Learning From Others' Mistakes: Penetration Testing IoT Devices in the Classroom

Tom Chothia
School of Computer Science
University of Birmingham
UK

Joeri de Ruiter[*]
Institute for Computing and Information Sciences
Radboud University Nijmegen
The Netherlands

Abstract

This paper shows how it is possible to use commercial off-the-shelf IoT devices in a taught cyber security course. We argue that the current level of IoT device security makes testing them an excellent exercise for students. We have developed a course based around this idea that teaches students basic penetration testing techniques and then sets two rounds of group assignments in which they get hands-on experience with performing a security analysis of an IoT device. In the first round, the students get devices which we know are vulnerable. In the second round, the groups are mixed and they get devices with no previously known vulnerabilities. This approach enables us to provide them enough guidance in the first round to get the experience needed to perform the analysis independently in the second round. This seems to have been successful because our student teams found previously unknown vulnerabilities in five devices in the second round of tests.

1 Introduction

Capture-the-flag style exercises [VBC+14, CN15, Fen15] and simulations of vulnerable systems [MTWP15, Sch15] have proven to be popular educational tools for cyber security. However, by definition, they do not provide students with the experience of analysing a real system. On the other hand, analysis of real systems in cyber security courses are usually restricted to paper and pencil exercises, due to the legal issues involved and the complexities of attacks against systems that are actually in use. As a solution to this problem we suggest the use of commercial, off-the-shelf IoT devices.

Many IoT device manufacturers seem unaware of the security and privacy risks introduced by connecting devices to the Internet, and vulnerabilities can be found using manual analysis techniques that can be taught to computer science majors in a few weeks. A few of the many common vulnerabilities include an insecure setup procedure that can be eavesdropped to learn Wi-Fi keys [Mar14, Pen15], leaking data in unencrypted or badly encrypted communications [Bac15] and accepting unauthenticated commands [Sec15, Mar14, Pen15].

In this paper, we suggest that IoT devices make perfect educational tools: basic vulnerabilities are common enough for students to find and they give students the satisfaction of finding real security issues. We have developed a course at the University of Birmingham in which we taught penetration testing techniques to 29 Masters students. This was taught as an upper-division course, and while there were no formal prerequisites, all the students had previously taken courses in programming and networking. They had not studied any of the analysis techniques used on the course though, and the only absolute requirement was that the students were able to install and run the (Linux) tools without guidance.

In the first four weeks of this eleven week course, the students were taught how to eavesdrop on and proxy network traffic, and how to reverse engineer smartphone apps. Following this introduction to security testing, the students were given two group exercises, where they were provided with a commercial, off-the-shelf IoT device which they had to analyse. For the first group exercise we provided students with devices we knew were vulnerable in order to build their confidence and ability. For the second round we gave them IoT devices with no known vulnerabilities to test. Marks were not awarded for finding vulnerabilities, rather we looked for a comprehensive analysis of the device, which made a strong assessment of its overall security, backed up with evidence. Still, five out of the seven teams in the second round found new vulnerabilities in the device they tested, four of which we classified as critical vulnerabilities. We are currently undergoing responsible disclosure with the companies concerned.

[*]Work carried out while at the University of Birmingham

Courses which look for new vulnerabilities have been taught before. Examples include Daniel J. Bernstein's 2004 "UNIX Security Holes" course[1], which resulted in the discovery of 44 new vulnerabilities, and SANS courses, which also look at commercial software. However, these courses are generally only taught to advanced students and focus on the analysis of software. As far as we are aware, our course is the only course so far that uses hardware IoT devices and in which intermediate level students have found new vulnerabilities.

The use of IoT devices was very popular with students, who rated this course as top in the school for how much they felt they had learned and how worthwhile they believed the course was. We will be teaching the course again in the next academic year, with only a few minor changes. All of the teaching material, including the lecture slides, exercise sheets and marking guides are available at `http://www.cs.bham.ac.uk/~tpc/Edu/Pentesting/`.

Next, we will discuss the course in more detail in Section 2. We continue in Section 3 by discussing the devices given to the students and the issues they identified. In Section 4 we discuss how the students' work was assessed and in Section 5 we discuss what worked well on the course and what did not, as well as student feedback. Finally, we conclude in Section 6.

