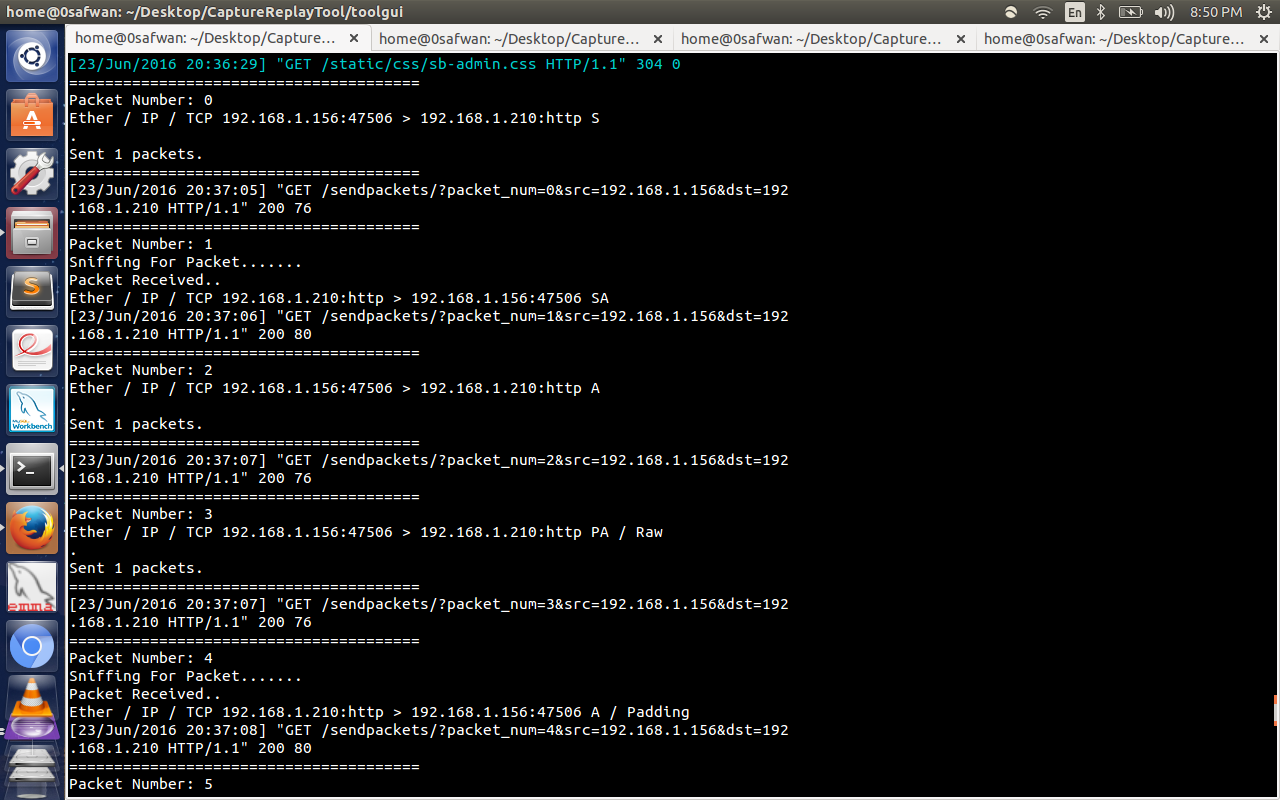
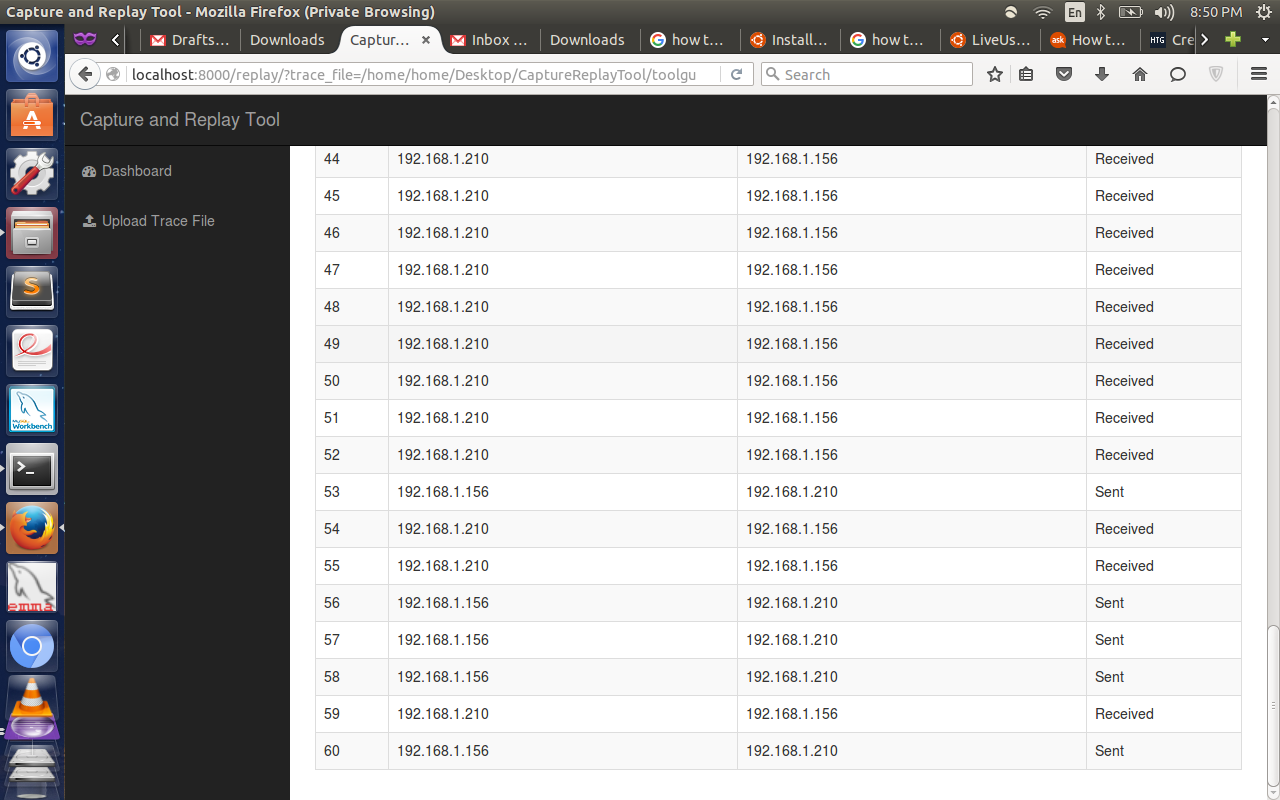
# CHAPTER 7

