**Features-Extractor**

Work inspired by E. Viegas, A. Santin and V. Abreu's paper:  
"Enabling Anomaly-based Intrusion Detection Through Model Generalization".

## About the project

The goal is to recreate an Intrusion Detection System (IDS) by training a Machine Learning model based on the traffic recreated within a virtual environment. The traffic generated is difficult to use to train machine learning models as it is scenario dependent, so it would lead to models trained for that specific scenario. To solve this problem it is necessary to treat the traffic generated in a way that is independent of the simulated session (virtual or real environment).

## Workflow

The traffic generated (HTTP, SMTP, SMNP, SSH) is listened to by tcpdump; The generated .dump file is converted into a file called totaltraffic.c containing array C using wireshark; Featuresextractor.py is launched; In the end, 50 features independent of the scenario are obtained and can be used for model training.

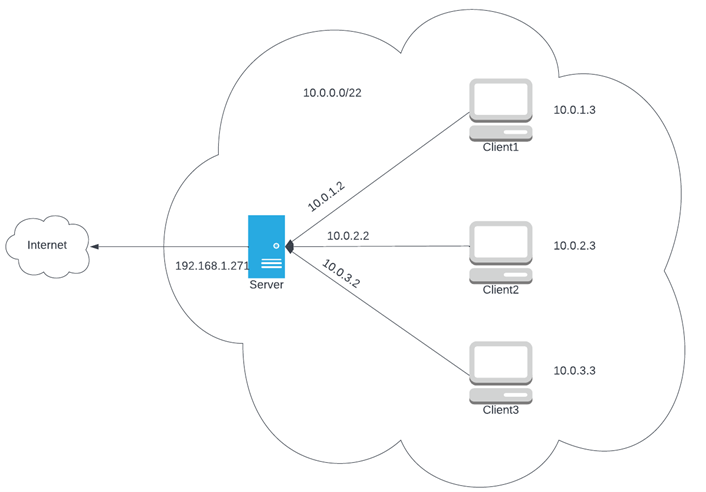
50 Features + Target:

1. IP\_TYPE
2. IP\_LEN
3. FR\_LENGHT
4. IP\_ID
5. IP\_RESERVED
6. IP\_DF
7. IP\_MF
8. IP\_OFFSET
9. IP\_PROTO
10. IP\_CHECKSUM
11. UDP\_SPORT
12. UDP\_DPORT
13. UDP\_LEN
14. UDP\_CHK
15. ICMP\_TYPE
16. ICMP\_CODE
17. ICMP\_CHK
18. TCP\_SPORT
19. TCP\_DPORT
20. TCP\_SEQ
21. TCP\_ACK
22. TCP\_FFIN
23. TCP\_FSYN
24. TCP\_FRST
25. TCP\_FPUSH
26. TCP\_FACK
27. TCP\_FURG
28. COUNT\_FR\_SRC\_DST
29. COUNT\_FR\_DST\_SRC
30. NUM\_BYTES\_SRC\_DST
31. NUM\_BYTES\_DST\_SRC
32. NUM\_PUSHED\_SRC\_DST
33. NUM\_PUSHED\_DST\_SRC
34. NUM\_SYN\_FIN\_SRC\_DST
35. NUM\_SYN\_FIN\_DST\_SRC
36. NUM\_FIN\_SRC\_DST
37. NUM\_FIN\_DST\_SRC
38. NUM\_ACK\_SRC\_DST
39. NUM\_ACK\_DST\_SRC
40. NUM\_SYN\_SRC\_DST
41. NUM\_SYN\_DST\_SRC
42. NUM\_RST\_SRC\_DST
43. NUM\_RST\_DST\_SRC
44. COUNT\_SERV\_SRC\_DST
45. COUNT\_SERV\_DST\_SRC
46. NUM\_BYTES\_SERV\_SRC\_DST
47. NUM\_BYTES\_SERV\_DST\_SRC
48. FIRST\_PACKET
49. FIRST\_SERV\_PACKET
50. CONN\_STATUS
51. TYPE