2 An IoT Device Based Security Course

2.1 Learning objectives

The main learning objective of the course is to give the students an understanding of what penetration testing is and how it is carried out. This includes familiarizing them with widely used tools and giving them real-life experiences by performing penetration tests on IoT devices. In particular, by the end of the course, the students should be able to:

- collect and analyse traffic sent between devices, smartphone apps and servers,
- understand commonly used network protocols, such as TLS and HTTP, and be able to test for weaknesses in their usage,
- reverse engineer code to look for communication protocols and common weaknesses,
- carry out a simple penetration test of an IoT device and find common vulnerabilities,
- present the results of a penetration test in the form of a report and a presentation.

The reports and presentations were expected to show evidence of all of these points, and were used as the key measure of whether the students had met these objectives.

The topics taught were chosen based on the learning objectives. We focused on teaching the use of tools that would assist the students in manual analysis. For example, we taught students how to use a proxy to examine HTTP traffic and look for weaknesses themselves, rather than focusing on the use of automated scanners to find vulnerabilities automatically. This builds a deep understanding of the underlying issues and allows students to find problems that automated scanners might miss. By the end of the course students should be able to use all of the tools taught independently.

2.2 Course schedule

Weeks 1 and 2: Introduction & website security We start with an introduction to the course itself and the concept of penetration testing. Different types of penetration testing are discussed, as well as relevant standards, such as PCI-DSS and NIST SP800-115 [Pay16, Nat08]. Both the legal and ethical issues around penetration testing are covered, followed by a quick introduction to web security, for which SQLi, XSS and CSRF attacks are discussed. A marked exercise was given, where the students were provided with a virtual machine on which a vulnerable website was running that they had to analyse. This taught the students important skills of intercepting and analysing web traffic.

- Tool: Using Burp Proxy[2] the students learnt how to proxy, intercept, view and alter HTTP traffic.

Week 3: Network protocols Network scanning was discussed as well as how cryptographic protocols work and fail. In particular the TLS protocol was introduced along with its common problems, such as acceptance of self-signed certificates.

- Tools: Students were shown how to scan networks and devices with Nmap[3], how to capture network traffic with Wireshark [4] and how to intercept TLS connections by setting up a man-in-the-middle attack using Burp Proxy.

Week 4: Reverse engineering apps Reverse engineering of Android apps was introduced in the fourth week.

- Tools: Students were shown how to use Apktool[5] to examine APK files, dex2jar[6] to turn Android code

[1]`https://cr.yp.to/2004-494.html`

[2]`https://portswigger.net/burp/proxy.html`
[3]`https://nmap.org/download.html`
[4]`https://www.wireshark.org/#download`
[5]`https://ibotpeaches.github.io/Apktool/`
[6]`http://sourceforge.net/projects/dex2jar/`

into Java class files for reverse engineering and JD-GUI[7] to decompile Java class files.

First devices issued At the end of week 4 the students picked their own teams for the first IoT penetration testing exercise. The teams then chose an IoT device to test from a list of devices that we know have vulnerabilities.

Weeks 5 & 6: Lab sessions Each student group had meetings with the course instructors to discuss the penetration testing of their device. Students were not told where to look for vulnerabilities. Instead, advice was limited to recommending that the teams worked out exactly how the setup of the device works, and what communications takes place between the device, app and cloud back-end (if any). Students were encouraged to apply what they had learnt during the course to test the security of each of these connections. Once they had results, the student teams worked on their report and presentation together with an instructor.

Week 7: Student presentations Each team submitted a written report and gave a 15 minute presentation on the results of their analysis, including, where possible, live demos of attacks. Details on how the presentations and reports were marked are given in Section 4.

Week 8: Buffer overflows and reverse engineering in IDA Students where shown how to reverse engineer executables to find functionality and, for instance, hard-coded passwords. It was also shown how simple buffer overflows can be found and exploited. This subject was taught after the first device penetration testing exercise as these are more advanced skills with a lower effort to reward ratio when looking for attacks, compared to the analysis skills taught earlier in the course.

