Documentation

Network Analyzer

MeLi Challenge Dev

Index

Requirements

Motivation

We need to find someone that is capable to make complex scenarios and analyze its behavior. For this purpose, we have several tools such as Virtual Computers and Traffic samplers. We know that sometimes the behavior is not the expected, so we want to analyze and make graphs of that to have a better understanding of it.

Goal

Build a virtual scenario with at least 3 virtual computers and install at least 2 script in each to make some traffic between them. Install the traffic sampler SFlow or use TCPDump to monitor the activity in the network. Finally, with the output of SFlow or TCPDump make a program that with some algorithms make an analysis of the data and show the script patterns in different graphs and/or tables inside a static html.

Project Architecture

Overview

The project consists of four main modules:

- Virtual scenario
- Network traffic generator scripts
- Data collection script
- Data analysis program

Virtual scenario

The virtual scenario consists of four virtual machines (terminals) which will send data to each other with the Network traffic generator scripts, and a central virtual machine in charge of collecting and processing the data generated by the terminals.

Network traffic generator scripts

Each terminal has three scripts in charge of generating traffic over the network (only to the other three terminals, not to the central computer). Each script sends different type of data at a different rate.

Data collection scripts

Each terminal is in charge of collecting the information of the incoming traffic, and saving it into a file.
The main computer has a script in charge of collecting the data generated by each terminal and merging it into a single file for the Data analysis program.

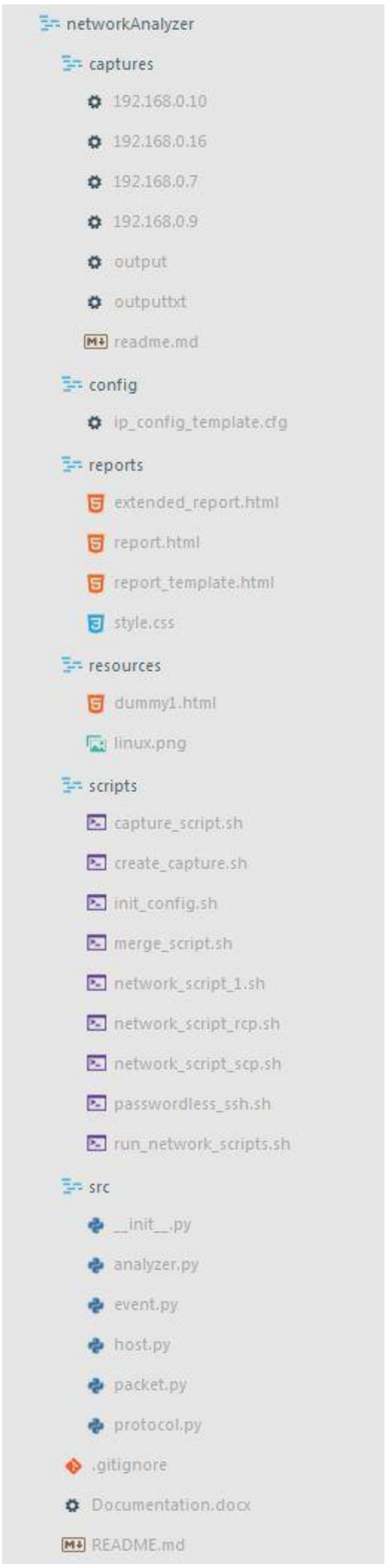
Data analysis program

The Data analysis program receives a file containing information about all of the network traffic, processes it and generates an HTML file, with useful information about the network traffic.

Used software

- ***OS:*** Ubuntu Server 14.04.4 LTS
- ***Virtualization Software:*** Virtual Box 4.3.28
- ***IDE:*** PyCharm Community Edition 5.0.4
- ***Programming Language:*** Python 3.4.3
- ***Version Control:*** Git
- ***Utilities:***
    - TCPDump
    - Mergecap (Wireshark Common package)
    - Pip3 (Python 3 Package Manager)
    - Yattag (Python Library)

Project structure



```
networkAnalyzer
  captures
    192.168.0.10
    192.168.0.16
    192.168.0.7
    192.168.0.9
    output
    outputtxt
    readme.md
  config
    ip_config_template.cfg
  reports
    extended_report.html
    report.html
    report_template.html
    style.css
  resources
    dummy1.html
    linux.png
  scripts
    capture_script.sh
    create_capture.sh
    init_config.sh
    merge_script.sh
    network_script_1.sh
    network_script_rcp.sh
    network_script_scp.sh
    passwordless_ssh.sh
    run_network_scripts.sh
  src
    __init__.py
    analyzer.py
    event.py
    host.py
    packet.py
    protocol.py
  .gitignore
  Documentation.docx
  README.md
```

The project is divided in six folders:

- Captures
- Config
- Reports
- Resources
- Scripts
- Src

Captures

This folder contains the output generated by each terminal while running TCPDump, and the file which contains the merged information, both in binary format and text.

