Hands-on Security Testing in a University Lab Environment

Phillip James
Lauren Powell
Liam O'Reilly
Faron Moller
p.d.james@swansea.ac.uk
lauren.powell@technocamps.com
l.p.oreilley@swansea.ac.uk
f.g.moller@swansea.ac.uk
Swansea University

ABSTRACT

In this paper, we present a novel approach to providing students with hands-on learning experiences for security testing. Specifically, we give details of a Raspberry Pi-based lab architecture which we developed as part of a security course, and provide example lab tasks that such an architecture affords. We provide evidence that introducing such hands-on labs within the security course directly improved student performance. Finally, we compare our approach to other existing approaches of which we are aware that are based on virtual machines and networks.

KEYWORDS

Computing education, Computer security, Computer architecture

ACM Reference Format:
Phillip James, Lauren Powell, Liam O'Reilly, and Faron Moller. 2020. Hands-on Security Testing in a University Lab Environment. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387366

1 INTRODUCTION

Traditional approaches to the delivery of computer security courses within undergraduate computer science and software engineering programmes typically consist of lectures followed by hand-written theoretical exercises or simple proof-of-concept programming exercises. The reason for this is essentially two-fold. Firstly, security is just one of many topics covered in such programmes, so only a limited amount of time and pre-requisite knowledge can be assumed. Secondly, using networked computers to carry out practical lab-based exercises which explore, e.g., session management, access rights, and code injection attacks will quickly breach campus-based security policies; and as we will note below, the expense of a bespoke security lab can easily be prohibitive, particularly when it is for a single course within a wider programme of study.

ITiCSE '20, June 15–19, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-6874-2/20/06…$15.00
https://doi.org/10.1145/3341525.3387366

However, computer security is of ever-increasing importance as a topic in any digital-based programme of study. In particular, coverage of security increases with each edition of the *ACM Computing Curricula*[1] and within the *BCS accreditation benchmarks*.[2] Along with data science, cyber security education is intensely promoted – particularly within the scope of degree apprenticeships – by the UK-wide *Institute of Coding*[3] (IoC). It has been highlighted by various sources [12, 14] that the traditional light-touch approaches to delivery do not provide the hands-on experience that is required by a typical security analyst role in a business setting.

At Swansea University, we are acutely aware of these issues as Lead of the *IoC in Wales* and through our *BSc Degree Apprenticeship in Software Engineering* programme on which some 100 employees of regional companies are enrolled. This programme is underpinned and monitored by the IoC in Wales *Industrial Advisory Panel* which provides ample motivation for finding ways to overcome the above barriers to delivering practical lab-based cyber security tasks.

In this paper we detail a novel method of delivery that provides a hands-on security testing experience using Raspberry Pi's. These micro PC systems provide a portable and easy to set up networked lab environment that is independent from university networks and systems. Weekly tasks can be easily distributed via SD cards that contain customised operating systems and vulnerable software, allowing students to experience a wide range of security vulnerabilities and gain practical experience in detecting and patching such vulnerabilities. We provide concrete examples of lab tasks and illustrate – through the experience of delivery – that such labs can be effective in improving student appreciation and understanding of security vulnerabilities. We also argue that the relative cheapness and ease of maintenance of the approach make it a much more attractive proposition than related approaches based on virtual machines and virtual networks, where the cost of implementation and maintenance overheads can be substantial for labs that involve large numbers of students. Finally, this work forms part of a greater community effort to develop and support delivery of hands-on teaching experiences for students, and we suggest a community-driven effort to share resources such as operating system images and associated lab setup information. We frame the paper as an experience report that can be utilised by others wishing to adopt such an approach.