- Tools: The free version of IDA[8] was used to examine executables and observe buffer overflow attacks in action. A limited number of IDA floating licences were used by students that wanted to use IDA with ARM, MIPS or 64-bit code in the second device test. The Metasploit Framework[9] was also introduced as a way to exploit out of date software.

Second devices issued At the start of week 8 the students had to reform themselves into different teams. The teams had to consist of members that were not part of the same team in the first IoT exercise. This was done in order to allow for maximal knowledge sharing between the different teams. The teams again chose an IoT device to test from a list of IoT devices, which had no known vulnerabilities this time.

Week 9: Lab sessions Each student group had a short meeting with the course instructors to report on the progress of the penetration test of their device. This time students were not given advice on what to look for, and were instead encouraged to apply what they had learnt.

Week 10: Automated scanning tools As the course was called "Penetration testing" we considered it to be important to teach some of the most common automated security scanning tools, such as Nessus Home[10]. This was left until the end of the course because we wanted to focus on developing an underlying understanding of the techniques and issues involved in security testing. For instance, we would prefer that a student is able to manually scan a network with Nmap, identify services and look for vulnerabilities in the Metasploit database, rather than just clicking 'Scan' in Nessus and reading the results. Also, the automated scanners we looked at could not detect the vulnerabilities found by the students in the first set of devices. Therefore, we argue that manual analysis is usually much more effective than fully automated security scanners.

Week 11: Student presentations In the last lecture of the course the students presented the results of their second penetration test exercise, which was on a device with no known vulnerabilities. Several groups also gave live demos of attacks they found. The results of these penetration tests are discussed in Section 3.2.

3 Devices Used and Results

In this section we describe the devices that were given to the students to be analysed. All devices connect to the Internet or a local network, via Wi-Fi or Ethernet. The devices are all controlled using a smartphone app, which has to be downloaded onto the owners phone. The IoT devices and the smartphone controller apps connect either via a back-end server or directly via Wi-Fi. The only exception to this was a smart padlock, used in the second round, that communicates with an app via Bluetooth.

3.1 Devices used for first group exercises

For the first exercise, we chose devices for which we knew there were easy to find weaknesses, i.e. weaknesses that could be found by directly applying just one of the

[7]`http://jd.benow.ca`

[8]`https://www.hex-rays.com/products/ida/support/download_demo.shtml`

[9]`http://www.rapid7.com/products/metasploit/download.jsp`

[10]`http://www.tenable.com/products/nessus`

	SETUP	APP-CLOUD	DEVICE-CLOUD	DEVICE-APP	CONTROL	LEAK	PASSWORD
Aria Scale (Fitbit)	✓		✓			×	
Wireless Cloud Camera		✓	✓	✓		✓	
iKettle 1.0 (Smarter)	✓				✓		✓
Indoor video camera	✓					✓	
Home security system		✓	✓				
Jumping Sumo (Parrot)	✓				✓		
Coffee machine (Smarter)	✓				✓		
Wi-Fi doorbell		✓			✓	✓	✓

Table 1: Vulnerabilities in IoT devices used for the first group exercise. ✓ indicates that the vulnerability is present and the students found it. × indicates that the vulnerability is present but the students did not find it.

analysis methods taught. This could, for example, be by eavesdropping on traffic using Wireshark and spotting an unencrypted password, or by proxying a TLS connection using Burp Proxy and finding that it accepted self-signed certificates. The devices were selected based on information available online (e.g. [Mar14, Pen15, Bac15, Sec15]) or following a quick manual analysis of devices we already owned. The reason for selecting a first set of devices with easy to find vulnerabilities was to make sure the groups would be able to find something using the methods they had learnt during the course. This builds the students' confidence in their own abilities to analyse these devices and gives them the satisfaction of finding real security issues.

The weaknesses in the devices can be divided into seven categories:

- SETUP: an insecure device setup procedure leaks sensitive information, such as the user's home Wi-Fi password, to local attackers.
- APP-CLOUD: the smartphone app leaks passwords, or other confidential data, when communicating to back-end cloud servers due to, for example, accepting self-signed certificates or not using encryption at all.
- DEVICE-CLOUD: the device leaks passwords, video or other confidential data when communicating to back-end cloud servers.
- DEVICE-APP: direct communication between the smartphone app and the device leaks passwords, video or other confidential data.
- CONTROL: the device can be controlled by an unauthenticated user on the same network.
- LEAK: the device leaks passwords or other confidential data on the local network (e.g. via unauthenticated access to a website showing the current settings).
- PASSWORDS: hardcoded default passwords are used in the app or device.