Config

This folder contains a configuration file that has the IP addresses of all the hosts in the network. This file will be used by the scripts to communicate the hosts with each other.
There is a template of the file containing the layout and information that the configuration file must have.

- ***ip_config.cfg***: The file has the following layout:
  - ip_main: IP of the central computer
  - my_ip: IP of the computer containing the file
  - ip_x (one for each terminal, with x being a number starting in 1): IP address of the other terminals

In the central computer, ip_main and my_ip are not needed.
In the terminals, the last ip_x is not needed.

Reports

The HTML reports generated by the Data analysis program will be saved inside this folder, along with its stylesheet.

- ***report_template.html:*** Contains the basic structure needed to generate the report HTML.
- ***report.html:*** Contains the report generated by the Data analysis program.
- ***extended_report.html:*** Same as report.html, with the addition of a table containing every packet sent through the network (takes more time to load).

Resources

This folder contains test files that will be transferred by the scripts in order to generate network traffic.

Scripts

Every script used by the system will be placed here. In order to make them work correctly, they should all be executed inside the scripts folder.

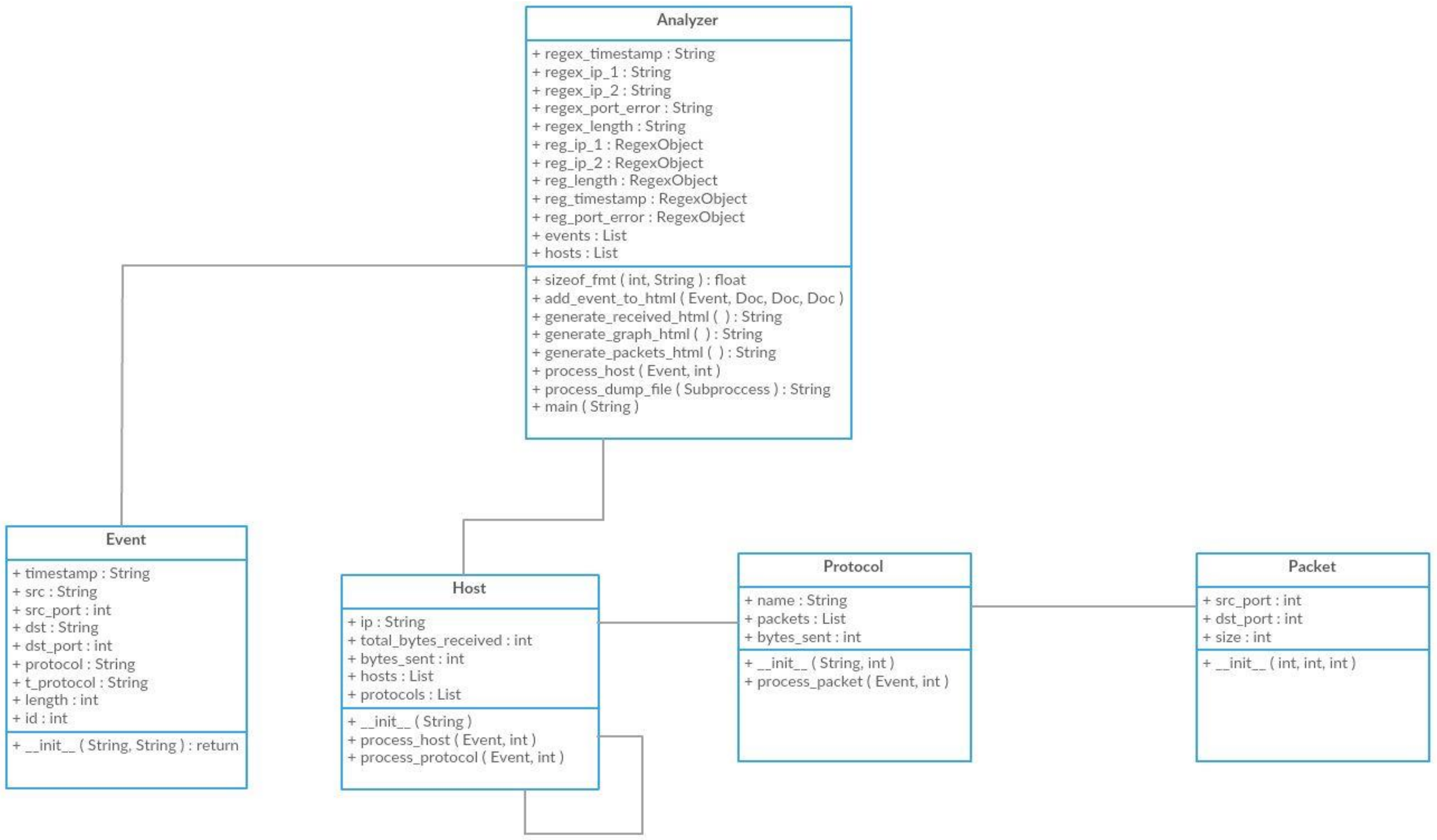
- ***capture_script.sh:*** When executed, is starts the network information dumping process and saves it to a file named as the host's IP. To be used in terminal computers.