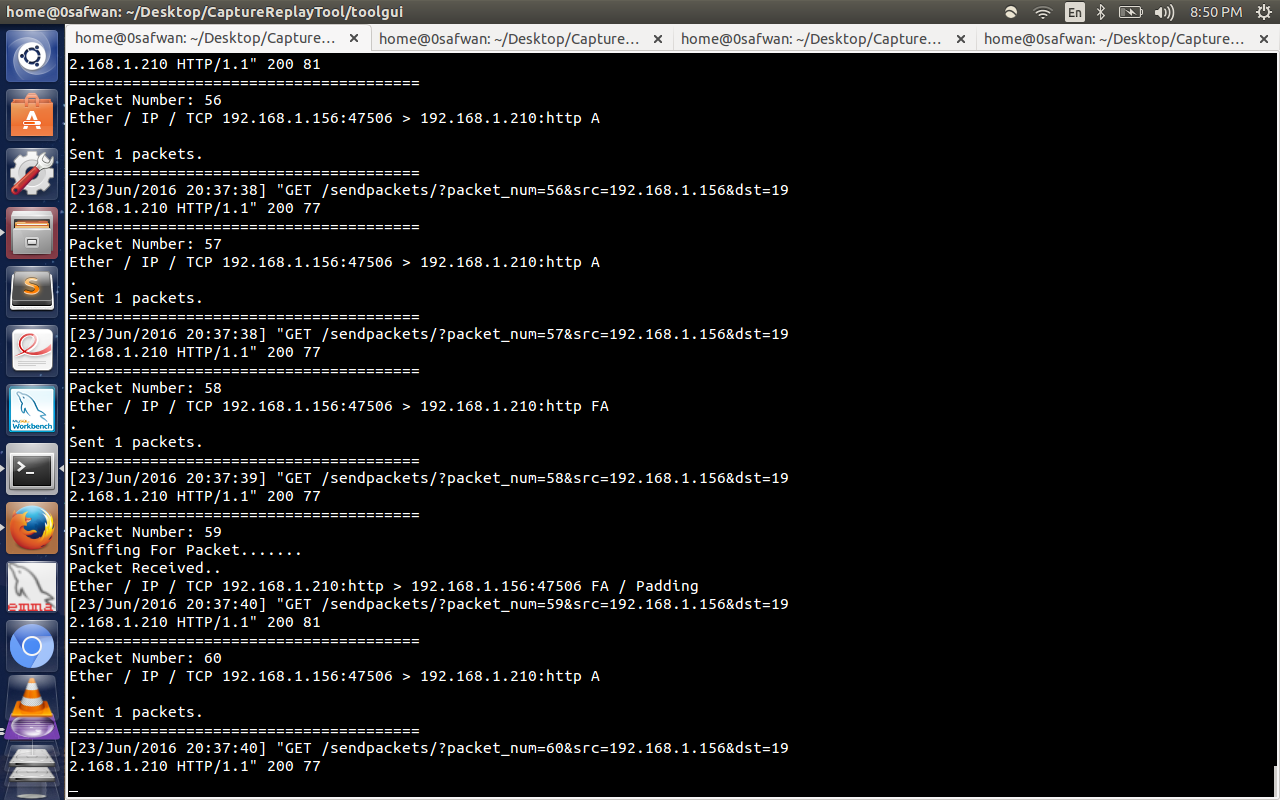
## RESULTS

The following screenshots show the regeneration of a transaction sequence between two hosts configured in the same network

