School of Computing and Information Systems
COMP30023: Computer Systems
Assignment 2
Due date: No later than 11:59pm on Thursday 24th May 2018
Weight: 15%

Project Overview

The aim of this project is to increase your awareness and familiarity with TLS Certificate checking. Whilst many applications will perform the certificate checking for you, it is important to understand the underlying process in order to be able to evaluate security. Your task is to write a program that validates TLS certificate files.

Your program must be written in C. Submissions that do not compile and run on a NeCTAR instance may receive zero marks. You must write your own certificate validation code using the OpenSSL library.

Project Details

Your task is to write a C program that reads in a CSV (comma separated value) file that contains two columns. The first column provides the file path for the certificate to test. The second column provides the URL from which that certificate belongs. Your program must step through each line in the CSV file, load the certificate specified in column one, and validate it, including checking the URL contained in column two.

The output of your program should be another CSV file, called output.csv. It must contain one line per certificate checked, in the same order as the input CSV file. Each line should contain three columns, the two columns from the input file, and a third column containing either the value 1 if the certificate is valid, or 0 if the certificate is invalid.

An example input file is shown below:

```
cert_one.cer,www.comp30023test.com
cert_two.cert,game1.onlinegaming.com
```

An example output file is shown below:

```
cert_one.cer,www.comp30023test.com,1
cert_two.cert,game1.onlinegaming.com,0
```

The minimum checking you are expected to do is as follows:

1. validation of dates, both the *Not Before* and *Not After* dates
2. domain name validation (including Subject Alternative Name (SAN) extension) and wildcards
3. minimum key length of 2048 bits for RSA
4. correct key usage, including extensions

Your program must compile to an executable called `certcheck` and must take one command line argument, which is the relative path to the input CSV file. For example, the following should run your program and check the certificates listed in mytestfile.csv `./certcheck mytestfile.csv`

You can assume that there are no restrictions on Subject Alternative Name's beyond the specification, and in particular that wildcard domains are allowed in both the Common Name and the SAN. Your checking code should handle such wildcards correctly. You can assume that all certificates will use RSA keys.

Program execution / command line arguments

To run your program on your NeCTAR cloud instance prompt:

```
./certcheck pathToTestFile
```

where:

- `pathToTestFile` is a valid relative path to the test CSV file.

Note: Program interface will be strictly enforced, if your program does not comply with the required standard you may get zero marks for part B and C.

Submission details

Please include your name and login id in a comment at the top of each file. Our plan is to directly harvest your submissions on the due date from your provided GitLab (**https://gitlab.eng.unimelb.edu.au**) git account. You must create a repository called `comp30023-2018-project-2` in your GitLab account and push your submission to it.

You must submit program file(s), including a Makefile. Make sure that your Makefile, header files and source files are added/committed and pushed to your GitLab repository. Do not add/commit object files or executables. Anything you want to mention about your submission, write a text file called README.

- If you do not use your Git repository for the project you will NOT have a submission and will be awarded zero marks.
- It should be possible to "checkout" the Git repository, then type `make clean` and then `make` to produce the executable `certcheck`.
- Late submissions will incur a deduction of 2 marks per day (or part thereof).
- If you submit late, you MUST email the lecturer, Chris Culnane cculnane@unimelb.edu.au.

Extension policy: If you believe you have a valid reason to require an extension you must contact the lecturer, Chris Culnane cculnane@unimelb.edu.au at the earliest opportunity, which in most instances should be well before the submission deadline.

Requests for extensions are not automatic and are considered on a case by case basis. You will be required to supply supporting evidence such as a medical certificate. In addition, your git log file should illustrate the progress made on the project up to the date of your request.

Plagiarism policy: You are reminded that all submitted project work in this subject is to be your own individual work. Automated similarity checking software will be used to compare submissions against each other and known public source code. It is University policy that cheating by students in any form is not permitted, and that work submitted for assessment purposes must be the independent work of the student concerned.

Using Git is an important step in the verification of authorship. You are encouraged to commit regularly so that you have a record of your work. This is also best practice when using version control software.

Assessment

Code that does not compile and run on a NeCTAR instance will be awarded zero marks. Your submission will be tested and marked with the following criteria:

- Part A Compiles from Git (5 marks)
  - Code successfully added to your git repository on **https://gitlab.eng.unimelb.edu.au** (1 mark)
  - Make file included (and it works) (1 mark)
  - Clarity and quality of code – appropriate comments and documentation where necessary (2 marks)
  - Code correctly implements command line interface (1 mark)
- Part B Basic Certificate Checking (5 marks)
  - Reads input CSV and write output CSV (1 mark)
  - Correctly validates *Not Before* date (1 mark)
  - Correctly validates *Not After* date (1 mark)
  - Correctly validates domain name in Common Name (2 mark)
- Part C Advanced Certificate Checking (5 marks)
  - Correctly validates minimum RSA key length of 2048 bits (1 mark)
  - Correctly validates key usage and constraints (2 mark)
    - BasicConstraints includes "CA:FALSE"
    - Enhanced Key Usage includes "TLS Web Server Authentication"
  - Correctly validates Subject Alternative Name extension (2 marks)

Example Code and Development

Example source code that opens an X509 certificate and reads some values from it, is available in the Assignment git repository:

`https://gitlab.eng.unimelb.edu.au/COMP30023/Assignment2.git`

This will show the basics of opening and reading values from an X509 certificate file using OpenSSL in C. However, this will not provide all the functionality that you require. Part of the exercise is to navigate through the OpenSSL documentation and discover the appropriate methods to call to extract and evaluate the data you need. Please Note: You must write your own validation code, you cannot use the built in OpenSSL certificate validation functions.

The documentation is available at:

`https://www.openssl.org/docs/manmaster/man3/`

A list of function calls you may not use is as follows:

- X509_check_ca
- X509_check_host
- X509_cmp_current_time
- X509_cmp_time

Testing your program

Your server will be partially evaluated using automated testing. A sample test script and sample certificates will be provided in the repository:

`https://gitlab.eng.unimelb.edu.au/COMP30023/Assignment2.git`

You may use this script to test your program successfully passes the following assessment criteria:

- Basic Certificate Checking (5 marks)