- ***create_capture.html:*** Collects dump files from terminals, merges them into a single file and creates a text file out of the merged file. To be used in central computer.
- ***init_config.sh:*** Installs dependencies and files needed for the system to work. To be used in all the computers.
- ***network_script_1.sh:*** Sends an echo request (PING) to every terminal every 10 seconds.
- ***network_script_rcp.sh:*** Copies repeatedly one of the resource files to every terminal, with a delay between 1 and 5 seconds between each transfer to the same terminal. Uses rcp command to copy the file over the network.
- ***network_script_rcp.sh:*** Copies repeatedly one of the resource files to every terminal, with a delay between 1 and 15 seconds between each transfer to the same terminal. Uses scp command to copy the file over the network.
- ***passwordless_ssh.sh:*** Script to create public key and send it to the other hosts in order to connect through SSH without authenticating. To be used in terminal computers. ***IMPORTANT***: run_network_scripts.sh will not work properly if this script is not executed first.
- ***run_network_scripts.sh:*** Script to run all the network traffic generator scripts simultaneously. To be used in terminal computers.
- ***merge_script.sh:*** Merges files passed by argument when executing script and saves file to captures/output.

Src

This folder contains the Data analysis program.

Classes diagram


Analyzer
+ regex_timestamp : String
+ regex_ip_1 : String
+ regex_ip_2 : String
+ regex_port_error : String
+ regex_length : String
+ reg_ip_1 : RegexObject
+ reg_ip_2 : RegexObject
+ reg_length : RegexObject
+ reg_timestamp : RegexObject
+ reg_port_error : RegexObject
+ events : List
+ hosts : List
+ sizeof_fmt (int, String) : float
+ add_event_to_html (Event, Doc, Doc, Doc)
+ generate_received_html ( ) : String
+ generate_graph_html ( ) : String
+ generate_packets_html ( ) : String
+ process_host (Event, int)
+ process_dump_file (Subproccess) : String
+ main (String)
Event
+ timestamp : String
+ src : String
+ src_port : int
+ dst : String
+ dst_port : int
+ protocol : String
+ t_protocol : String
+ length : int
+ id : int
+ __init__ (String, String) : return
Host
+ ip : String
+ total_bytes_received : int
+ bytes_sent : int
+ hosts : List
+ protocols : List
+ __init__ (String)
+ process_host (Event, int)
+ process_protocol (Event, int)
Protocol
+ name : String
+ packets : List
+ bytes_sent : int
+ __init__ (String, int)
+ process_packet (Event, int)
Packet
+ src_port : int
+ dst_port : int
+ size : int
+ __init__ (int, int, int)

Classes Description

analyzer.py

Class used as Main. It is in charge of parsing the dump file, processing it and generating the HTML reports.

event.py

Class containing information about network events. Each event logged in the dump file, and considered relevant, is an instance of this class.

host.py

Class containing information about each host in the network. Each instance of this class is a different host. It also helps analyzer.py in the processing of the dump file.

protocol.py

Class containing information about every protocol used by each host. Each instance of this class is a different protocol. It also helps analyzer.py in the processing of the dump file.

packet.py

Class containing information about the data transferred through the network. Each instance does not actually represent a packet *per se*, but a whole file (group of packets) instead.

System Lifecycle

Step 1

Create five virtual machines, each one running Linux OS (preferably Ubuntu). All the computers must be in the same network. While installing, when prompted which packages should be installed, select OpenSSH.
The software used to create the virtual machines in this project was VirtualBox.

Step 2

Download the project to each computer.

Step 3

Inside the scripts folder, run the init_config.sh script in order to download the necessary dependencies and programs.

Step 4

Run the ifconfig command on the shell on every computer, in order to find out the IP address of each computer. Inside the config folder, copy the file ip_config_template.cfg with the name “ip_config.cfg” (without quotes). Open the new file and fill in the required information. For more information on what each field should contain, refer to the “Project Structure > Config” section in page 5.

Step 5

Inside the script folder, execute the passwordless_ssh.sh script on the terminal computers. Create the public key in “home/user/.ssh/id_rsa.pub” where user is the username of the host. When prompted to enter a passphrase, leave it blank and press enter.

Step 6

Execute the capture_script.sh script on the terminal computers. Doing this will make TCPDump start recording.

Step 7

Execute the run_network_scripts.sh script on the terminal computers. Traffic should start to be generated between the terminal computers.

Step 8

On the central computer, run the create_capture.sh script. It will collect the dump files from the terminals, merge it to a single file and save it inside the captures folder, under the name of output.

Step 9

On the central computer, run the Data analysis program with the command “python3 analyzer.py dump_location” where dump_location is the location of the merged dump file. If Python 3 is not installed, run the command “python” instead.