

Secure Programming Coursework

Arthur Chan, Joseph Hallett 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 **is issued in good time** so you can start the work and see what is required, but some parts will be easier after lectures and lab exercises yet to come. Provided you have attended the relevant lectures and lab sessions in the course, the work here should take at 30 hours, including time for needed reading. The practical will be awarded a mark out of 60. The deadline for submission is **4pm, 14th March 2016**. The final page summarises the submission instructions.

Virtual machinery

We provide a virtual machine for you to use. The VM has two users, **user** and **root**. The passwords are the same as their usernames.

To install the VM, you should use a virtual disk file stored on a local disk on your machine. For example, working in the Forest Hill 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 from the file:

```
/afs/inf.ed.ac.uk/group/teaching/module-sp/SecureProgramming-Coursework.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 1.2G. 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. 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 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`.

Additionally the VM runs a web server on **port 80**. This is forwarded to **port 8080** on the host machine.

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

1. Bugs in OpenSSH (20 marks)

In January 2016 two bugs were disclosed in OpenSSH: *CVE-2016-0777* and *CVE-2016-0778*.

1. In your own words briefly describe the two bugs. State what versions of OpenSSH they affect, and what remedial action ought to be taken. (5 marks)

CVSS scores help categorize bugs and provide information about the bugs seriousness and attack information.

- CVE-2016-0777 was given a CVSS v3 score of 6.5 AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N.
- CVE-2016-0778 was given a CVSS v3 score of 9.8 AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H.

2. Describe what each of the CVSS scores mean, including the meanings of the string components. How severe is each CVE? (5 marks)
3. With reference to the BSIMM6 report describe three security activities and how they might have discovered these bugs. (5 marks)


Another SSH server has only had 7 CVEs reported against it, and none since 2013. OpenSSH has had 80 CVEs reported, three of which this year.

A friend has suggested that you switch from using OpenSSH to this new server as, based on the CVEs, OpenSSH seems to be more buggy. They remind you of the *Heartbleed* bug from 2014, and tell you that they don't trust OpenSSH anymore.

4. Discuss your friend's opinions and the relative merits of switching your SSH server from OpenSSH to the new system. (5 marks)

2. Buffer overflow (20 marks)

All code for this question is in the `/home/user/exploit` folder on the virtual machine. You have been given a program called `vulnerable` which has a buffer overflow vulnerability. Your task is to exploit the program so that the message **Correct Password!** is printed.

1. Describe how the program can be exploited, giving the steps required to make it print the successful login message. Describe each step using bullet points. There should be roughly four steps to complete the exploit. (10 marks) 

2. Create an exploit script called `exploit` that takes the path of the program as its first argument and completes the challenge. We will run **your** program by executing:

```
./exploit ./vulnerable
```

It must not output anything other than the output produced by the vulnerable program. You may use any scripting language to write your exploit, provided it runs as described. (5 marks)

3. Provide a patch file that fixes the overflow vulnerability. (5 marks)

Note: a patch file can be created with the command

```
diff -c <oldfile> <newfile> > question2.diff
```

Keep a copy of the original file so you can make the patch file!



3. Web Security (20 marks)

Inside the virtual machine there is a very naive web app for image sharing and voting. You may access the web-app through `http://localhost:8080`. The server is poorly configured and is susceptible to a number of vulnerabilities. There are 10 users created for you in advance. They are `user1 ~ user10`. The passwords are the same as respective usernames.

NOTE: because the steps below will corrupt the web application on the virtual machine, we have provided a reset script to restore the original state of the database of the web app. You can run this by executing the command `/home/user/resetdb.sh`. In case you have managed to corrupt even more than the database, you will need to re-import the virtual machine (or make a VirtualBox snapshot to restore from).

1. Describe how the app is vulnerable to an XSS based attack. Your description should include code to demonstrate the vulnerability (e.g., by injecting the javascript `alert("Hello World!")`) as well as *step by step* instructions. (4 marks)
2. Describe how the app is also vulnerable to an CSRF attack. The attack should cause a victim vote for the attacker (another user) transparently. Describe the steps to perform the attack from the perspective of both the attacker and victim. Again, give your answer as a clear and unambiguous series of steps detailing the necessary inputs. (4 marks)

3. The web app can be found in `/srv/http`. Perform a code review and security audit, and report up to four potential vulnerabilities.

For each vulnerability, describe what the source of the vulnerability is, how it might be exploited and, by giving a *brief* correction to the code, how it may be fixed.

You should provide you description and answers in **answers.pdf** and submit the overall corrected version of the code in a single patch file called **question3.diff**. Remember to **backup** the files in `/srv/http` before modifying them. You will need the original code to run the *diff* command.

You may ignore the code in the `css/`, `js/` and `fonts/` folders. (Hint: configuration issues count too!). (12 marks)

You should understand the web application in the first two questions by running it, and in the last stage by inspecting the code. As well as manual code review, security testing tools can be very useful. Some server-side analysing and auditing tools include:

- **RIPS**, see <http://rips-scanner.sourceforge.net/>
- **WAP**, see <http://awap.sourceforge.net/> and OWASP's page https://www.owasp.org/index.php/OWASP_WAP-Web_Application_Protection.

You can also try client-side debugging or pen-testing tools, for example

- **HackBar**, a plugin that works in Firefox or Chrome. In Firefox, search in `about:plugins` to add it to your browser.

If you use these kind of tools, we recommend using a fresh browser profile to install them, and **do not execute on real websites** unless you have the express permission of the website owners to do such testing.

Although security auditing tools can help to identify web application vulnerabilities, they are not necessarily comprehensive so it is unwise to rely solely on them. The security of web application still requires careful design and secure coding practice.

Submission instructions

You should submit your answers electronically with the command:

```
submit sp cw filename
```

Where *filename* is:

answers.pdf A PDF document containing the answers to each question *in order*.

exploit The script required to exploit the program for *Question 2.2*.

question2.diff The patch file generated for *Question 2.3*.

question3.diff The patch file generated for *Question 3.3*.

Wrong *filename* will not be accepted.

Please submit by the deadline of **4pm, 14th March 2016**.

You're reminded that late coursework is not allowed without "good reason", see the fourth year honours course guide for details.

For more details about this, and the procedure to follow if you must submit late. In particular, if you have a good reason to submit late, use the ITO support form to make a request rather than asking us.