[1] www.acm.org/education/curricula-recommendations
[2] www.bcs.org/deliver-and-teach-qualifications/university-accreditation
[3] instituteofcoding.org

The remainder of the paper is structured as follows. In Section 2 we describe the course that incorporates hands-on lab exercises. Then in Section 3 we present the lab architecture that has been deployed, before explaining in Section 4 the types of lab tasks that the architecture supports. (We include some example indicative lab tasks as appendices.) In Section 5 we evidence the positive student feedback created by the approach, which demonstrates an overall improved student experience. Finally, in Section 6 we compare how the approach differs to the use of virtual machines and networks, before providing concluding remarks in Section 7.

2 COURSE OVERVIEW

The lab structure and content described in this paper arose through efforts to develop hands-on content for an existing *Cryptography and IT Security* course which is delivered for both final-year undergraduates and for masters students. The aim of this course is to examine theoretical and practical aspects of computer and network security. The course is fairly traditional in nature and covers a wide range of topics including:

- Cryptography (e.g., encryption, DES, AES, hash functions).
- Security threats and their causes.
- Security criteria and models.
- Access control mechanism.
- Penetration testing.
- Vulnerabilities and attacks (e.g., port scanning, SQL injection).
- Security tools and frameworks (e.g., IPSec, TLS, SSH).
- Blockchain technology and Anonymity networks.

It is expected that students leave this course able to identify security threats and their causes in modern computing infrastructures, and with a critical awareness of the limits of these techniques. They should gain experience building secure systems and be able to enhance, where possible, the security of existing systems.

Up until 2017, this course was delivered as a traditional university course with two hour-long lectures per week and a one hour lab. For the 2017/2018 academic year, based on feedback from the IoC in Wales Industrial Advisory Panel reflecting on our BSc Degree Apprenticeship programme, significant effort was put into a full redesign of the labs to ensure that students were gaining the hands-on skills required to enter the security sector. This included introducing new lab equipment and new lab content. Due to the success of this revamped course, we introduced a masters-level course exploring the practical skills of penetration testing, where the lab component is double-weighted in terms of delivery time to lectures.

3 LAB ARCHITECTURE

The key motivation in setting up our new lab was to improve students' understanding of the course content by providing a hands-on practical experience. There were, however, a number of constraints placed on any possible solution:

(1) The budget for investing in the course was limited (c. £5000).
(2) It was clear that using existing university systems and networks would not be possible due to stringent (albeit understandable) restrictive usage policies.
(3) The lab should support students working offsite.
(4) The solution should support class sizes up to 250 students in order to "future-proof" the course in terms of expected growth.

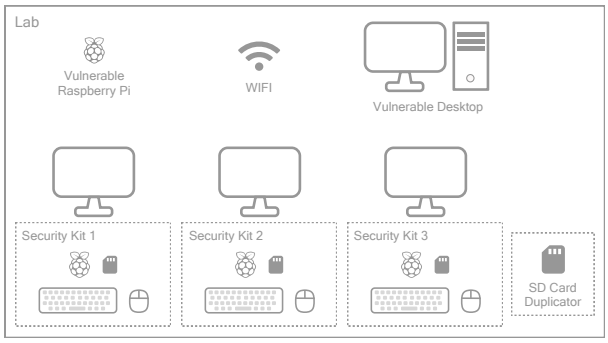

Figure 1: The new security testing lab setup.

These constraints immediately ruled out the use of a virtualised lab such as those explored in [1, 2, 6, 8, 15], due to the cost of such a solution in terms of the infrastructure needed to support 250 virtual machines. Equally, approaches that rely on distributing virtual machine images to students, e.g. [3–5], posed technical problems: these virtual machines would either need to be run on university machines, breaking current policies; or be installed by students onto their own machines, which would breach university expectations on students (they are not required to provide their own computing devices, nor install specific software on their personal computers) as well as require substantial bandwidth to distribute several hundred instances of a virtual machine image for each lab, with virtual machine images easily reaching several gigabytes.

To overcome these challenges, we created a lab architecture (see Fig. 1) that provides a portable kit that students can use within the university lab and at home (e.g. for the individual non-networked lab tasks presented in Section 4). Each kit consists of:

- A Raspberry Pi: a low-cost, reasonably-powerful computer.
- A mini keyboard, mouse, and HDMI cable.
- An SD card containing an operating system. The operating system provided varies depending on the lab task being explored. Raspberry Pi's typically run a lightweight Linux distribution known as Rasbpian, though work with other operating systems including Kali Linux and Ubuntu Server.