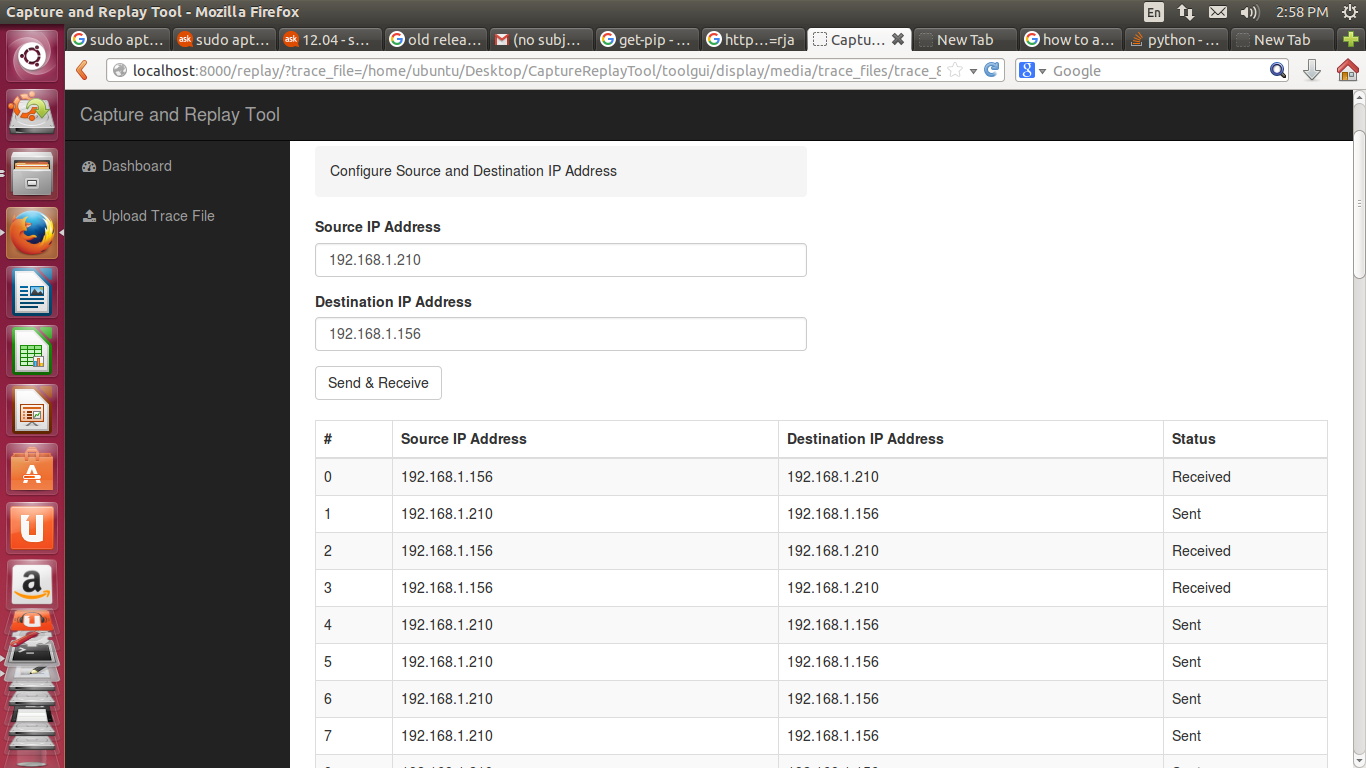
**On host 1:**

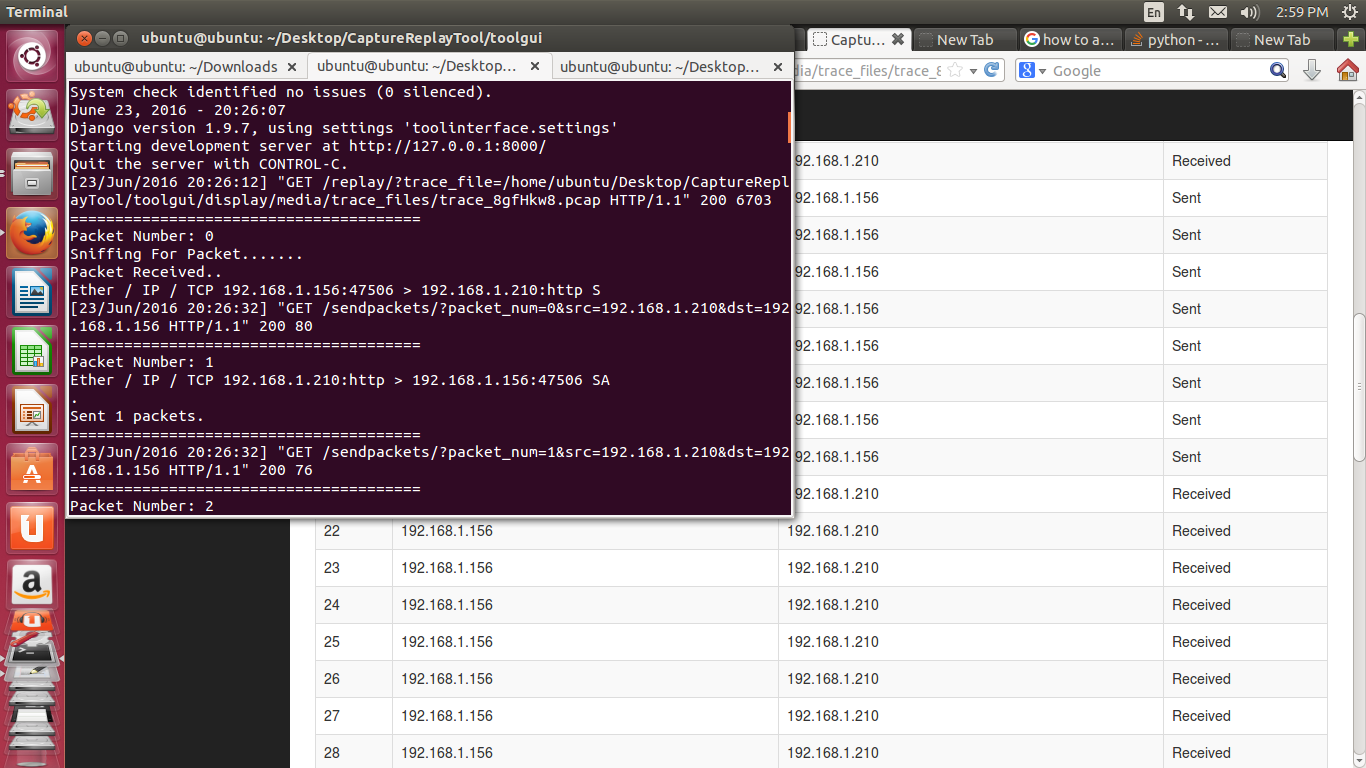


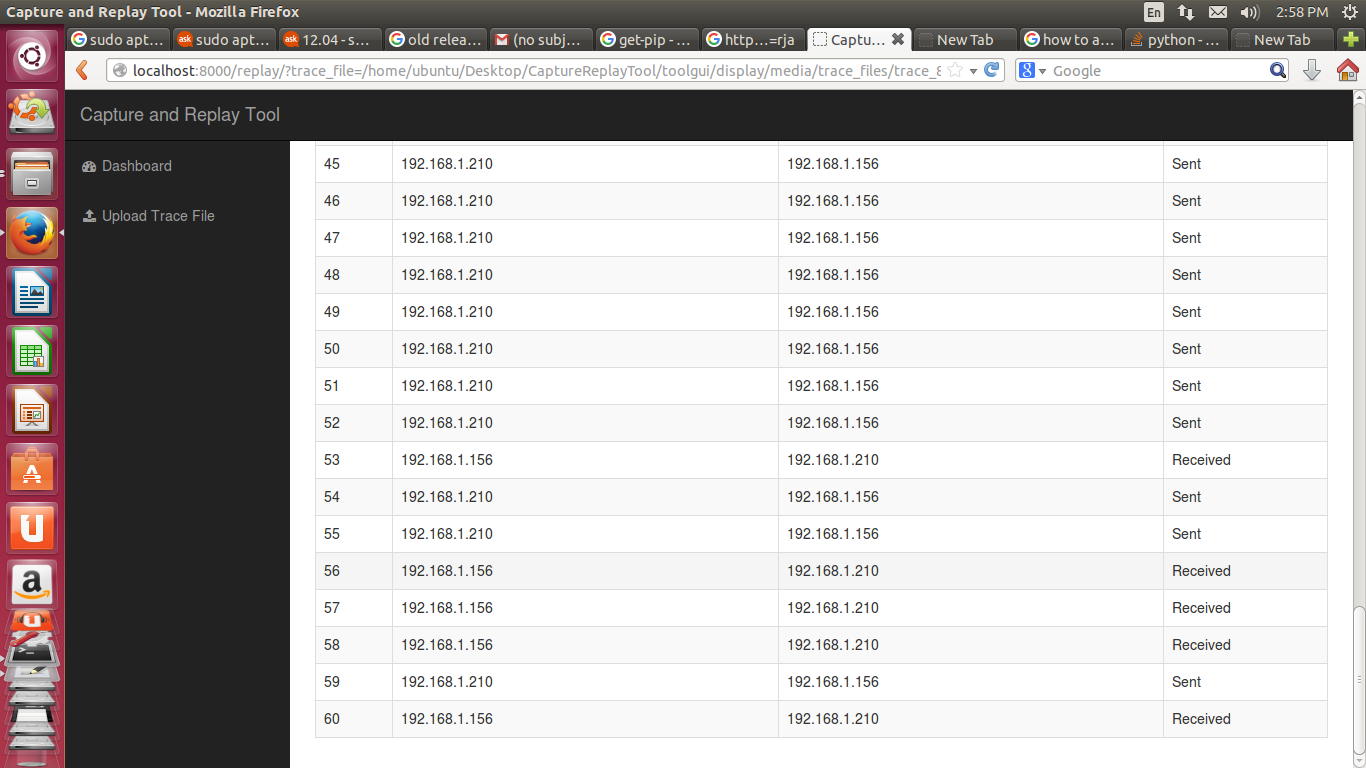


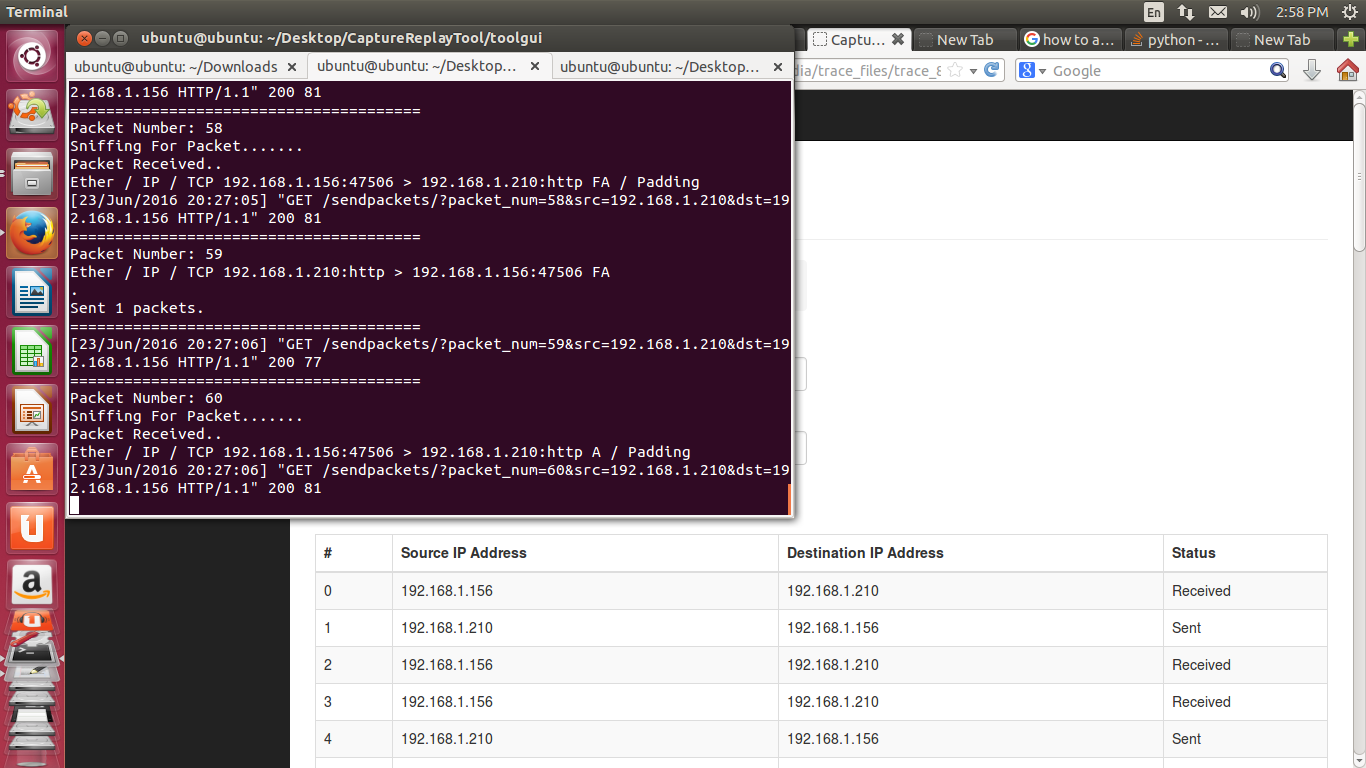