The vulnerabilities present in the tested devices can be found in Table 1. The majority of these vulnerabilities involved sensitive data being sent unencrypted or protected by a TLS connection for which self-signed certificates are accepted. The student teams could find these by monitoring traffic with Wireshark and proxying TLS connections using Burp Proxy. Some vulnerabilities only appear during setup whereas others only appear once the device is actually used. Therefore, the students had to work out how the device operated both during setup and when in use. Performing a scan using Nmap on the devices and sending messages to the open ports allowed the students to control some of the devices and find configuration information including Wi-Fi passwords. The smartphone apps could be reverse engineered to find the commands used to control the corresponding device. The reverse engineering also revealed hardcoded passwords included in the app.

The students were not guided to particular vulnerabilities but, as Table 1 shows, they were able to find most of the vulnerabilities with just the methods learnt in the first four weeks of the course. For the vulnerability missed: the team with the Aria Scale did Nmap the device and find open ports, but did not investigate them further.

3.2 Devices used for second group exercises

For the second round of tests, we used devices that did not have any known vulnerabilities. The seven devices used were selected from popular IoT devices from Amazon and leading manufacturers, such as Samsung. It was made clear to the students that the aim of this exercise was not to find vulnerabilities, but rather to carry out a convincing assessment of the device. This means a well-argued report showing that the device avoids all basic security flaws would get as many marks as, if not more than, a report that found a major vulnerability. Indeed, we expected most of these devices to be secure but, to our surprise, most of the groups found at least some weakness in the device they analysed.

We required the students to form teams with none of the same members as in the first round, to accommodate this we had to make one large team of five people, hence we only had seven teams for the second round. The devices analysed and the students' results are as follows:

- The Samsung SmartThings Starter Kit[11]. No vulnerabilities were found for this device. The setup was done using an Ethernet cable to a home router. There was no direct communication between the hub and the phone, as all communication went via the back-end server. These connections used a TLS connection with proper checks to prevent the use of self-signed certificates. We note that a recent paper [FJP16], released after the students analysis, found that 3rd party apps for Samsung SmartThings were overprivileged, and this could be abused. This was outside of the scope which the students defined for their analysis, as was analysing the Z-wave and ZigBee protocols used to connect devices.

- The Canary All-In-One Home Security Device[12]. No vulnerabilities were found with this device. An audio cable connecting the device and a smartphone was used to securely set up the device, after which all communication takes place via the Amazon cloud over channels that are using correctly configured TLS. The control app was heavily obfuscated, so the students did not have enough time to analyse it. The team did find that the device uses Bluetooth Low Energy, which has known vulnerabilities [Rya13], and they flagged this as a minor cause for concern.

- A smart camera with face recognition. The students found that this device had a secure setup and used TLS correctly. However, when the app and camera were on the same local network, video was sent unencrypted and could be eavesdropped. This camera had an additional feature that used face recognition to identify people and could send an alert to the owner if a stranger entered their home. The students found that the face recognition could be fooled using photos of people taken from their Facebook profiles.

[11]`http://www.samsung.com/uk/smartthings/`
[12]`https://canary.is`

- A smart electricity socket. By monitoring local network traffic with Wireshark, the students found that messages sent to the socket were not authenticated, and they then reverse engineered the app to find the commands to control the socket.

- An outdoor CCTV camera. The students found that this camera sent video, usernames and passwords unencrypted on the local network. By proxying the app traffic in Burp Proxy, they found that when using the app to view video from a remote location all configuration information was sent unencrypted to the app; this included the users home Wi-Fi password.

- A smart padlock. As well as opening and closing the padlock from your phone this device also lets users grant temporary access to the padlock to another user via e-mail. The team found that the padlock communicated securely with the smartphone app using Bluetooth, and the app communicated securely using TLS. By installing their own TLS certificate on the phone, eavesdropping the traffic and reverse engineering the app, they found that the permission to open the lock was not tied to the e-mail address, but could be used by anyone that got hold of the link in the e-mail. They also found that the temporary permission included a validity period, to indicate when the app was allowed to unlock the lock, and a master code that could open the lock at anytime. The validity period was only enforced by the app, and so could be bypassed by an attacker.