The overall cost per kit is approximately £50. In addition to this, several WiFi routers were purchased (one per 30 Raspberry Pi's) along with two SD cards per Raspberry Pi. An SD card duplicator supporting the re-writing of 32 SD cards at a time was also purchased. This infrastructure allows for the following use case:

> *At the start of the course, each student*[4] *collects a kit that they use for the duration of the course. At the start of each lab, the student collects that week's lab task and associated SD card from the lecturer, returning the card from the previous week's lab class. They set up their Raspberry Pi, using existing lab monitors, and connect their Raspberry Pi to the dedicated ad-hoc security lab WiFi network. They then complete the lab, taking the kit home with them if additional study time is required.*

In order to prepare a lab, the lecturer is required to develop an SD card image for each student. This process is fairly simple and involves installing software packages onto the base operating

[4]For the course considered here, students worked in pairs to complete lab tasks.

system that has been chosen for that week's lab, before duplicating the SD cards. In addition to this, the lab may also require additional vulnerable machines (either Raspberry Pi's or standard desktops) to be deployed on the network. Should a student accidentally break fundamental components of their lab machine whilst performing (sometimes risky) tasks, then they can simply re-flash their SD card with the image for that lab.

This approach successfully scales to large cohort numbers: in the second instantiation of the course, we deployed over 100 Raspberry Pi kits at the same time, supporting over 200 students working in pairs. We supported the lab with one teaching assistant per 40 students (i.e. 20 pairs). This required minimal overhead in terms of growth from the initial run of the course (roughly 100 students); the only overhead was the time needed to duplicate more SD cards. However we do note that for network-heavy tasks such as network scanning, additional wireless routers are required to support the communication overhead produced by the additional number of devices connected to the network.

4 TYPICAL LAB TASKS

To illustrate the experiences that our lab architecture affords, we describe labs that have been delivered as part of a *Cryptography and IT Security* course. The lab is typically delivered to around 100 students at a time, and lab tasks fall into three distinct categories:

(1) Individual non-networked tasks which involve, e.g., using locally-installed software to explore a vulnerable piece of software (e.g. buffer overflows), or applying a particular security technique (e.g. steganography). Such labs allow students to explore vulnerabilities in software that would pose a danger should the software be available on standard university machines.
(2) Individual networked tasks which involve, e.g., running network scanning tools, or finding vulnerable systems (which are additional Raspberry Pi's or PC's connected to the network by the lecturer). Such labs allow students to run tools over the dedicated network without breaking various policy restrictions that are in place for the university network.
(3) Group-based networked tasks which involve, e.g., sharing data across a network, for example in order to share encrypted data with other students (e.g. using PGP), or exploring Blockchain technology. Such labs allow students to learn to use software such as peer-to-peer crypto-mining software that would be against university policy.

To give an insight into typical lab content and the hands-on competencies that students are able to explore thanks to such a lab environment, we have included appendices that present an example from each category of lab.

- Appendix A contains an example of an individual non-networked task that allows students to explore the importance of using TLS encryption. In particular, students are given an SD card that runs a web application over HTTP. They then undertake the task of session hijacking before learning how to secure the web application by creating a self-signed certificate using OpenSSL.
- Appendix B contains an example of an individual networked lab task that allows students to explore network scanning using the NMap tool. This lab relies on there being additional hosts on the network such as another Raspberry Pi running a simple web service. Students learn how to scan the network and look for vulnerabilities on the network. They also learn fundamental skills such as the use of SSH to remotely access machines.
- Appendix C contains an example of a group-based networked lab task. In this lab, students learn how to encrypt and decrypt data using PGP. They gain hands-on experience of the processes involved in encryption and decryption, and use this knowledge to share messages with friends on the network.