**On Host 2:**







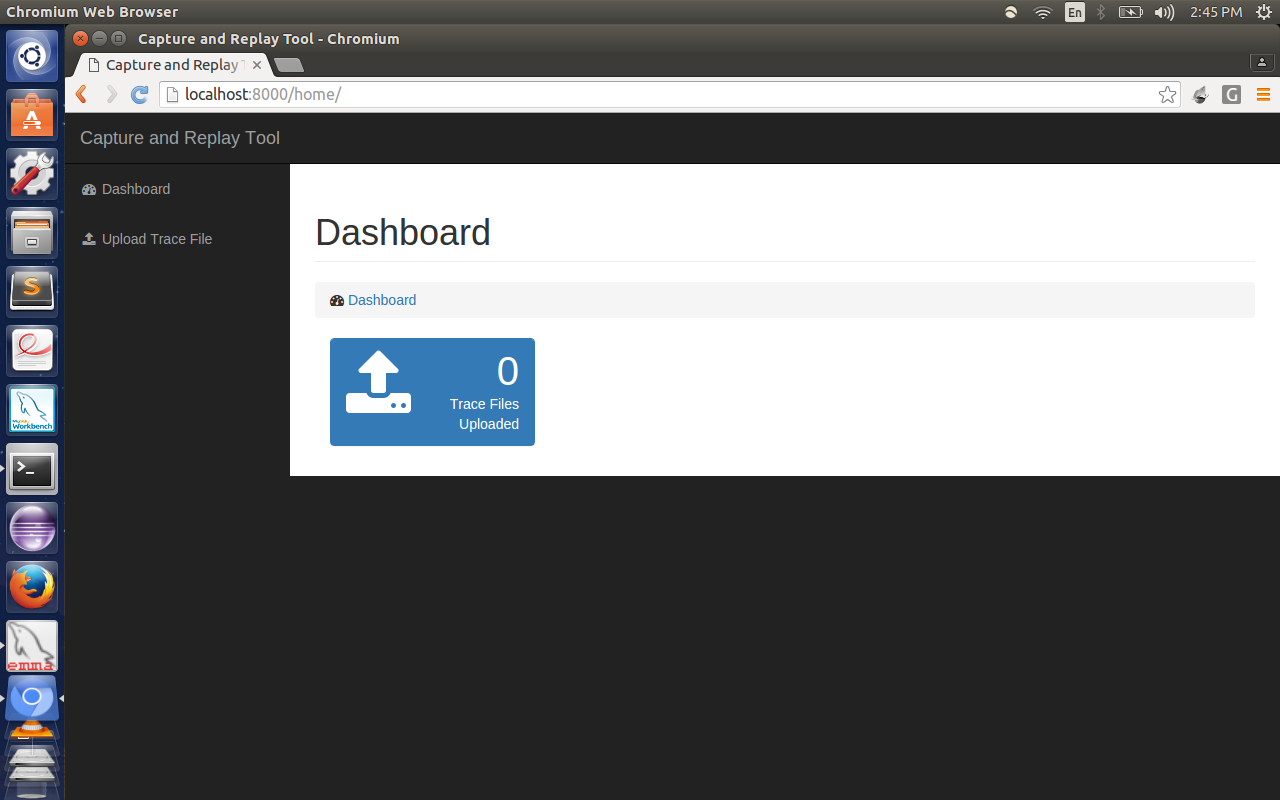


## APPENDIX-A

**Screenshots**

The following screenshots show the various features of the capture and replay tool

* Home Page

