PV204 Security technologies

Hardware Security Modules (HSM), PKCS#

Petr Švenda svenda@fi.muni.cz
Faculty of Informatics, Masaryk University

CS
Centre for Research on Cryptography and Security

Laboratory

- Utilization of HSM capabilities over PKCS#11 interface
  - SoftHSM PKCS#11 token
  - Login user
  - Import keys
  - Use keys
- PKCS#11 usage in other software
  - Using PKCS#11 token as keyfiles storage for TrueCrypt

Order of steps

1. Intro into PKCS#11 API (not covered at lecture)
2. Install and create own virtual SoftHSM token
3. Commented debug throw PKCS11Example code
4. Homework assignment

Prepare SoftHSM (Windows)

- Download binary for your OS
  - https://github.com/disig/SoftHSM2-for-Windows
- Prepare system variables
  - set SOFTHSM2_CONF=h:\Apps\SoftHSM2\etc\softhsm2.conf
- Create and initialize new software token
  - softhsm2-util.exe --init-token --slot 0 --label "My token 1"
- Troubleshooting:
  - Softhsm2-util crash: dll is not available (PATH, try to put softhsm2.dll into current folder)
    - Still crash, check if softhsm2.dll is used (NOT softhsm2-x64.dll)
  - Error: Could not initialize library (check your system variable SOFTHSM2_CONF – name of file should be also included)
    - Check also directories.tokendir inside softhsm2.conf
  - ERROR 30: Could not initialize the token (wrong path to software tokens in softhsm2.conf - check)

Prepare SoftHSM (Linux)

- Use libsofthsm
- http://manpages.ubuntu.com/manpages/utopic/man1/softhsm.1.html

Software token(s)

```
>softhsm2-util.exe --init-token --slot 0 --label "MyToken 1"
*** SO PIN (4-255 characters) ***
Please enter SO PIN: ******
Please reenter SO PIN: ******
*** User PIN (4-255 characters) ***
Please enter user PIN: ****
Please reenter user PIN: ****
The token has been initialized.
```

- New directory (GUID) with software token created in SoftHSM2\var\softhsm2\tokens\ folder
- Multiple tokens can be created
  - Change --slot 0 to --slot X for additional tokens
  - Otherwise token in slot 0 is overwritten

Management of software PKCS#11 token

```
>softhsm2-util.exe
Support tool for PKCS#11
Usage: softhsm2-util [ACTION] [OPTIONS]
Action:
  -h              Shows this help screen.
  --help          Shows this help screen.
  --import <path>  Import a key pair from the given path.
                   The file must be in PKCS#8-format.
                   Use with --file-pin, --slot, --label, --id,
                   --no-public-key, and --pin.
  --init-token     Initialize the token at a given slot.
                   Use with --slot or --free, --label, --so-pin, and --pin.
                   WARNING: Any content in token token will be erased.
  --show-slots     Display all the available slots.
  -v              Show version info.
  --version       Show version info.
```

```
Options:
  --file-pin <PIN>  Supply a PIN if the file is encrypted.
  --force          Used to override a warning.
  --free           Initialize the first free token.
  --id <hex>       Defines the ID of the object. Hexadecimal characters.
                   Use with --force if multiple key pairs may share
                   the same ID.
  --label <text>    Defines the label of the object or the token.
  --module <path>   Use another PKCS#11 library than SoftHSM.
  --no-public-key   Do not import the public key.
  --pin <PIN>       The PIN for the normal user.
  --slot <number>   The slot where the token is located.
  --so-pin <PIN>    The PIN for the Security Officer (SO).
```

AT THIS MOMENT, WE HAVE AT LEAST ONE INITIALIZED TOKEN

Use of PKCS#11 – program API

- Pre-prepared project for Visual Studio
  - PKCS11Example inside 06_SoftHSM
- Example tests of functionality in PKCS11Test
  - List available tokens (slot, token)
  - List of supported cryptographic mechanisms
  - PIN login/change (user CKU_USER, admin CKU_SO)
  - Create and find objects (public, private)
  - Generate random data on token
- Compile, run and inspect in debug mode
- Try to understand what functions are doing

Own work – during this lab

1. Write own function, which will insert private object with label “VeraCrypt secret1” into token
   – Private object => user must be logged in (C_Login)
2. Write own function, which will list all private objects on token including values
   – C_FindObjectsInit, C_FindObjects, C_FindObjectsFinal
3. Change insert function so that value of objects will be randomly data generated by token itself
   – obtained previously via C_GenerateRandom() function

Use of PKCS#11 – TrueCrypt/VeraCrypt

- Use P#11 token to increase security of VeraCrypt password
- Settings→Security tokens→Select library
  - Point to softhsm2-x64.dll
- Important: at least one private object must exists on token
  - VeraCrypt will search for private objects on token and fail with GENERIC_ERROR if not found
  - Use private object “VeraCrypt secret1”
- Volumes→Create new volume
  - (Set standard volume info in wizard)
  - Volume Password→Use keyfiles→Keyfiles →Add token files
  - New volume should be created and PIN required on mount

Homework – RSA with PKCS#11 token

- Create application capable to decrypt with RSA private key stored on PKCS#11 token
  - Private key will stay on a token after application end
- Decryption key (RSA-2048b) is generated on-token
  - C_GenerateKeyPair()
  - Public key is exported into file
  - Private key is usable only after PIN verification (CKU_USER)
- Token will decrypt only after login with user PIN
  - PKCS#1 format for RSA will be used (CKM_RSA_PKCS)
- Use SoftHSM as PKCS#11 token for testing
- Produce short (1xA4) text description of solution
  - Steps and principal usage difference to Signature applet from HW02

Homework – RSA with PKCS#11 token

- Provide code that will demonstrate:
  - RSA keypair generation
  - Search for object with private key and successful decryption of data
  - Failure of decryption when user PIN is not supplied
  - Destruction of keypair object on token
- You may use existing code as inspiration, but **you can't cut&paste**!
  - https://www.ibm.com/support/knowledgecenter/SSLTBW_2.1.0/com.ibm.zos.v2r1.csfba00/testpkcs11_code.htm%23testpkcs11_code
  - Be aware – this code doesn't search for key objects
- Submit before: **8.3. 6am** (full number of points)
  - Every additional started day (24h) means 1.5 points penalization