4.1 Problem Based Learning

In addition to supporting the individual and group-based networked tasks outlined above for large cohorts, the lab setup has also been used for a smaller (20-30 students) masters-level penetration testing course. Within this course, there is a focus on so-called Problem Based Learning (PBL) [13], whereby tasks are designed based upon real-world problems. Students then typically work in groups to research possible solutions to a problem before applying these solutions. It is a technique that is praised for developing students' learning abilities and problem solving skills.

As our approach to labs affords easy deployment of a customised network environment, it allows for a lecturer to simulate, for example, a real-world network setup, with students being deployed different systems running different software (simply by creating several differing SD card images). This allows a true hands-on experience of penetration testing. It facilitates PBL tasks as students are required to research the new network presented to them and to explore and research new vulnerabilities that may be apparent on the network. However, based on our experience, we do not believe that such an approach would easily scale beyond a small cohort due to the requirement of developing a range of Raspberry Pi images.

5 STUDENT PERFORMANCE AND FEEDBACK

Overall, the effects of the new lab have been positive in terms of: student engagement in the courses; results achieved by the students; and student feedback. Here we provide an analysis of results achieved year-on-year based on three cohorts of third year Computer Science students. In 2016-2017 the course was newly introduced and delivered with traditional paper/programming based labs, whilst in the following two academic years 2017-2018 and 2018-2019, the new Raspberry Pi based lab was used and lab tasks were adjusted based on experiences from the previous year.

5.1 Effects on Engagement and Performance

Overall, there has been an increase in the average marks achieved by students since the introduction of the new hands-on exercises, as illustrated by Table 1. This increase comes largely from two aspects. Firstly, the module lecturer has noted an increase in engagement, interest and questions from students during delivery of the module. Secondly students display a better understanding during the final examination which is worth 70% of the course mark, rather than marks achieved for lab work which is worth 10% of the course mark. This highlights that – although the content delivered in the course has remained the same – having well-designed hands-on practical experiences for students can improve their understanding and retention of theoretical concepts. Looking a little more closely into student performance, we can see from Figure 2 that there was

Year	2016-2017	2017-2018	2018-2019
Average Mark	56.8%	59.8%	63.5%

Table 1: Average student marks across three cohorts.

indeed a shift in student achievement based around the change in lab content. In particular, we can see that the percentage of students achieving below 50% shrunk after the introduction of the new lab setup, whilst the percentage of students achieving marks in the range of 50% to 80% increased year-on-year.

Students who struggle with the course (i.e. those scoring below 30%) seem to be relatively unaffected by the change in lab format. However, looking more closely at the data, these students do not engage regularly with the degree programme – they have low attendance rates across all of their courses – and thus the expected effects are not realised.

5.2 Student and Lecturer Views

Student feedback across cohorts that have used the Raspberry Pi based labs has been highly positive, with students not only commenting on the enjoyment they have from undertaking the lab:

"The labs were exceptionally useful and really interesting as we actually got to look at security vulnerabilities ourselves rather than just through theory."

but also noting that they felt the labs were very effective in helping them understand course content:

"I found that the labs were very effective and fun to do. It helped me learn a lot by just attending. It's nice to work on Raspberry Pi."

A number of students also noted that they had gained valuable practical experience by undertaking the labs:

"With the labs I got to apply and learn various techniques. Overall they were very enjoyable and I got to see theories discussed in lectures applied. I feel this really aided in my understanding. Most importantly they were fun with a decent amount of challenge presented to test my knowledge."

"I found the labs and the live demos very useful and made the course much more interesting. I was worried coming into the course that it would not be shown to us and we would just be taught theory."

These quotes illustrate that our approach meets its aims from a student satisfaction perspective. However we note that the first run of the lab was not without minor issues, particularly with regards to the process of students being able to set up their lab equipment from week to week with one student noting:

" Use a lab room that has better access to ports in the monitor. It's a struggle to plug in the Raspberry."

