Secure Programming Coursework (Part 2)

Arthur Chan and David Aspinall

School of Informatics, University of Edinburgh

This is an **individual** assessed practical exercise. It is the only assessed coursework for the Secure Programming course. It consists of two parts, this is the second part. Provided you have attended the relevant lectures and lab sessions in the course, the work for both parts should take about 30 hours. The practical will be awarded a mark out of 100. The single deadline for submission (both parts) is **5pm, Fri 15th November 2019**. The final page summarises the submission instructions.

Virtual machinery

We provide a **different** virtual machine for you to use in Part 2. The VM has two users, `user` and `root`. The **passwords are the same as their user names**.

To install the VM, you should use a virtual disk file stored on a local disk on your machine. If you are working in the Appleton Tower Lab on the DICE machines, you can use the directory `/tmp/sNNNNNNN` if there is enough space. Configure VirtualBox to use the right disk area by setting `File → Preferences → General → Default Machine Folder`. Next, **import the appliance** (`File → Import Appliance`) from the file:

```
/afs/inf.ed.ac.uk/group/teaching/module-sp/SecureProgramming-Coursework-Part2.ova
```

If you are working remotely or on your own machine, we recommend taking a copy of the `.ova` file first rather than importing over directly from AFS. The file is about 1G. It may be more convenient to use a USB stick than download it over the Internet.

Important: make sure that your VM disk files are stored in a directory which is only readable by you. Beware that `/tmp` are disk areas which are **not backed up and specific to the workstation you are using**. So if you are using a lab machine (and anyway, for safety), **back up your work** by saving any work that you do inside the virtual machine (edited source files, etc) in your home directory.

Using the machine

You should complete all questions as the unprivileged user called `user`. The user has sudo rights to execute some commands (restarting of services).

The machine is set to use NAT. Once started you can either use the console window, or (recommended): SSH in via your local machine over port 2222, with: `ssh -p 2222 user@localhost`.

We've supplied some tools to make things easier but feel free to install additional software in the VM. In your **answers3.pdf**, please describe **all tools you have used**, including Linux packages, browser plugins used in your host machine, etc.

3. Secure Server Management (50 marks)

The VM is set to use NAT. We have already set four port forwarding rules in the VM. If you think it is necessary, you can add more port forwarding rules (these are set in the Virtual Box machine settings in the Network section). If you do so, please include all the new or modified port forwarding rules in your **answers3.pdf**. The existing port forwarding rules are as follows:

For SSH Virtual Machine port 22 is forward to host machine port 2222. (TCP)

For HTTP Web Server Virtual Machine port 80 is forward to host machine port 8080. (TCP)

For HTTPS Web Server Virtual Machine port 443 is forward to host machine port 44443. (TCP)

For OpenSSL Virtual Machine port 12345 is forward to host machine port 54321. (TCP)

All the answers and requested comments should be written in **answers3.pdf**, while there are additional files required for each question below which you should submit separately. The full list of submission files for this part of the coursework is summarized at the final page.

3.1 OpenSSH Configuration (5 marks)

The server has the SSH service turned on. It allows remote users to connect to the server through port 22 (which is forwarded to port 2222 on your host machine). Each student is given the same VM with the same set of login information. Using a default password is not a secure way to manage your own web server, others can possibly break into your VM and ruin your work. So the first thing you need to do is to secure the ssh connection with some configuration. This should prevent others using a default credential log into your VM. You must meet the following requirements.

a) Only allow `user` to login through ssh. Deny all other users (including root) to login through ssh service. (2 marks)

b) Disable password authentication and only allow `user` to login with a private key. [Hint: You may need to generate key pairs with *ssh-keygen* command which exists in almost all linux environment] (3 marks)

You will need to restart the ssh server to apply some of the changes. The command is

/usr/bin/systemctl restart sshd

You will need to use the root account to restart the ssh server. Please read the man page of sshd by **man sshd** for more details.

For this question, submit a tarball **sshconfig.tar.gz** which includes all the config files you have added or modified. Then provide a brief description in **answers3.pdf** to describe the use of each of your modification and how it links to the requirements. If you think the config files need to be stored in a specific path to make something work, please also mention that in your description. For clarity, please use absolute paths of filenames in your description (starting from root /).