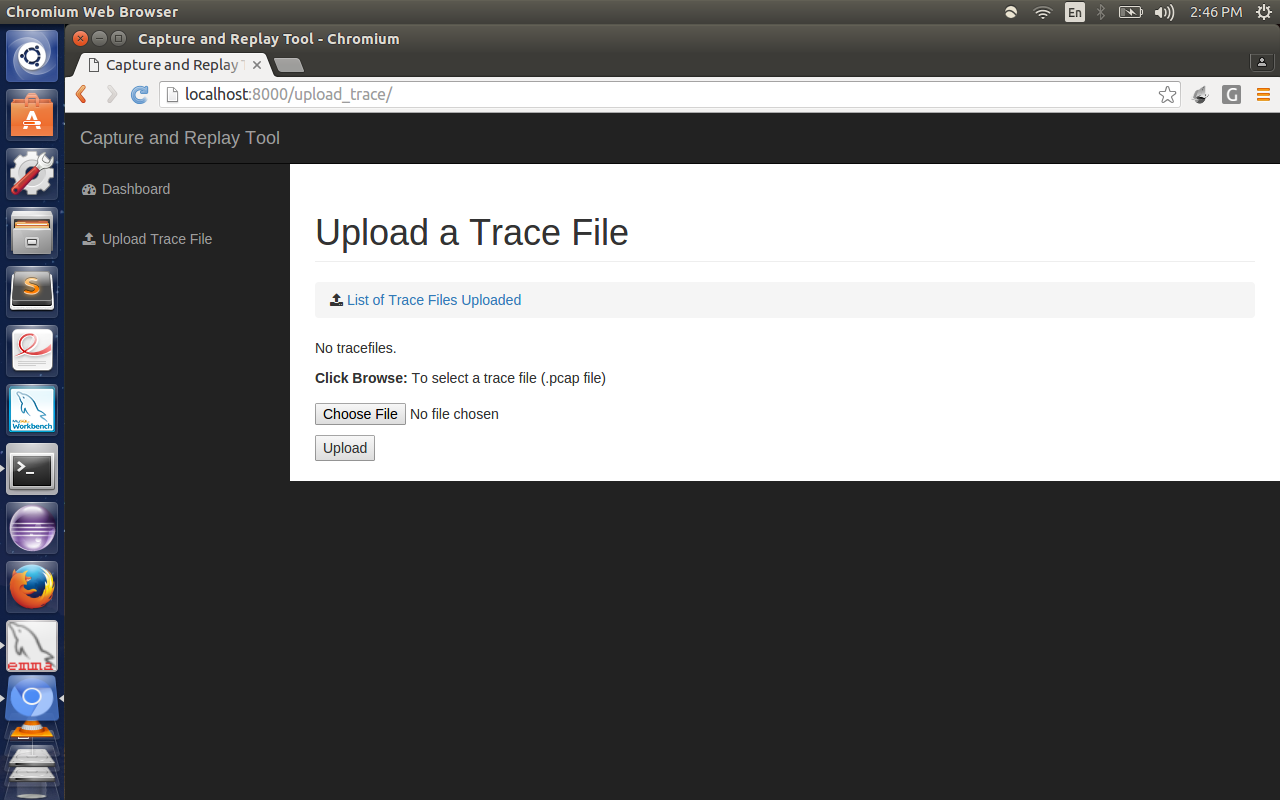


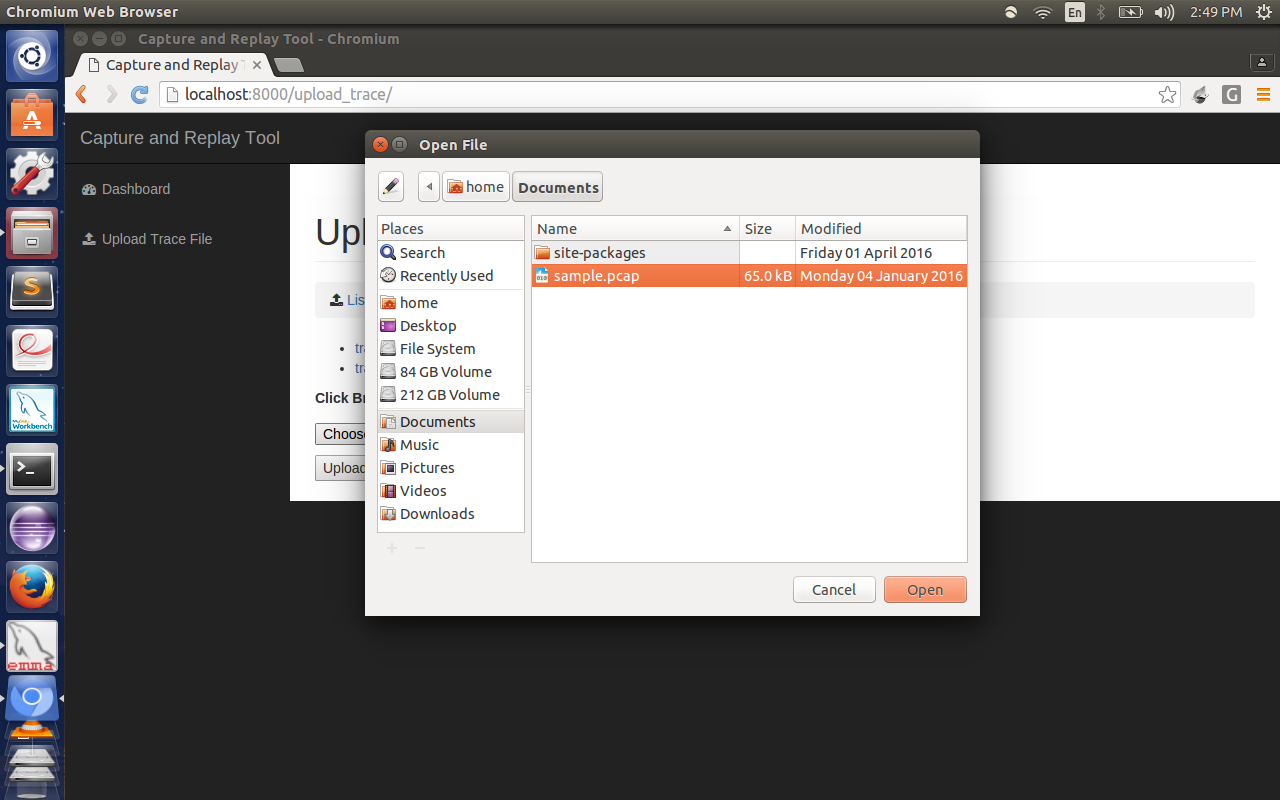
This is the main page of the Capture and Replay Tool GUI. This page displays how many trace files have been uploaded so far.