This feedback was taken on-board, and solved for the next cohort by simply fitting an additional HDMI cable as a permanent feature of the university lab monitors.

Similarly, in the first run of the course another student noted:

" The system changes in each lab. It's hard to review previous labs."

This highlights that some students can struggle with large changes in technology from week to week, and that when designing lab tasks, one may want to consider using, for example, similar operating systems across the labs.

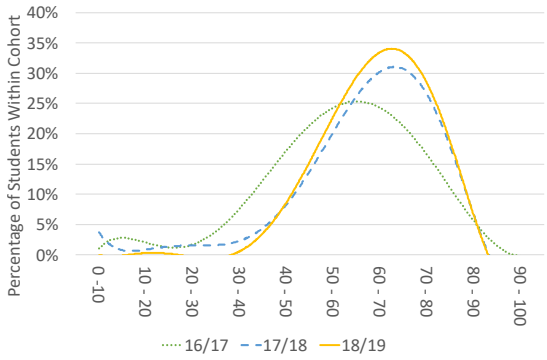

Figure 2: Student performance over previous three cohorts.

Course lecturers note that the lab setup has encouraged discussion from students, and that – with the labs providing real-world examples – students are more engaged with the topics presented in lectures. The lecturers have also observed a deeper level of knowledge exhibited by students in exams. They note that there is an initial learning curve in setting up and deploying the labs; however, after the first few SD card images have been developed, this becomes no more effort than developing any other practical lab.

6 ALTERNATIVE APPROACHES

Our aim to provide students with hands-on practical experience of security testing is of course not a new one; see for example [5, 6, 14]. It is a solution to a problem that has been re-iterated by both academics [6] and industry[5]. In particular, there have been a number of approaches that have utilised either locally-run virtual machines or complete virtual lab environments.

Haag and Vranken [7] developed a virtual lab setup that is based on students running an entire virtual lab on a single desktop PC. This relies upon students installing virtual machines on their own personal computers. There is then a series of virtualisation layers, the topmost being a single virtual machine that hosts a second virtualisation layer based on NetKit which allows the configuration of a virtual network. They note that the hardware requirements for running the virtual lab are very modest and that few virtual machines can already be run smoothly on a standard PC. In an earlier approach that focused less on virtual networks, Du et al. [5] also explored lab deployment using downloadable virtual machines. Similarly, Chothia et al. [3] have also recently explored hands-on experiences for security testing through gamification of lab tasks. Their framework is based on a single virtual machine image that students download onto their own computers at the start of the course. They note that this requires students to open this image using free and open-source virtualisation software.

The above approaches have their merits, but rely upon two fundamental constraints that are not always feasible. Firstly, they rely on running virtual machines either on university desktop PCs – which requires a technical overhead in terms of maintenance and raises questions relating to university policies on student access levels – or on students' own equipment – which raises questions of availability and reliability. Secondly, for large class teaching,

[5]https://www.infosecinstitute.com/courses/hands-on-security-training/

distribution of the virtual machines will hit network bandwidth constraints which will negatively impact on delivery.

Looking into solutions that make use of a virtual host infrastructure, there are a number of groups that have explored hosting virtual machines and networks using server setups. Wu et al. [15] have developed an entire virtual hands-on lab infrastructure. This approach has been used for large cohorts, but requires a cluster of servers with varying amounts of resources on each server. These servers are all administered and connected to two networks, namely, the production internet-accessible network and a private internal network. The private internal network handles communication between the virtual machines and provides separation between labs. This also means that the labs are not restricted to running on a single physical server.

This is a view and approach that has also been taken by Chen and Tao [2], and Gaffer and Alghazzawi [6]. We note that a number of professional industry courses make use of such hosted virtual setups by giving users access to virtual machines and networks via online labs. Whilst we agree that such approaches are indeed a good way to deliver hands-on experience to students, such a costly and high maintenance approach is not feasible within all University courses and labs, especially if security is only one course amongst many others being delivered as part of a wider curriculum. We feel that our approach – whilst perhaps slightly more restricted in terms of deployable software and systems (which are restricted to systems that can be run on a Raspberry Pi) – provides a lightweight and ad-hoc viable alternative that is less costly and less demanding on university infrastructure and technical support.

