Meeting 19.10.22 – Notes

General overview:

Product penetration testing (simulating attacks) - identifying vulnerabilities, firmware analysis – now automated.

*Embedded devices- every manufacturer does things differently. Different operating systems etc.

Goal:

-create environment fulfilling expectations (get it up and running FAST)

-use for test chain (no critical vulnerabilities)

-testing different firmware (research perspective)

EMBA and EMBARK

EMBA - core of the firmware analyser (command line interface); extraction of firmware and identification of vulnerabilities. Structure workflow to optimize it and make it reproducible

EMBARK - previous AMOS project.

Identifying system components in an automated way.

General goal: Which exploits and vulnerabilities are in the software?

Docker container does analysis things. Aggregator puts everything together.

EMBArk will start EMBA in the background and present you with a dashboard.

So it is a download environment, puts everything in the database. NOW we want to automate whole analysis process, with huge amount of firmware.

In longer perspective: web environment. Then showing graphs-how many vulnerabilities are in the firmware in updates etc.

EMBA and EMBArk on GitHub:

https://github.com/e-m-b-a

EMBA: at the end creates HTML report.

EMBA download - 3 commands, but will be 5-6 GB (!)

Mike can send us first firmware to test.

Our Project:

Create firmware downloader (search for firmware, download, store on hard disc in database and then we can analyse it)

We might not need EMBArk at all, but only as a possibility to feed everything into embark, but we DON'T NEED TO DO IT. So we can start with EMBA only.

1. Start with analysis process with EMBA, only if we have time at the end we can do EMBARK (it's optional). It’s because we want stable, robust core (downloading modules can be challenging). We need core organizing everything centrally.

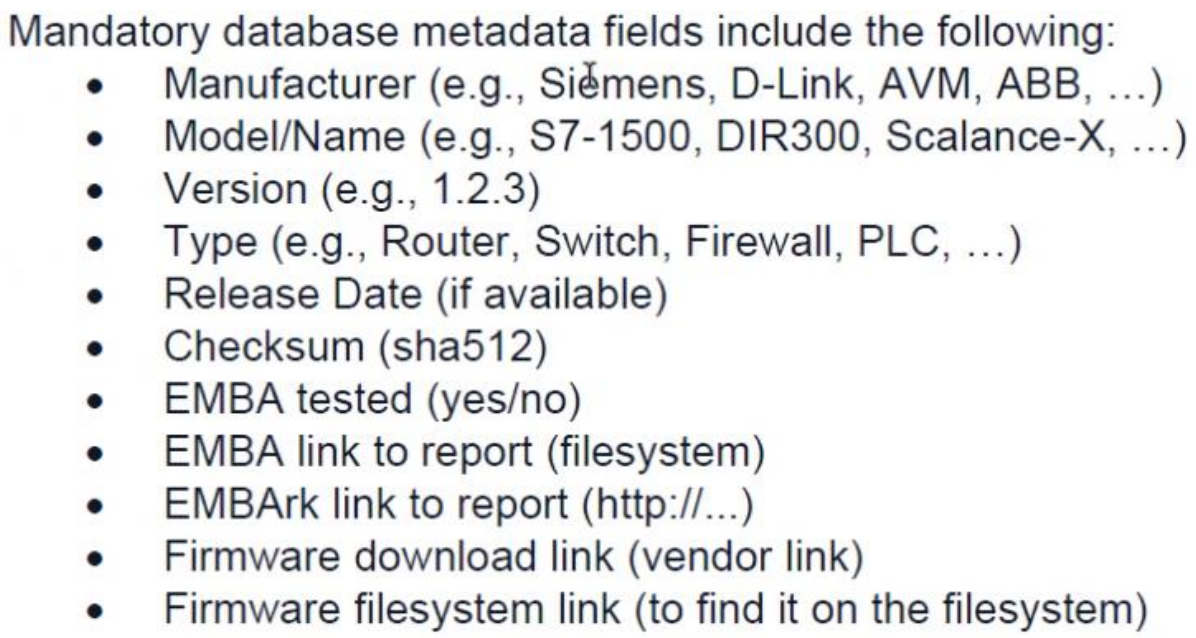
2. We need to analyse websites of vendors, how do we get firmware from there (sometimes you need to log in etc). Find the firmware, metadata. It's all vendor dependent.

3. Burp suite - You can see all the request your browser does for the website. We'll need to analyse communication (how to find download firmware) and then rebuild it in python.

4. It needs to handle different modules from different vendors

5. Before downloading checking if we already downloaded it.

6. Analyse website for metadata



Checksum- making sure we don't have the same firmware multiple times

We will SKIP EMBARK integration at first - so no EMBArk link to report.

7. Good logging mechanism - we are trying to access the website, but the links are not available anymore, we need to check older firmware - if they are still available, so we can assume if the website completely changed or is temporarily unavailable.

And if we tried 3 times in couple days then we deactivate the module that looks broken.

- Core must be robust
- Logging must be good (ex. whole website changed) or to see ex. log-in mechanism – and if they ex. disactivated our account
- Need to check if the firmware is already in our file system

ORGANISATION of GitHub:

Introduce code quality scanners as soon as possible (so that we have high quality default state).

Ex. naditer, pyconstyle, shellcheck, pylinter - we could also write our own check project script, so at least python thing.

https://github.com/e-m-b-a/embark/blob/main/dev-tools/check_project.sh

There is also some code scanning in github in CodeQL

- VMware running a current Kali Linux (currently we are using 2022/02) or Ubuntu:jammy (22.04 LTS) – SOFTWARE MUST BE OPTIMIZED FOR THESE TWO!

The wiki is kept up to date - check it out

Questions:

-how many firmware do we analyse? The more the better.

-Include intelligent load balancing not to overwhelm the website of the vendor. So don’t do it as fast as possible, do it "smart".

-How many firmwares to download? - should be capable of downloading all. But we don't need to download couple of hundreds if the mechanism is the same. If the mechanism is different in the same website we'll need samples of all these different ones.

-The project should provide information WHAT IS THE ISSUE, ex. URL changed, is gone. You must log in, special link is gone, the firmware with ID xxx is not available anymore but 002 is available. We need to be as error tolerant as possible.

-We should then ex. check it again a week later - INCLUDE IDEAS around it,

Email soon: FEW POSSIBLE TIME SLOTS FOR A WEEKLY 30min meeting - between 8.30 - 16