02270 Cybersecurity Fundamentals

Protocol Security Exercises

The exercises use the protocol analyzer OFMC 2024 that is found on DTU Learn. The distribution includes executables for Windows, Mac and Linux.[1] Note that this has no GUI, but should be called from the command line interface. You may need to convince your operating system that this is not malware you downloaded from the Internet :-) You can put the executable into the folder where you have your example files, or more conveniently, put OFMC into a folder that is included in the PATH variable where your operating system looks for executables.

The Example from the Lecture In the lecture, we have demonstrated the Needham-Schroeder Public-Key Protocol (NSPK) as an example. This is given as the file `nspk.AnB`. You can check this file with OFMC by simply using the command `ofmc nspk.AnB` This should give you the attack trace from the lecture.

The tasks below are all of the following form: you are given a protocol (taskX.AnB on Learn) that has an attack and you should understand the attack and find a fix for it. For NSPK, there is the fix proposed by Lowe in the file `nsl.AnB`. Note that for a correct protocol, OFMC will not terminate but check for more and more parallel sessions, unless you specify a certain number of sessions as follows: `ofmc --numSess 2 nsl.AnB` Two sessions are usually sufficient. The size of the search tree is more than exponential in the number of sessions, so it may appear like OFMC "hangs" (although it does not). You can always stop OFMC with Control-C. (It may take a moment to actually stop).

In all the tasks below, the aim is to fix the protocol without changing the goals or the initial knowledge of the participants, i.e., we want a solution for the original objective of the protocol - only changing the exchanged messages. For this reason, we have a capture-the-flag mode that is given the original protocol and the fixed version and checks that initial knowledge and goals are the same:

`ofmc --numSess 2 --ctf` *TaskX.AnB YourFixX.AnB*

This would stop with a `CTF compliance` error, if the file to analyze (*YourFixX.AnB*) deviates from the reference file (*TaskX.AnB*) in goals or knowledge (or the use of channels).

You can check that NSL is a valid answer to the challenge of NSPK:

```
ofmc --numSess 2 --ctf nspk.AnB nsl.AnB
```

[1]For compiling the sources yourself, you need the Glasgow Haskell Compiler.

Your mission, should you choose to accept it... are the following points:

- Understand the objective of the protocol, looking just at initial knowledge and goals: what is the initial security relationship between roles, and what new things are established or transmitted?
- Understand how the protocol is supposed to work: what steps are done and why? Are there any elements that are unclear to you, i.e., why is a certain element really necessary? Feel free to play with it and temporarily remove an element that you don't understand.
- Understand the attack by OFMC: map every step of the attack to a step of the protocol. You should see that each message the intruder sends can really be constructed by the intruder, and that each reply from an honest agent corresponds to a legal step in the protocol execution by this agent.
- Where did it go wrong? Where did something happen that the protocol designer probably did not intend to happen like this? What can be done to change this?

Exercises for week 4
See `TaskX.AnB` on DTU Learn.

Task 1 This protocol was indeed deployed like this in an initial version of a well-known protocol, and it reflects a classical security flaw that appears repeatedly in protocol design. The public constant `tag` simply acts as an identification of the protocol.

Task 2 This protocol is an abstraction of single-sign-on protocols and a simple security flaw that can occur in them. Try to relate this protocol to a system like MitID: explain how to map participants in a MitID session to those of the protocol and why they can have this initial knowledge but not more.

Exercises for week 5

Task 3 This is an example of a protocol from a major company that initially had a flaw that was discovered with OFMC before deployment and fixed.

Task 4 A classical problem: transmit a payload message securely between two parties who have as their only security relationship a guessable password. In the given task however, the initial knowledge contains more messages than the shared password. What exactly are these, how does that reflect a common real-world scenario? How can you use this initial knowledge to fix the guessing attack?

Claim: without the additional knowledge (i.e., with barely the guessable password), every protocol to securely transmit the payload is vulnerable to guessing attacks. Can you find an argument for or against it?