7 CONCLUSIONS

In this paper we presented a novel hands-on security testing lab based on Raspberry Pi computer systems. This lab is low maintenance, portable and independent from university networks and systems. It allows students to gain real-world experience in running security tests and, in particular, detecting, understanding and fixing security vulnerabilities. We have shown that such lab experiences improve student understanding and, whilst being enjoyable, also give students an appreciation of how the theoretical topics they are learning within lectures apply to real-world settings. Finally, we have compared our approach to others, highlighting the benefits and drawbacks when deploying labs for large cohorts of students.

Looking to the future, we would like to expand on the initial successes of the development by calling for an open effort in developing and sharing resources for use with Raspberry Pi's. Taking inspiration from projects such as SEED [4, 5] that has developed over 30 downloadable virtual machine images – most of which have been used and tested within university courses – we would like to create an open repository of Raspberry Pi lab images for students. This, we feel, would provide a number of benefits. Firstly, it would provide a wide range of resources for students to practise their skill sets. Secondly, it would allow academics to share resources, effectively reducing development time for content. Finally, in conjunction with Technocamps [9–11] we are producing a "Capture-the-Flag-in-a-Box" kit suitable for both Primary and Secondary classrooms, to inspire schools and to explore security testing.

Appendix A – An Individual Non-networked Lab Task

This lab will explore general web security. This week your Raspberry Pi will have a single user account with the username "pi" and password "raspberry".

In this lab we will consider a typical vulnerability that may be present when a website does not use SSL. You will be using Wireshark to monitor network traffic. You will then capture a session cookie and use it to log into a website without entering a user name and password. The aim of the lab is to highlight why SSL should always be used when dealing with web based authentication!

As the Raspberry Pi's do not have internet access, the lab will make use of a locally-hosted version of the lab class tracker.

Task 1: Your first task is to find the session cookie of a user who is logged into the system. To do this, start Wireshark by running:

```
sudo wireshark
```

Once Wireshark has opened, select the "any" option under the interface list and then click start. Wireshark will begin to record network traffic.

Next, open a web browser (Chromium) and start a new **incognito window**, which can be opened by selecting the three dots in the browser and selecting "New incognito window". This will ensure a new session is started with any websites you connect to. Open up the following web-page:

```
http://labsystem.local
```

Log in to the website with the credentials:

Username: `admin` Password: `secretpassword`

Once logged in, return to the Wireshark window and stop monitoring by clicking the stop button. Within the captured network traffic, try to locate the last HTTP response. Once located, expand the "Hypertext Transfer Protocol" section located in the section below the network traffic. Here you can see information about the HTTP response including session cookie information for the logged in user! This is stored in the "laravel_session" cookie. Copy this cookie information into a text file by right clicking on the entry and selecting "copy" then "value".

You have now essentially captured the information needed to gain access to the lab tracking system as the admin user, but without needing to authenticate.

Task 2: For this task you will use the information captured in Task 1 to gain unauthorised access to the local lab tracker website.

Open a new **standard** browser window and go to the lab tracker website. You should notice that you are not logged in to the system (if you are, then you did not use an incognito window in Task 1! Start again...)

Do not log in to the website. Instead, open up the "Developer tools" view by going to the three dots in the right hand corner of the browser and selecting:

```
More tools -> Developer tools
```

This will open a new section within your browser that contains a "Cookies" tab.

From within the "Cookies" tab, you can edit the values of cookies that are sent as part of HTTP requests made by your browser. Edit (by double clicking) the value of the "laravel_session" cookie to match the value you captured in Task 1. **Be careful to copy only**

the relevant value from the data you captured. Refresh the web-page, and you should see that you are now logged in to the lab system website, without authenticating!