NOTE:

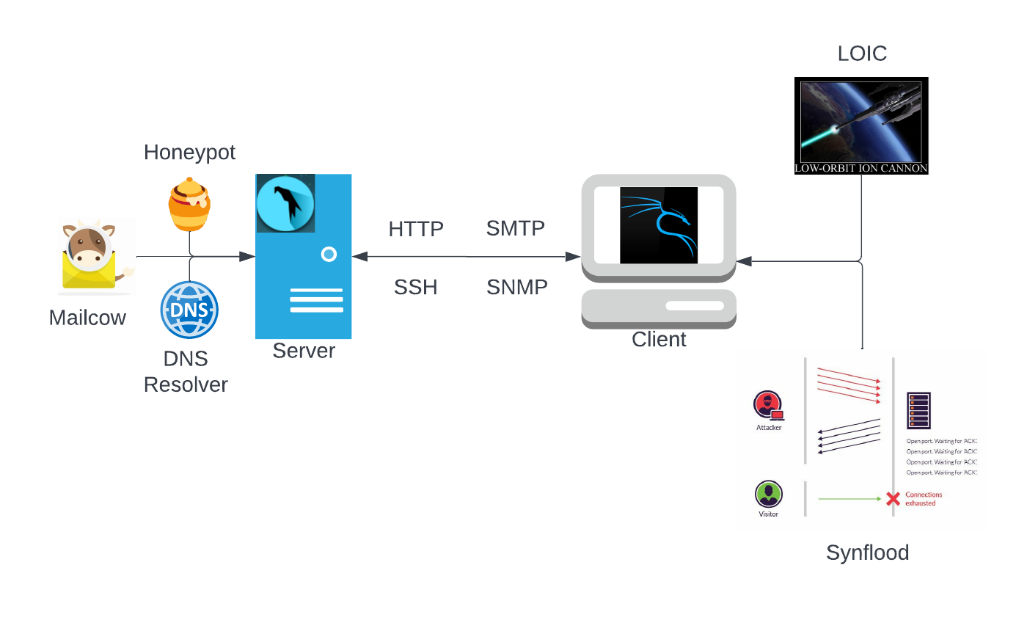
SRC\_DST and DST\_SRC indicate the direction of the transmission which will be sent and received respectively. The target variable can take on two values: Attack: if the packet comes from the client; Normal: if the packet comes from the server;

You need to create a virtual environment like the one shown in the figure:



You can use any virtualizer, the important thing is that the client machines can only communicate with the server. The server is the only access point to the internet and takes care of providing connectivity to clients. The goal is to create an environment that is as isolated as possible.

Client and server implement different types of services:



To implement the described scenario, Debian-based distributions (ParrotOS and Kali Linux) were used. The following configuration in \ etc \ network \ interfaces can be used for each client

#Client1 configuration (dhcp or static)

