Lab 03 - Authentication and Key Establishment Lab

Objectives

- Authentication protocols
- Diffie Hellman
- Man in the Middle attacks

Preparation

You may use the UPB's OpenStack cloud to instantiate a Virtual Machine to be used for this lab! Read these instructions if you wanna know how!.

Overview

In the last lecture (Lecture 05 - Authentication and Key Establishment), we studied various authentication protocols and how their behavior and security may be analyzed.

In the current lab, we test a MitM attack on a simple, but broken Diffie-Hellman based protocol.

Tasks

00. Setup

- First, download the lab code from here (inside the VM).
- Again, we use Docker for its remote provisioning features:

```
docker pull ropubisc/auth-lab  # to update image
mkdir ~/auth-lab  # to store your MitM solution persistenly
# you may use the --debug or --mitm argument at the end of the docker command
# when ran with no arguments, it runs a direct Client-Server simulation (no MitM)
docker run --rm --name auth-lab -v $(pwd)/auth-lab/:/home/hacker/auth-lab -it ropubisc/auth-lab
```

- Note: the `~/auth-lab/` folder is used as persistent volume so you won't lose + sync your work inside the container!

01. Man in the Middle

- This one should be clear: code a MitM attack to get the flag (it's only one :D)!
- You must create (hint: start from `server.py`) / modify the `~/auth-lab/mitm.py` file and run it inside the container (with `--mitm` argument for the real case);
  - The middle-man should listen on UDP on port `1337`;
  - You may also use a **debug mode** by supplying the `--debug` as first argument to the Docker image; find the logs inside `/var/log/auth-lab.log`;
- *Start from the sample client & server sources and code your way to it!*
  - ***Hint**: First, you should make sure that the MitM script routes messages correctly!*

02. Bonus: implement authentication

- Start from the client & server samples and implement authentication to both peers (either symmetric or asymmetric – RSA recommended);
- Since you cannot easily modify the container, use your own virtual environment (install `py-diffie-hellman` and `pycryptodome` using `pip`);

Lectures

- Lecture 01 - Introduction
- Lecture 02 - Cryptography
- Lecture 03 - Hardware Security
- Lecture 04 - Access Control
- Lecture 05 - Authentication and Key Establishment
- Lecture 06 - Application Security
- Lecture 07 - Operating System Security
- Lecture 08 - Network Security
- Lecture 09 - Web Security
- Lecture 10 - Privacy Preserving Technologies
- Lecture 11 - Forensics

Labs

- **kernel**
- Lab 01 - Introduction
- Lab 02 - Cryptography
- Lab 03 - Authentication and Key Establishment Lab
- Lab 03 - Hardware Security
- Lab 04 - Access control
- Lab 05 - Authentication in Linux
- Lab 06 - Application Security
- Lab 07 - Operating System Security
- Lab 08 - Network Security
- Lab 09 - Web Security
- Lab 10 - Forensics
- Lab 11 - Privacy Technologies
- Lab 12 - Security and Machine Learning

Support

- Useful resources
- Virtual Machine

Table of Contents

- Lab 03 - Authentication and Key Establishment Lab
- Objectives
- Preparation
- Overview
- Tasks
  - 00. Setup
  - 01. Man in the Middle
  - 02. Bonus: implement authentication