- A second smart home alarm system: this device has a secure setup phase, uses encrypted radio messages to communicate with sensors and uses TLS correctly to protect communications with its back-end cloud server. However, the team found that the smartphone app also used an unencrypted websocket to communicate with the back-end. The messages to the server used an authentication token; by monitoring the websocket traffic and reverse engineering the app, the team found that an attacker that could eavesdrop on the app could re-use the token on a secure TLS connection to send a forged message to the back-end server in order to disarm the alarm, reset user details, including passwords, or put the alarm into panic mode.

3.3 Responsible disclosure

We are currently in the process of responsible disclosure with the companies that manufacture the vulnerable devices listed here. Once this process is finished we will include full details of all devices on our website. We note that some IoT manufacturers such as Fitbit and Smarter have decided not to fix vulnerabilities in their products if they can only be exploited at setup time or by an attacker on the local network. Therefore, it is possible that some of the new vulnerabilities discovered by the students will also not be fixed. Although this is not a part of their grade, we have involved the students in this, to give them experience with the responsible disclosure process and show that their work had a real impact.

4 Student Assessment

Student assessment was based only on coursework, as there was no exam. The web and buffer overflow exercises were marked as normal exercises. The device penetration tests were marked on the following criteria:

- *Analysis of Device Functionality 25%*: How well have the students discovered how the device operates, and tested it for security vulnerabilities? The students were expected to check for common issues on all connections and the device itself. However, checking for advanced issues such as buffer overflows in the device firmware or hardware side channel attacks was not expected.

- *Risk Analysis 10%*: Was the scope of the analysis clearly defined and sensible? Were all risks identified and their criticality appropriately rated? For teams that did not find vulnerabilities, did they make a convincing case that it was likely that no simple vulnerabilities existed?

- *Report 20%*: Were all issues clearly identified and a completely convincing case made to support the findings? Were all key issues discussed in a precise and logical, yet concise manner?

- *Substantialness of Achievement 15%*: Did the team solve challenging and difficult problems?

- *Presentation 20%*: Was the presentation well-prepared and clearly showing all key findings? Was a live demo of an attack included, where appropriate?

A full detailed marking guide is available online[13]. We note that marks were not awarded for finding vulnerabilities, rather for a full and comprehensive analysis of the device, backed up with a well written and well argued report. Indeed, in the second round of tests the two groups that did not find vulnerabilities achieved the top and third best score, based on the quality of the analysis undertaken.

[13]`http://www.cs.bham.ac.uk/~tpc/Edu/Pentesting/`

The students were also required to include a breakdown of what each individual member of the team worked on and how much they contributed to it as a percentage. This teamwork report accounted for the final 10% of the testing marks. Teams received a mark based on how well they worked together. We note that the aim of this was not to rate individual students, but rather to encourage the students to work together and find ways for everyone to contribute. This seems to have been successful because all teams reported a significant contribution from all team members.

All teams produced good first round penetration testing reports, although some teams lost marks for overestimating risks, technical mistakes (such as assuming that a device had a static IP address when it was in fact being assigned by DHCP), incomplete assessments (such as not checking all the open ports) or including too many irrelevant details. In the second round, all teams managed to find vulnerabilities, or make a convincing case that their device was secure, showing that they had achieve the learning objectives.

After the first round of reports we required each student to read all of the other teams' reports and write a 2-page assessment. The aim of this was to let the students see and think about what the other teams had done, both on the technical and presentation side. The reviews were marked on the technical assessment of the other teams' work and the justification of the comments. This exercise seems to have worked well for sharing good practice between the students because the reports were significantly better in the second round.

The final mark for each student was calculated as 10% for the written exercises, 40% for the first penetration test, 10% for the assessment of the other reports and 40% for the second penetration test. As well as the teamwork benefits, getting all students to change teams for the second round of tests lead to more distributed marks that better reflected the students' ability. No student failed the course, and the final distribution of marks was comparable to the marks that the same cohort of students achieved on other courses with written exams.