Using the navigation column on the left hand side, “upload trace file” option can be selected to upload the trace file.

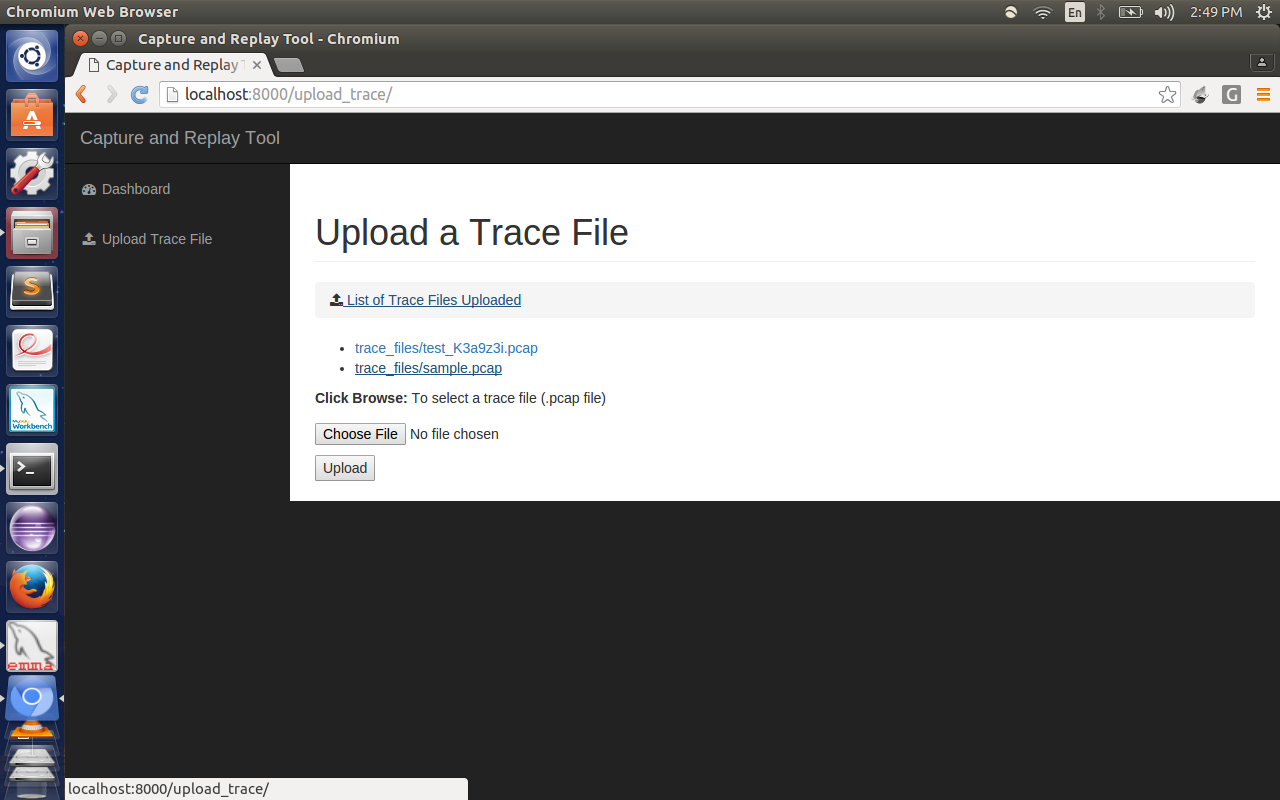
* **Upload Trace File Page**

This page will show the list of trace files that are uploaded (if any). The trace file can be uploaded by clicking the “choose file” button to select a trace file(.pcap file).