Challenge Task: This is a bonus question that is worth an extra mark this week. Your task is to set up an SSL version of the lab tracker running on port 443. To do this you will need to read about OpenSSL and how you can use it to create a self signed certificate. You will then need to re-configure Apache (which is running the website) to use the self signed certificate. Finally, you should demonstrate that the attack from Tasks 1 and 2 is no longer possible.

Appendix B – An Individual Networked Lab Task

This week's lab will explore using NMap, a simple port scanner. Port scanning helps a system administrator (or hacker) to determine what services are running on a system. Any vulnerable or insecure services discovered may be exploited to gain unauthorized access.

For this lab, you may find the following commands useful:

- `nmap <host_range>` performs a scan of machines in the given ip range. For example 192.168.0.0/24 will scan devices with IP addresses between 192.168.0.0 and 192.168.0.255.
- `nmap --open <host>` will display all open ports on the given host, where the host is specified by host IP.
- `nmap -sV -p <port number> <host>` will aim to establish what services and versions of software are running on a given port on the given host IP.

Task 1: Establish which hosts are up and running on the network. For such hosts, find out which ports are open and what services, including versions, are running on these ports.
Task 2: Find all hosts that are running a web server on port 80. Check the web pages being served by these machines – can you find anything out? With the information you find, try to answer the question "How many goats does Geoff have?" Hint: SSH! Be quiet and look around!
Task 3: Suppose that Geoff has detected your intrusion into his files and decides to change the guest password. Can you think of a way to ensure that you still have access to the server? Hint: Look around hidden places, you may find a hint!

Appendix C – A Group Networked Lab Task

This week's lab explores PGP (Pretty Good Privacy) for key generation and encryption.
Task 1: Use PGP to generate a private and public key pair (1024 bit, RSA) for each member of the group. To do this, you can use the following command:

```
gpg --gen-key
```

During this process, you will be prompted to enter a "passphrase". This is used to encrypt your key, just in case your key gets into the wrong hands.

Once this process is complete, you should have several files that have been generated into the .gnupg folder in your home directory. Change to this folder and check you have the following two files:

- pubring.gpg – which will contain your public key, along with public keys you trust (you will see this later).
- secring.gpg – which will contain your private key; this should NEVER be disclosed.

Finally, you can use the following command to list the keys you have available:

```
gpg --list-keys
```

Now that you have a public key, we can use PGP to export a version of it which your friends can use to encrypt files that you can then decrypt using your private key. To export your key, use the following command:

```
gpg --export -a --output name-key.asc emailaddress
```

Of course, you should replace `name` with your name (no spaces!) and `emailaddress` with the email address you entered when you created your key in the previous exercise. This is the email address you see when running `gpg --list-keys`; you will get a file named "name-key.asc" containing an ascii version of your public key.
Task 2: For the remainder of this lab, you need to find a friend and exchange one member of your group with their group. Firstly, share your exported public key (name-key.asc) with your friend's group (and receive their exported public key to your group). You can do this using secure copy:

```
scp filename pi@ip_address:
```

This command will require you to know two things:

- the IP address of your friend's Raspberry Pi.
- the password of your friend's account on their Raspberry Pi.

Next, import your friend's key by running:

```
gpg --import theirname-key.asc
```

where their name is obviously the name of the file you received containing your friend's key. Again you can verify this has worked by checking that the imported key is in the list of trusted keys using: `gpg --list-keys`.
Task 3: Now you and your friend are ready to communicate securely using public key cryptography; for the final task, you need to demonstrate that you and your friend can do so. To do this, you can use gpg to encrypt and decrypt files. Create a text file with a secret message in it. You can then encrypt and send this file.

To encrypt a file (secretFile.txt) you can use the command:

```
gpg -a -e secretFile.txt
```

This will prompt you to enter the user id corresponding to the public key you want to use for encryption. You should enter the email address that your friend has used for their public key. After the command has finished, you will be left with the original unmodified file and an encrypted version of the file: secretFile.txt.asc.