5 Discussion

The key aspects of our course are the two large penetration testing group exercises on off-the-shelf IoT devices. The students had all previously taken courses in computer science, but had not seen the analysis methods taught on the first part of the course. From the results, it is clear that the students learnt enough to effectively analyse the devices they were given. We will certainly be running the course in the same format again next year.

The total cost of the devices used on the course was $\sim$\$1600, this works out at around \$50 per student taught.

These devices can be reused from one year to the next, although we do intend to provide new devices next year for the second round of tests, to again give the students a chance to find new vulnerabilities.

Letting the students pick their own teams for the first exercise but forcing them to switch for the second exercise worked well in terms of knowledge sharing. The same held for having each student review all of the reports from the first round. Reviewing the other reports forced students to consider their own work and this indeed resulted in much better reports in the second round. Additionally, students learned what technical methods worked well, and, if they were not already using them, could try them out in the second round. An example of this is that one team successfully used *iptables* and *sslsplit* in order to analyse TLS traffic from a device in the first round, so in the second round, three teams used these in their analysis (and we will teach these tools as part of the course next year). Some teams included day by day logs, reporting on what they tried and what the results were. This was very helpful when marking the report and will be made compulsory next year.

Another method of grading would have been to use peer assessment and award the teams grades based solely on the review by other students. This approach would massively reduce the time needed to run the course. However, this may not have the same educational benefits as marking by the course instructor.

As students were acting as attackers it is important to consider legal and ethical issues. In the first lecture we outlined the relevant laws in the United Kingdom, which allow the students to "hack" their own devices. We also made it clear that attacks against the cloud back-end used by a device were strictly out of bounds. Before issuing the devices we required each student to sign a declaration stating that they understood this, and if they found vulnerabilities they would not make them public without the explicit permission of the course staff (the form we used for this is available on our website).

Defining the scope of the analysis We let the students define the scope of their analysis, where our only requirement was that attacks against cloud back-ends were strictly out of scope. We found this to be a useful exercise as it required students to think about their devices and possible security models. An alternative would have been to define the scope of the tests for the students, e.g., by restricting the analyses to the communication channels between the device, app and cloud, and to only local attackers with Wi-Fi access or remote Internet attackers. This would have made the group penetration tests a little less challenging, requiring less high-level analysis, Also, it might have meant that the students would have missed some of their most interesting results, e.g., the broken invite system on the smart padlock.

Student feedback At the end of all courses taught in our school, students fill in detailed feedback questionnaires. These let students rate a wide range of aspects of the course on a scale from 0 (worst) to 4 (best), as well as providing free form comments about what they liked and what could be improved. Our penetration testing course obtained the maximum possible score of 4 out of 4 for how worthwhile the students found it, how much they were learning and how interesting they found the course. It was the only course taught in our school to get such positive feedback. The students rated the difficulty of the course as 2.74 out of 4, slightly harder than the school average of 2.66. In terms of how happy they were overall, the students rated the course as 3.92 out of 4. In the free form answers students highlighted the practical nature of the course, the skills they had learned and the IoT devices as particular positive aspects. The only suggestion for possible improvements was to allow more time to analyse the devices.

To dig deeper into the students' feelings about the course we carried out a focus group study with 12 student volunteers from the course. As a self-selecting sample, their views may not have been representative, but it did represent a significant proportion of the 29 students that took the course. We found that the students gave their primary reasons for liking the course as "the interesting lectures" and "passionate" course staff, closely followed by "finding real attacks", "interesting IoT devices" and "learning hands on pentesting". Students also stated that they liked "thinking like an attacker". Seven of the students stated that they were going to list the new vulnerabilities they had found as a point on their resume/CV. This suggests that, while it was not the primary cause of the positive feedback, the use of IoT devices and real vulnerabilities did play an important part in the course's popularity with students. When pushed for negative comments, there was a consensus that the students would have liked more time to study the devices. A few students also felt that the teams were not well balanced; probably a fair criticism, given that the students were allowed to form their own teams. This issue could be addressed by assigning teams at random.