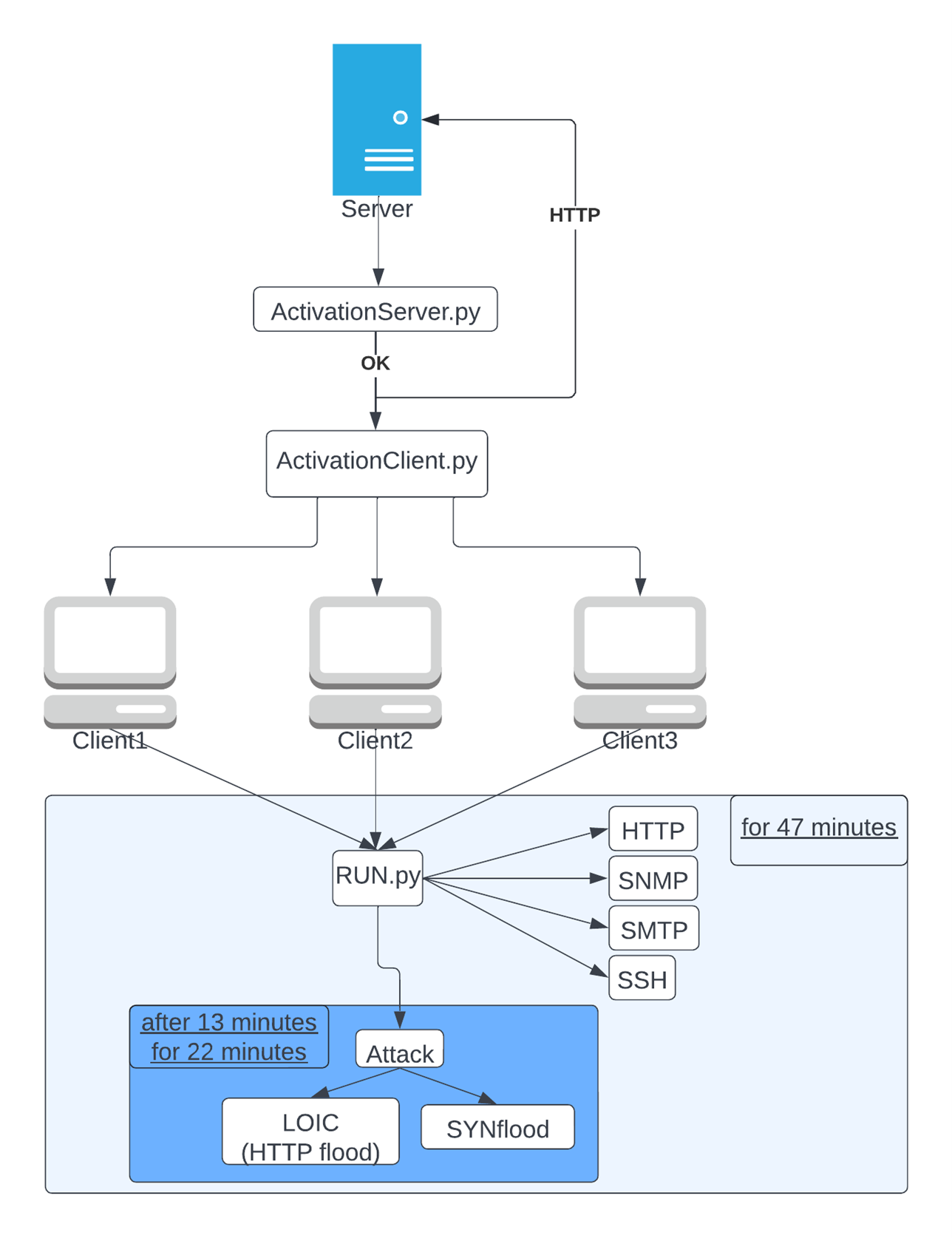
auto eth0

iface eth0 inet dhcp

#Default gateway

post-up route add default gw 10.0.1.2

To automate the traffic acquisition process, the clients have been synchronized with the server following the scheme shown here:



ote: RUN.py is a script that runs one of the 4 scripts shown and after a certain period starts both the LOIC and SYNflood attack.

LOIC is required to generate the HTTPflood attack (downloadable from the link: <https://sourceforge.net/projects/loic/>).

To use it you can use mono:

sudo apt install apt-transport-https dirmngr gnupg ca-certificates

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF

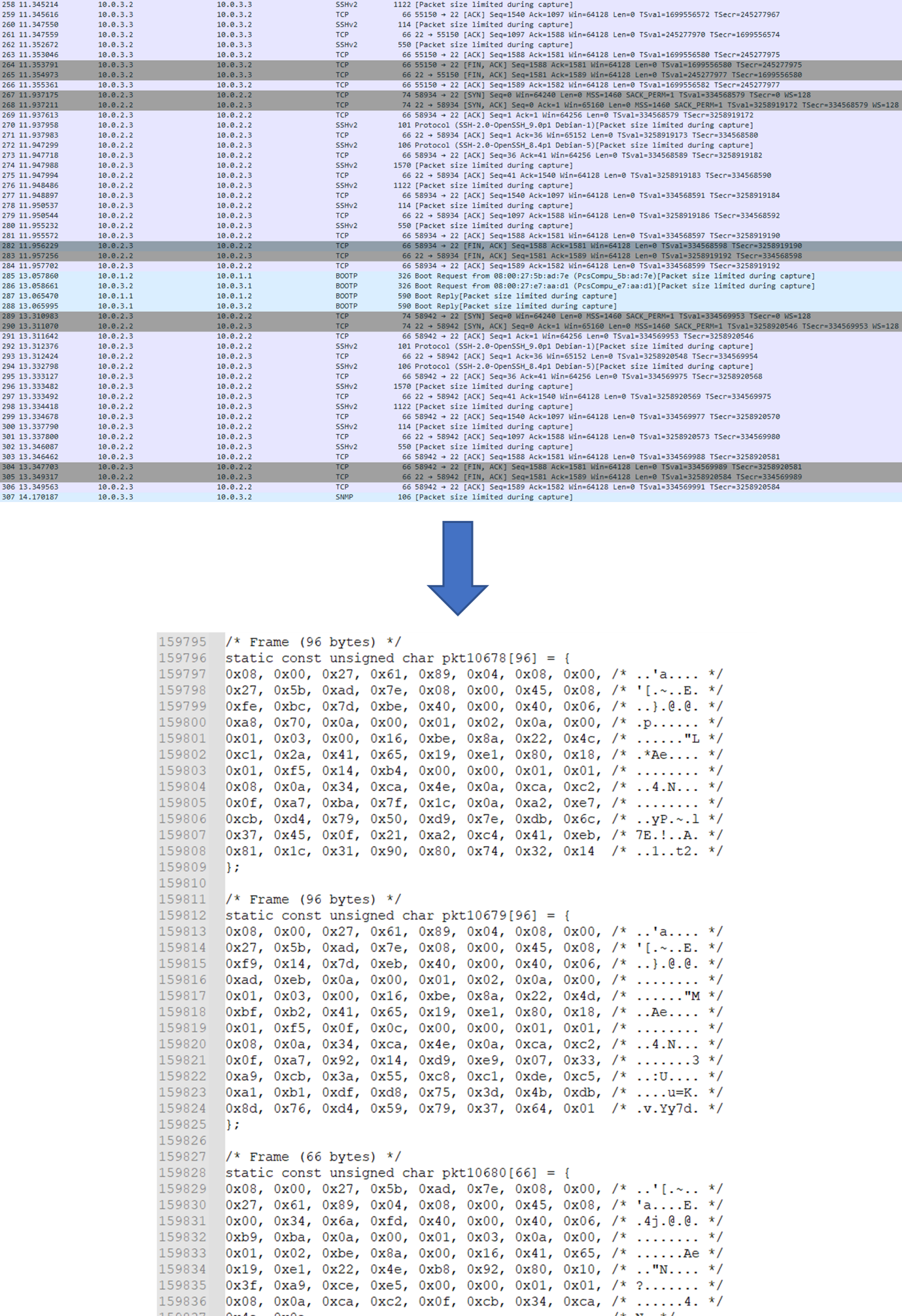
echo "deb https://download.mono-project.com/repo/debian stable-buster main" | sudo tee /etc/apt/sources.list.d/mono-official-stable.list

sudo apt update

sudo apt install mono-devel

For synflood use the metasploit suite

To use featuresExtractor you need to convert packages to C array. You can use Wireshark for this purpose:



The file to be obtained must have the structure shown, featuresextractor will take care of extracting the information and creating the dataset.

To the featuresExtractor.py it is necessary to pass first the list of IPs of the Server interfaces and then that of the Client. Order is important.

Here is an example to understand how to use featuresextractor. The files used (downloadable from example) must be present in the same folder where the script is launched.

First of all we extract the file:

tar -xvf example.tar.gz

We run the script:

sudo python featuresExtractor.py [10.0.1.2,10.0.2.2,10.0.3.2] [10.0.1.3,10.0.2.3,10.0. 3.3]