Project Assignment a.a. 2021-22

A project can be done by groups with no more than 3 persons (3 or 2 persons is recommended).

Each group should implement the project with the related tool, produce a small report (up to 5 pages) and present the work with slides (15 minutes).

During the presentation every member is requested to discuss a part of the project and run the project on its computer (possibly apply small modifications on the fly during the presentation).

Steps for the project:

1. Choice of the project and communication of the student names to the teachers
2. Schedule meetings to discuss about work done in the project and results
4. Submission of the project:
   - the project must be documented including the design choices
   - the code and the documentation must be submitted as a single zip file, at least two days before the exam to the following addresses: cinzia.bernardeschi@unipi.it, maurizio.palmieri@ing.unipi.it

Project evaluation: unsatisfactory, satisfactory, good. To attend the oral test, it is necessary to have a positive evaluation of the project. The evaluation is a bonus for the final mark.

Project 1

Design the “Triple Modular Rundundancy Multiprocessos Model”, a modified version of the “Fault-Tolerant Multiprocessos Model” available in the Mobius tutorial section https://www.mobius.illinois.edu/wiki/index.php/Tutorials.

Computers are organised according to Triple Modular Rundundancy fault tolerant architecture.

The Computer element is the same used in the tutorial.

The Voter is composed of 4 chips, of which 1 is a cold spare.
Assume:

- Failure rate of the voter chip = 1/10 failure rate of the computer chip
- Failure rate of the spare chip = 1/2 failure rate of the voter chip

Compare the unreliability of the original “Fault-Tolerant Multiprocessor Model” from the tutorial with the “Triple Modular Rundundancy Multiprocessos Model” objective of this study assuming a mission time of 20 years.

Project 2

Consider the Headlamp system of an autonomous car.

Starting from the system high level architecture and the operational environment, analyse threats and damage scenarios related to safety.

The Headlamp system definition reported in the Annex H of ISO/SAE DIS 21434 will be provided.

Develop an ADVISE model in Mobius for security evaluation, which consists of

- an attack graph with roughly 20 elements (knowledge, access, skill, attack step and goal).

- 3 different adversary profiles

Use reasonable values for the parameters of the model.

Evaluate the probability of each adversary to achieve the goals.

Analyze the results.

Project 3

Consider a Platooning scenario of terrestrial vehicles. Platooning relies on closed-loop co-ordination among vehicles to achieve stability and short inter-vehicle distance.

Co-simulate a scenario with an autonomous platoon of 3 cars following a leader car with a desired distance of 15 meters.

The required fmus for the controller (Cooperative Adaptive Cruise Control control law) and the vehicle model will be provided.

Analyse reasonable data alteration attacks that can lead to a crash between any two car.

An attack can last from a few steps to many steps. Consider also an attack that does not lead/ leads to a crash, depending on short-duration or long-duration.

Students can use the attack fmu introduced during the course to implement the data alteration attack.

Project 4

Consider the Android malware archive at the following link (disable the antimalware before executing malware analysis): https://mega.nz/file/RINQzQKT#KDnuXqL1jmpF0Qrb8Vo2V_jbrJLBsUL6N3Wy2z4Sg_Y

The archive contains malware belonging to three different families: fakebank, overlay, reddrop.

The objective of the project is: choose one malware family and write a report on the analysis of the different samples of the selected family (the archive contains 4 samples for each family). The idea is to find the malicious payload in the different samples of the selected family.

The report must include the following sections:
1) Antimalware analysis: by submitting the samples to the tools web VirusTotal and Jotty;

2) Static analysis: by using bytecodeviewer (https://github.com/Konloch/bytecode-viewer) to highlight interesting snippet of code (include and discuss such code in the report); by using MobSF framework (https://github.com/MobSF/Mobile-Security-Framework-MobSF) to extract automatically features from the code interesting for malware analysis.

3) Dynamic analysis: use MobSF to execute the application in a controlled environment (a simulator) and observe the beahvior of the application.