Plagiarism This may be an issue with any course that does not have a final exam. Assigning each team a different device largely negates the possibility of students copying off each other. However, for the first round devices some vulnerability reports could be found online. We note however that we called for a complete analysis of how the device functions and that online vulnerability reports only contain information about a single vulnerability. Therefore, copying this would not directly lead to a high mark. In addition, the students also had to present their work to the whole class, including a demo of attacks found, and answer questions about it from the class, showing that they are able to repeat and understand the attacks. We note that, in the first round, one of the teams missed a vulnerability that is reported online suggesting that all of their findings really were their own work. If plagiarism in the first round became a concern, teams could additionally be interviewed to test that the students really did have a deep understanding of the attacks they were presenting. Using devices with no known vulnerabilities for the second round made plagiarism impossible.

6 Conclusion

We have shown that it is possible to use off-the-shelf consumer IoT devices as the basis of a successful cyber security education course; these provide a realistic alternative to CTF challenges or other comparable exercises. The results of our course bear out our assumption that the security of many consumer IoT devices is weak enough to make analysing them a perfect student exercise. Our course worked well and was very popular with students. Therefore, we would encourage other educators to consider using IoT devices as examples and exercises in their own courses.

The sustainability of a course such as this is an open question. It is possible that IoT device manufacturers will secure their devices, and so remove the rationale for this course. A continuous supply of broken IoT devices would also help keep the course fresh. So, a marked increase in the security of IoT devices would make the course difficult to teach well. However, companies such as Smarter and Fitbit have been aware of the vulnerabilities in their systems for some time and do not seem inclined to fix them. Many devices cannot even be patched, so will remain broken, while other manufacturers seem to consider attacks from inside the local network not important to defend against. We believe this means that it will be possible to teach a course based on broken IoT devices for many years to come.

Acknowledgement Part of this work was supported by the grant GEN1214 from "The Higher Education Academy". We thank the anonymous referees for their helpful comments that improved the paper.

References

[Bac15] Backspace. Fitbit Aria Wi-Fi smart scale, `https://www.hackerspace-bamberg.de/Fitbit_Aria_Wi-Fi_Smart_Scale`, 2015.

[CN15] Tom Chothia and Chris Novakovic. An offline Capture The Flag-style virtual machine and an assessment of its value for cybersecurity education. In *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE '15)*, 2015.

[Fen15] Wu-chang Feng. A scaffolded, metamorphic CTF for reverse engineering. In *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE '15)*, 2015.

[FJP16] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. Security analysis of emerging smart home applications. In *Proceedings of the 37th IEEE Symposium on Security and Privacy*, May 2016.

[Mar14] Mark J Cox. Hacking a Wifi kettle, `http://www.awe.com/mark/blog/20140223.html`, 2014.

[MTWP15] Jelena Mirkovic, Aimee Tabor, Simon Woo, and Portia Pusey. Engaging novices in cybersecurity competitions: A vision and lessons learned at ACM Tapia 2015. In *2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE '15)*, 2015.

[Nat08] National Institute of Standards and Technology (NIST). Special publication 800-115: Technical guide to information security testing and assessment. Technical report, 2008.

[Pay16] Payment Card Industry (PCI). Data security standard: Requirements and security assessment procedures v3.2. Technical report, 2016.

[Pen15] PenTest Partners. Hacking a Wi-Fi coffee machine, `https://www.pentestpartners.com/blog/hacking-a-wi-fi-coffee-machine-part-1`, 2015.

[Rya13] Mike Ryan. Bluetooth: With low energy comes low security. In *7th USENIX Workshop on Offensive Technologies (WOOT '13)*. USENIX, 2013.

[Sch15] Schreuders, Z. C. and Ardern, L. Generating randomised virtualised scenarios for ethical hacking and computer security education: SecGen implementation and deployment. In *Workshop on Cybersecurity Training & Education (VIBRANT15)*, 2015.

[Sec15] Security Affairs. How to hack a Parrot drone on the fly `http://securityaffairs.co/wordpress/39363/hacking/hacking-parrot-drones.html`, 2015.

[VBC$^+$14] Giovanni Vigna, Kevin Borgolte, Jacopo Corbetta, Adam Doupé, Yanick Fratantonio, Luca Invernizzi, Dhilung Kirat, and Yan Shoshitaishvili. Ten years of iCTF: The good, the bad, and the ugly. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE '14)*, 2014.