You can now use the secure copy command (see above) to send the encrypted file to your friend.

To decrypt a file you can use the command:

```
gpg secretFile.txt.asc
```

GnuPG/GPG automatically figures out who the file is encrypted for, and checks to see if you are in possession of the private key (you are, as the file is meant for you!). At this point, you will be prompted for your passphrase. Once this process is finished, you will be left with a file: secretFile.txt that contains the decrypted message from your friend.

REFERENCES

[1] Harry Bulbrook. 2006. Using virtual machines to provide a secure teaching lab environment. *White paper. Durham Technical Community College, Durham* (2006).

[2] Li-Chiou Chen and Lixin Tao. 2012. Teaching Web Security Using Portable Virtual Labs. *Educational Technology & Society* 15(4) (2012), 39–46.

[3] Tom Chothia, Chris Novakovic, Andreea-Ina Radu, and Richard J. Thomas. 2019. *Choose Your Pwn Adventure: Adding Competition and Storytelling to an Introductory Cybersecurity Course.* Springer, Berlin, Heidelberg, 141–172. https://doi.org/{10.1007/978-3-662-59351-6_12}

[4] Wenliang Du. 2011. SEED: Hands-On Lab Exercises for Computer Security Education. *IEEE Security Privacy* 9, 5 (Sep. 2011), 70–73. https://doi.org/10.1109/MSP.2011.139

[5] Wenliang Du, Karthick Jayaraman, and Noreen Gaubatz. 2010. Enhancing Security Education with Hands-on Laboratory Exercises. In *5th Annual Symposium on Information Assurance (ASIA '10)*. Albany, New York, 56–61.

[6] Salma M Gaffer and Daniyal M Alghazzawi. 2012. Using virtual security lab in teaching cryptography. *International Journal of Modern Education and Computer Science* 4, 1 (2012), 26–32. https://doi.org/10.5815/ijmecs.2012.01.04

[7] Jens Haag, Harald Vranken, and Marko van Eekelen. 2019. *A Virtual Classroom for Cybersecurity Education.* Springer Berlin Heidelberg, Berlin, Heidelberg, 173–208. https://doi.org/{10.1007/978-3-662-59351-6_13}

[8] J. Keller and R. Naues. 2006. A Collaborative Virtual Computer Security Lab. In *2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*. 126–126. https://doi.org/10.1109/E-SCIENCE.2006.261059

[9] Faron Moller and Tom Crick. 2015. Technocamps: Advancing Computer Science Education in Wales. In *Proceedings of WiPSCE'15: the 10th Workshop in Primary and Secondary Computing Education*. London, England, 121–126.

[10] Faron Moller and Tom Crick. 2018. A university-based model for supporting computer science curriculum reform. *Journal of Computers in Education* 5(4) (2018), 415–434.

[11] Faron Moller, Liam O'Reilly, Stewart Powell, and Casey Denner. 2019. Teaching Them Early: Formal Methods in School. In *Proceedings of FMFun'19: Formal Methods – Fun for Everybody*. Bergen, Norway.

[12] Brian A. Pashel. 2006. Teaching Students to Hack: Ethical Implications in Teaching Students to Hack at the University Level. In *Proceedings of the 3rd Annual Conference on Information Security Curriculum Development (InfoSecCD '06)*. ACM, New York, NY, USA, 197–200. https://doi.org/10.1145/1231047.1231088

[13] John R. Savery. 2006. Overview of problem-based learning: definition and distinctions, The interdisciplinary. *Journal of Problem-based Learning* (2006), 9–20.

[14] Giovanni Vigna. 2003. *Teaching Network Security through Live Exercises.* Springer US, 3–18. https://doi.org/{10.1007/978-0-387-35694-5_2}

[15] Dinghao Wu, John Fulmer, and Shannon Johnson. 2014. *Teaching Information Security with Virtual Laboratories.* Springer, Cham, 179–192. https://doi.org/{10.1007/978-3-319-03656-4_16}