3.2 Fun with Heartbleed (10 marks)

Heartbleed is a serious bug in OpenSSL which was discovered in 2014. Now you will have the chance to fix Heartbleed in this virtual machine. There are three folders in `/home/user/` for this question.

/home/user/key Holds the key and certificate that you need to start up the openssl service.

/home/user/openssl Holds the openssl package which is compiled from the source version 1.0.1f.

/home/user/openssl-1.0.1f-source Holds the source code for openssl version 1.0.1f.

The server has been configured with the correct `PATH` setting. Whenever you execute the `openssl` command, it will execute `/home/user/openssl/bin/openssl` by default.

If you need to regenerate the key, you may need to use this command.

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365 -nodes
```

If you need to recompile the source (after you fix something), please go into the source code folder and execute this command.

```
./config --prefix=/home/user/openssl enable-tlsext no-shared && make && make install_sw
```

The above command will compile the source code and overwrite the package in /home/user/openssl automatically.

If you want to start the OpenSSL service, please execute this command.

```
openssl s_server -key /home/user/key/key.pem -cert /home/user/key/cert.pem -accept 12345 -www
```

The above command will start a web OpenSSL service on port 12345, which will forward to port 54321 on your host machine.

a) In **answers3.pdf**, briefly explain how an attacker can exploit the Heartbleed bug. What are the possible consequence of such an attack? (2 marks)

b) Try to review the code and fix the Heartbleed bug. Briefly describe your fix in **answers3.pdf**, then submit a separate patch file **question3.diff** that shows your change in the code. Your patch file should only fix Heartbleed bugs. Zero marks will be awarded for this question if your patch 1) alters the functionality, removes or stop the hearbeat features, or 2) affects or fixes other part of the code which is not related to the Heartbleed bug. (8 marks)

Hint 1. Version 1.0.1f is the last OpenSSL version affected by Heartbleed bug, the patch update for subsequent versions may help you to discover how to fix the Heartbleed bug.

Hint 2. There are many Heartbleed testing tools around the Internet for testing your patch. (Don't execute those tools on a real server unless you have the owner's permission!)

Hint 3. Heartbleed is a bug in the stage of processing a "heartbeat" (hence its name); try looking in the **ssl** folder in the source code to find the problem.

Even if you cannot fix the bug, if your description is partially correct you will get partial marks.

For this question, submit a patch file **question3.diff** to demonstrate your fix to the heartbleed bug. We will apply your patch and test if it really prevents Heartbleed.

3.3 Digital Signature Service (35 marks)

Suppose that the main purpose of the VM is running a digital signature service. Your task now is to build up a web based application for digital signature generation and validation.. Passwords and keys are highly sensitive data and so you must focus on security in your implementation. Apply as many countermeasures as you deem sensible when you are developing your app, but you must start from the basic framework that we provide. Do *not* switch to a different language, database or application framework (even if you think that might be a better route to security).

The VM is provided with a httpd server already running on port 80 (which is forwarded to port 8080 on the host machine). When you open the page in a browser, you should see a login page for the digital signature service which does nothing. This is just a sample page for you to begin with. All the code for your web application should be stored under **/srv/http/**. You should not assume any of the template code given to be sufficiently secure.

In the virtual machine, we provide sqlite3 as a database system you should use. A sample database file is already created for you in the **/srv/http/db/** folder.

To interact with the database, you can use **sqlite3 ds_service.db**. This runs the SQL command interpreter directly. As well as traditional SQL commands, sqlite3 provides several control commands. Some more common ones are:

.help Print all control command for sqlite3

.tables Show all the tables in the database

.quit Exit sqlite3

Besides the sqlite3 database, some PHP functions are provided for you in **/srv/http/include/functions.php**. They include basic user registration, login and database connection code. You can use this directly, or modify it as you like.

Your web application must have the following features:

a) Signup and login (3 marks)

Provide a way for new users to register with the service. Generate a new private key / public key pairs for this user and store them in the system. It is up to to design a suitable way to store them and deny user access to any private keys stored in the system. The key type should be 2048 bits with RSA After users log in, they should be redirected to a dashboard with three functions which you will implement in question (b), (c) and (d).

