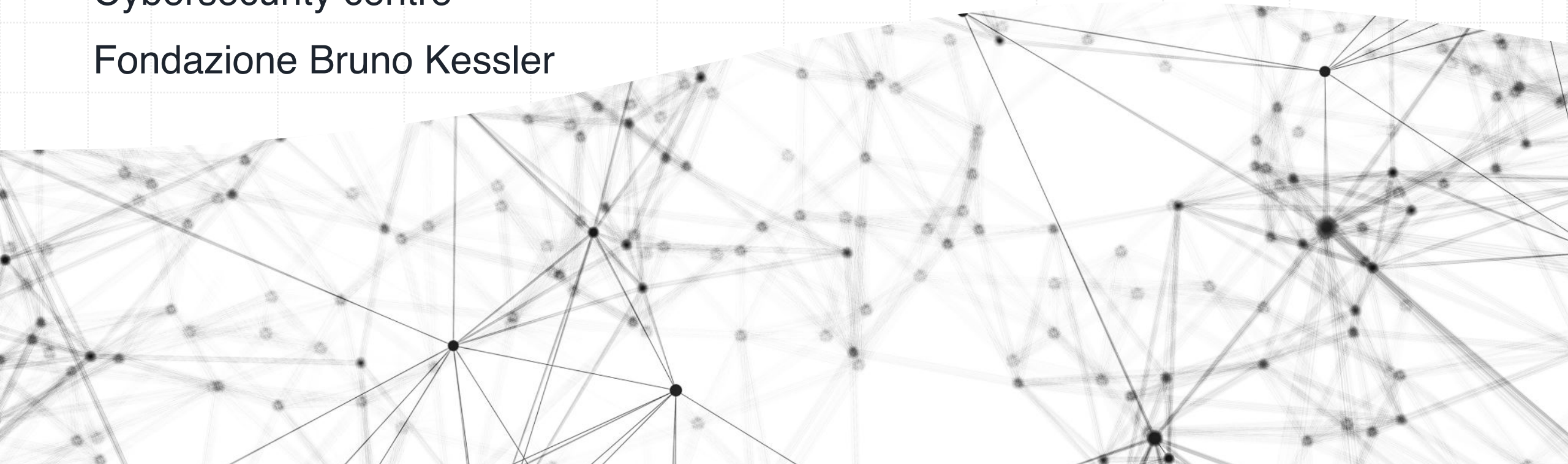
System Deployment and testing

Roberto Doriguzzi Corin

Cybersecurity centre

Fondazione Bruno Kessler



Outline

- Implementation of a stand-alone DL-based IDS
- Local testing
- Deployment and testing the IDS on a GPU-enabled system

Stand-alone IDS

A program that can be **installed** on a target machine (different from the one used for development) and that can be executed **in background**

- Such a program accepts parameters from the **command line** (e.g., the location of the pre-trained model(s), the network interface to monitor)
- It can access the **ingress network interface**(s) for traffic collection
- It **writes its output on the file system**, on the standard output, or to communication channel(s) (e.g., Local/Remote Procedure Channels such as HTTP, MQTT, UDP, etc.) where a decision system might be also connected

Laboratory: from Junyper notebook to python program

1. First, train and **save** an ANN model with the code of laboratory 03-Hyperparameters (or use one available in the “Models” folder with suffix IDS2017
2. The first part of this laboratory consists of converting a DL-based IDS written as a Juniper notebook into a stand-alone Python program
3. The python program must be able to support the following command-line arguments:
   - Path to the ANN model
   - Path to the test set and…
   - Path to the ingress network interface (alternatively, the path to a network traffic trace)
   - Path to the output CSV file where the program writes the classification results

Local testing

In this experiment, we verify that **the system works on the development machine**

- Start the stand-alone Python IDS and verify that everything is working as expected
- Two configurations can be tested at this point:
  - Local testbed
  - Local pre-recorded traffic trace

Final deployment and testing

Finally, we deploy our system on a GPU-enabled embedded device and we test is by generating attacks either with a traffic generator tool, or by replaying a pre-recorded traffic towards the embedded system.

- Connect the device to your laptop with a ethernet cable and configure a point-to-point connection (**IP of the device is 192.168.0.3**)
- Upload the stand-alone python program and the DL model to the device
- Execute the IDS so that it monitors the Ethernet device connected to the laptop
- Start the network attack using `tcpreplay`

Useful tips

Packet fragmentation

- As the packets of the trace are bigger than the MTU of your card, they will be dropped.
- You can either increase the MTU of your interface with command
  - `$ ifconfig eth0 mtu 15000` or
- Truncate packets to the MTU length (default 1500 bytes):
  - `$ tcprewrite --mtu-trunc --infile=input.pcap --outfile=output.pcap`

- You can test with different MTU sizes (remember that packet length is one of the features

Useful tools

Replay a pre-recorded traffic trace through a network interface

- `$ sudo apt install tcpreplay`
- `$ sudo tcpreplay-edit -i eth0 ./traffic_trace.pcap`
  - `(–M PACKET_RATE (in Mbps) or –t for topspeed)`
- Use options `--enet-dmac` and `--enet-smac` to override the MAC addresses of the packets if needed

Example

```
tcpreplay-edit -i en0 --enet-smac f8:4d:89:88:2a:52 --enet-dmac
00:04:4b:e5:7a:e1 ./IDS2017-dataset.pcap
```