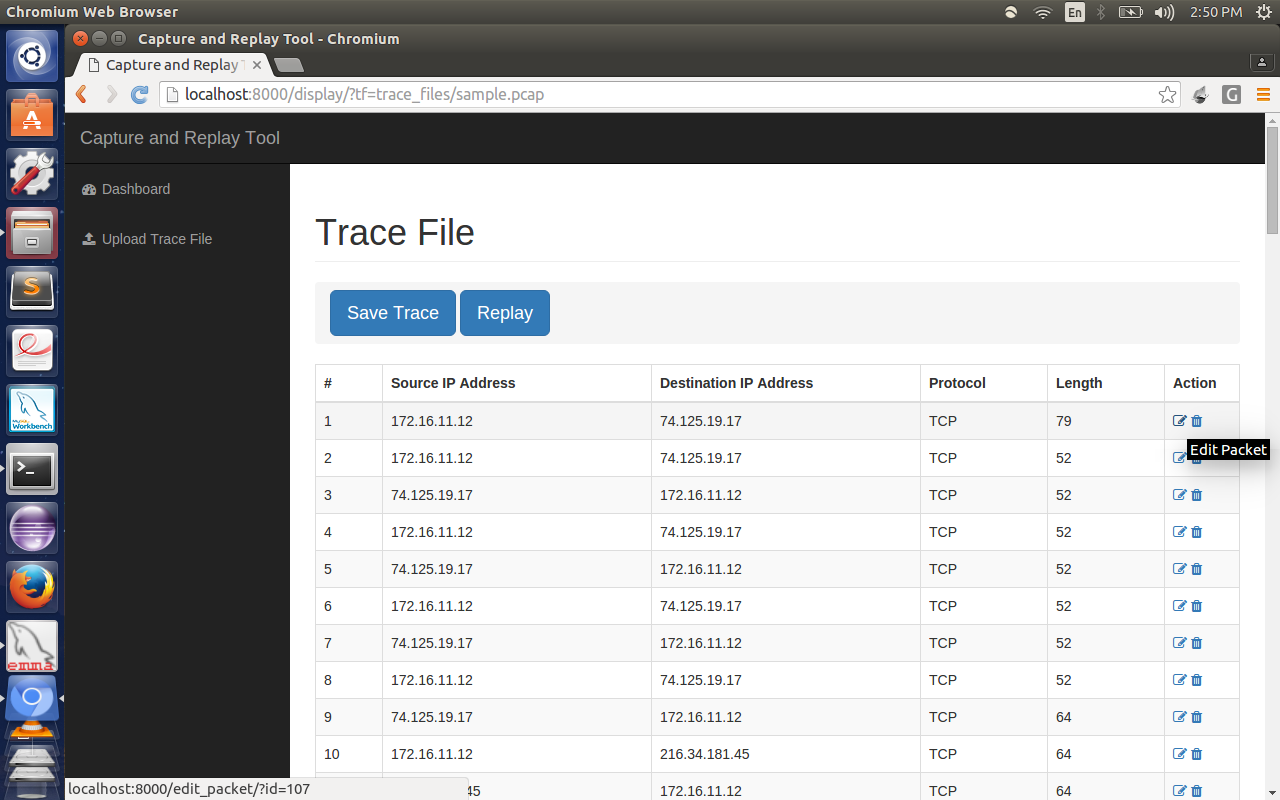


Here “sample.pcap” file is choosen to be uploaded.



This page shows the trace files that were uploaded before. When a trace file that is uploaded is clicked in this page, its contents are displayed in a tabular manner.

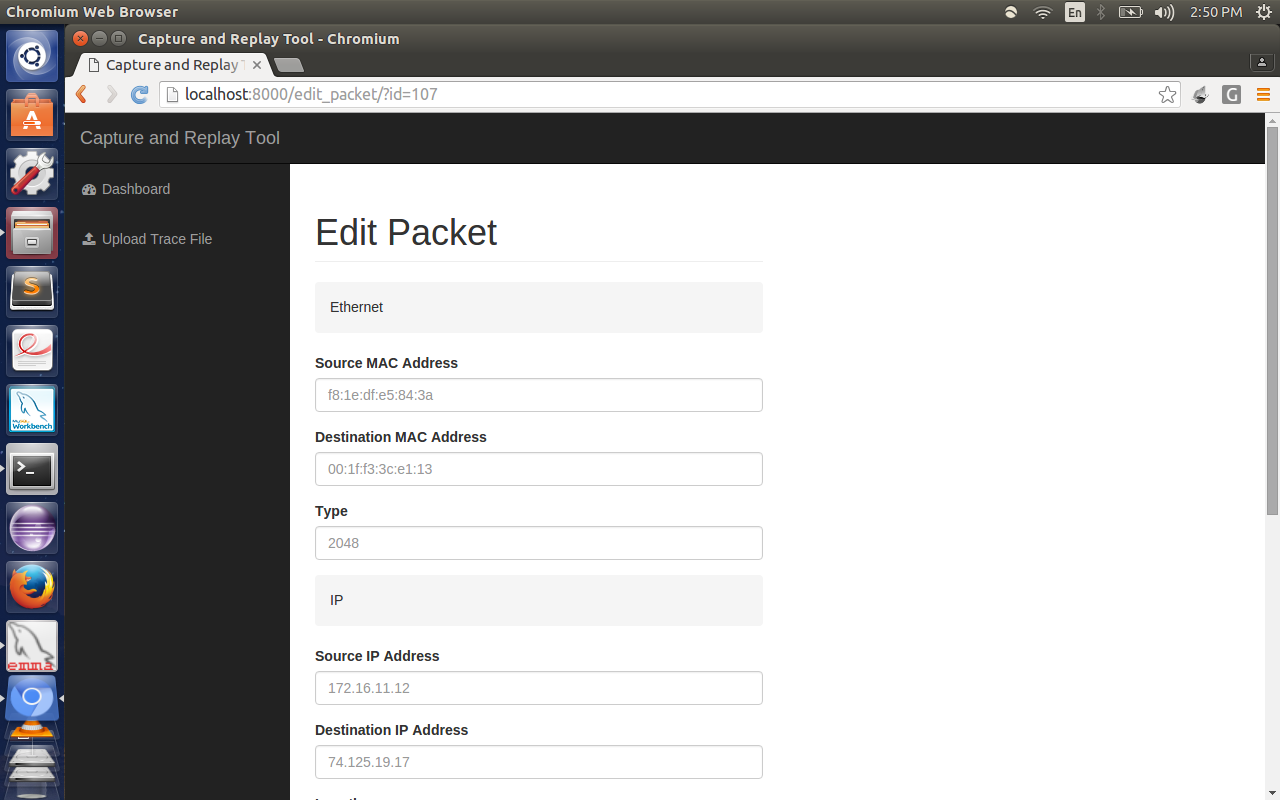
* **Page which displays trace file contents**