b) Export Public Key (2 marks)

Provide an interface to allow the user export their own public key in pem format to files. Never allow the users to export or access their private key.

c) Digital Signing (3 marks)

Allow the user to input some text message. The text message should not have more than 10000 characters. Generate a digital signature with the user's private key and display the resulting signature on screen.

d) Digital Signature Verification (3 marks)

Allow the user to input some text message. The text message should not have more than 10000 characters. The user should also provide a public key file in pem format and a digital signature signed for the inputted message by the owner of the provided public key file. The system should then verify the digital signature and display the result on screen.

e) Security evaluation and countermeasures (24 marks)

Perform a security audit of your web application and consider possible attacks; check that your code is secure and implement appropriate additional mitigations. You should consider the application, the web server configuration and its use of the operating system, and the interaction with the client browser. Do not add auxiliary general countermeasures (additional programs such as intrusion detection, etc) or make further alterations to the operating system, even if you think these might be worthwhile.

To help, you might like to use a list of common web application flaws, or conduct a simple `STRIDE` style threat-modelling exercise.

Provide **at least six** separate security mitigations and describe them briefly but clearly in **answers3.pdf** along with a pointer to the code or change that addresses the issue. The description should include but not limited to the vulnerabilities, potential attacks and how your mitigations work. You may also include fixes to our provided code.

Remark: You may need to restart the http server to apply some changes. The command is

/usr/bin/systemctl restart httpd

Please read the man page of httpd, **man httpd** to understand the server better.

For this question, marks will be awarded for (a)–(d) if you implement the features correctly. Then, all your security consideration and measures will be evaluated, inlcuding those applying to Part (a) to Part (d). You should consider that this web application will be published on the Internet using a cloud hosting platform. Thus, all users (including potential attackers) will be able to abuse your application in any way they want.

Since we are only concerned with secure programming, please do not put effort into interface design (although there can be security-motivated reasons to do so, we will not evaluate them here). Marks will be awarded on required features and a careful security analysis implementing sensible mitigations.

You should submit a tarball **code.tar.gz** which contains all the code for your web application. If you alter the configuration of the web server or other components, you must also include those files in the tarball. In your **answers3.pdf**, briefly describe how you fulfil the requirements, as well as the security measures addressed in part (e). For any config file in the tarball, please give its absolute storage location (starting from root /) for clarity. Or store the files will full paths in the tarball.

Submission instructions (Part 2)

You should submit your answers electronically with the command:

`submit sp cw` *filename*

or if you want to submit multiple files:

`submit sp cw` *filename filename* ...

Where *filename* is:

answers3.pdf A PDF document containing your description to question 3.

sshconfig.tar.gz A tar gz file containing all your modified config for *Question 3.1*.

question3.diff The patch file generated for *Question 3.2*.

code.tar.gz A tar gz file containing all your code for question 3.3.

You can create the tar gz file by

tar -czf code.tar.gz BASE_DIRECTORY_OF_ALL_YOUR_CODE

You can create the diff patch file by

diff -c oldfile newfile > question3.diff

Wrong *filename* arguments will not be accepted. The PDF documents should be well-formatted printable A4 PDFs, you may generate them with whatever program you want. Text answers should be brief and to-the-point.

Repeated submission of the same *filename* will overwrite the previous submission. Take care: the submission does not keep a history of submitted files, we will mark the most recent files and their submission timestamps must be before the deadline to avoid standard lateness penalties.

The coursework is separated into two parts and released in stages, but both parts have the same final deadline.

You must submit both parts of the coursework by the **final deadline 5pm, Friday 15th November 2019**.

You're reminded that late coursework is not allowed without "good reason", see the Informatics policy on coursework deadlines[1] for details, and the procedure to follow if you must submit late. In particular, if you have a good reason to submit late, use the Student Support Team contact button to make a request rather than asking us.

[1] http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests