Introduction to Computer Security: Exercise 1

Deadline: 22 February, 12:00noon

1 Introduction

The exercises for this module will be based on analysing the security of a virtual machine. For this exercise you must download the VM, get it working and perform a number of cryptographic operations.

1.1 Getting the VM working

You first need to download the VM image and install it, as per canvas page.

There are many user accounts on the VM; you begin the module with access to just one:

Username: `employee427`, password: `employee427`

You must use your own copy of the VM for this (and every other) exercise. You must not share your VM with other students.

In the home directory you will find a token split into two files `theFirstTokenPart1`. and `theFirstTokenPart2`. The token is the concatention of these two files. Submit this token on the website:

`https://www.cs.bham.ac.uk/internal/courses/comp-sec/token`

[1 marks]

1.2 Encryption in Java

1.2.1 Introduction

The previous employee in your post started writing an encryption program, but they didn't write the decryption part yet. For this part of the exercise you need to complete the program so that it also decrypts files. In you home directory will find the files `ex1CTR.enc`, `ex1CCM.enc` and `ex1RSA.enc`, these are files that have been encrypted with AES in CTR mode, AES in CCM mode and RSA. You will also find the Java file `EncTool.java` which is a command line tool that can be used to encrypt files.

This tool uses the "Bouncy Castle" crypto libraries. These can be found in the bcprov-jdk15on-151 jar file in your home directory, and Java needs to be pointed to this. So you can compile the code using:

```
javac -cp bcprov-jdk15on-151.jar EncTool.java
```

and to run it using:

```
java -cp bcprov-jdk15on-151.jar:. EncTool <mode>
          <key:128 bits in as hex>  <inputFile> <outputFile>
```

E.g. the `ex1CTR.enc` file was created using the command:

```
java EncTool -encAESCTR 4B6150645367566B5970337336763979
                              plainText.txt ex1CTR.enc
```

and the `ex1CCM.enc` file was created using the command:

```
java EncTool -encAESCCM 4B6150645367566B5970337336763979
                              plainText.txt ex1CCM.enc
```

RSA encryption must also includes the name of the keyStore:

```
java EncTool -encRSA <keyStore> <keyName> <inputFile> <outputFile>
```

where `keyStore` is the file name of a key store and `keyName` is the name of the RSA key within the store. The tool will ask you for the password for the keystore which is "password".

The file `myKeyStore` contains an RSA key pair called `mykey` and is protected with the password: `password`. The `Ex1RSAencrypted` file was created with the command:

```
java EncTool -encRSA myKeyStore mykey plainText.txt ex1RSA.enc
```

1.2.2 Exercises

1. Complete the method `decryptAESCTR()`, so that the program can decrypt messages encrypted with AES in CTR mode. Decrypt the `ex1CTR.enc` file (with the key given above) and submit the token it contains to the website:

   ```
   https://www.cs.bham.ac.uk/internal/courses/comp-sec/token
   ```

   [1 marks]

2. As discussed in lectures, CTR mode encryption does not authenticate the encrypted data, therefore it can be altered by the attacker. The plaintext of the message encrypted in the file `ctr.enc` is `Pay Tom 1000 pounds`, you do not have access to the encryption key. Change the ciphertext so that the decrypted message would read `Pay Bob 9999 pounds`. I strongly suggest that you make your own cipher texts with EncTool and experiment with them, and using the hex editor ghex, which is installed on the VM. Submit your edited `ctr.enc` file to Canvas.

[3 marks]

3. Complete the method `decryptAESCCM()`, so that the program can decrypt message encrypted with AES in CCM mode. Decrypt the `ex1CCM.enc` file (with the key given above) and submit the token it contains to the website. (You might also like to try editing some cipher texts and see that CCM mode detects the changes).

   [1 marks]

4. Complete the method `decryptRSA()`, so that the can decrypt message encrypted with RSA mode. Decrypt the `ex1RSA.enc` file (with the private key in the keystore) and submit the token it contains to the website. N.B. the RSA key used (and therefore the size of the encrypted AES key) is 2048 bits (= 256 bytes) long.

   [2 marks]

1.3 Getting help:

There is documentation for all of the tools on their respective websites and many "how to" guides can be found via Google. You can come to the lab sessions or to my office hours with questions and to get extra help with the exercise. You can also use the discussions on Teams.