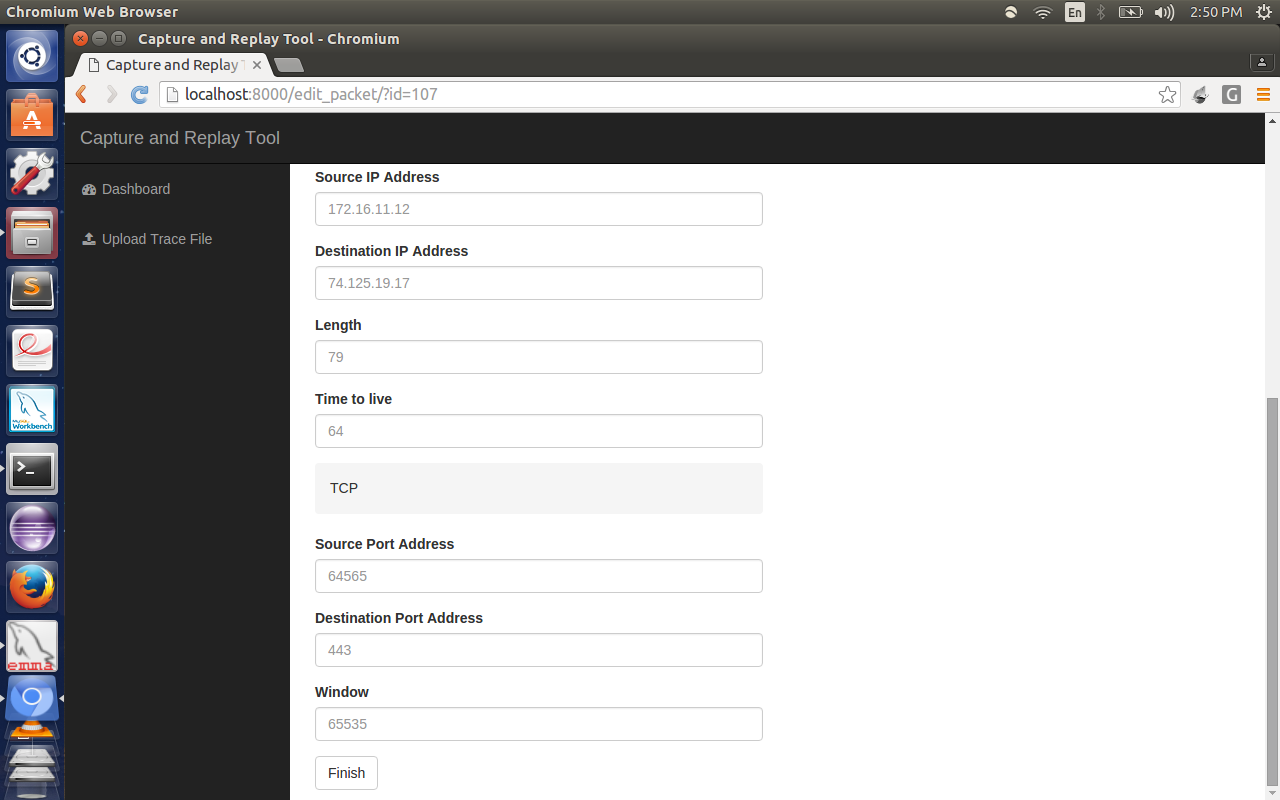
The above page displays the contents of the trace file “sample.pcap”. The table displays the Source IP address, Destination IP address, Protocol and Length of every packet in the trace file. Each packet is provided with two options in the action column: Edit packet and Delete Packet.

When edit packet is clicked for a packet, a new page opens which displays all the editable fields of the packet.

* **Edit Packet Page**

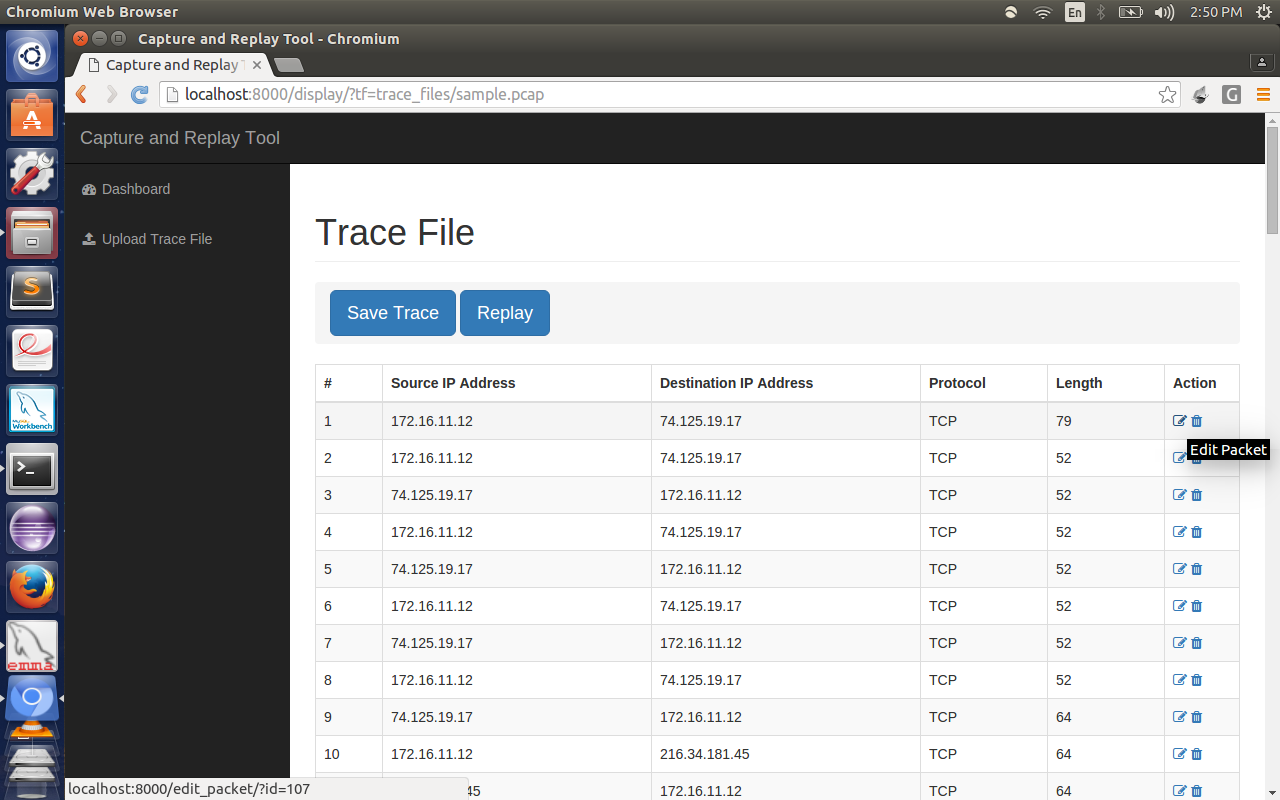


This page shows that Source MAC address, Destination MAC address, Source IP address and Desination IP address can be edited.



This page shows that Source Port address, Desination Port address, etc can be edited.

Once the finish button is clicked, the edited packet will be saved.



The trace file can be saved using the trace file button.

When the replay button is clicked, a new page opens where the packets can be replayed by clicking a button.