An Offline Capture The Flag-Style Virtual Machine and an Assessment of its Value for Cybersecurity Education

Tom Chothia
School of Computer Science
University of Birmingham, UK

Chris Novakovic[*]
Department of Computing
Imperial College London, UK

Abstract

Online Capture The Flag (CTF) competitions are a popular means of engaging students with the world of cybersecurity. This paper reports on the use of a virtual machine (VM) framework that has been developed as part of cybersecurity courses offered to both second-year undergraduate and master's degree students in the School of Computer Science at the University of Birmingham; the framework features CTF-style challenges that must be solved in order to complete the courses' formative assessment. As well as acquiring flags from the framework, students must also provide traditional written answers to questions and sit an examination. We analyse how well students' performance on the CTF-style challenges correlates with their achievement in the remaining formative assessment and examination, thus providing evidence to show whether CTFs are effective as an assessment tool in academic cybersecurity courses.

1 Introduction

Live security exercises, such as *Capture The Flag* (CTF) competitions, are a popular and fun means of engaging with cybersecurity topics. The aim of these events is typically for a team of participants either to defend a host running vulnerable services while simultaneously attacking other teams' hosts running the same services (*attack/defence-style CTFs*), or to solve standalone challenges from a range of categories, such as reverse-engineering, forensics and web security, without the need to directly attack other teams (*Jeopardy-style CTFs*). In either case, the successful exploitation of a vulnerable service or solving of a challenge reveals a *flag* to the victor, which scores points for the team when submitted to a *flag server* operated by the CTF organisers; the submission of flags for more difficult challenges scores greater numbers of points for the team, and the team that scores the most points over the course of the competition is declared the winner.

The pedagogical benefits of academic CTF competitions — the largest and longest-running of which, iCTF[1], has been held annually since 2002 — have been widely reported (e.g., [8, 6, 4]); amongst the most commonly-reported are students' increased motivation to learn about cybersecurity, the enjoyment and satisfaction of competing, and improved practical knowledge of theoretical aspects of cybersecurity as a result of participating. It would surely, therefore, be desirable to introduce CTFs into the academic curriculum. However, there are several barriers to doing so:

1. The infrastructure underpinning a CTF is typically large and complex (e.g., [8, 4]); the computing power (and, for attack/defence-style CTFs, network bandwidth) required for the smooth operation of a CTF is usually not available for teaching purposes.

2. CTFs require continuous supervision by their organisers to guarantee the smooth running of the competition. Given that most CTFs run over a short period of time (usually 12–48 hours), a pool of organisers working intensive shifts can ensure that the CTF runs successfully (although this is not always the case: our University's CTF team [1] frequently reports broken CTF challenges, which are then fixed by dedicated CTF organisers). This would not be viable in an academic teaching scenario where courses run over a period of several months.

3. CTFs inevitably involve attacking vulnerable services available on a network. We, and others [2], have found University IT support staff hesitant to allow malicious traffic to pass over any network they control. Additionally, there is a danger that stu-

[*]This work was undertaken while the author was affiliated with the School of Computer Science at the University of Birmingham.

[1]The UCSB iCTF: `http://ictf.cs.ucsb.edu`

dents' attack traffic could interfere with other students or machines that are not part of the competition.

We have developed a framework that enables academic cybersecurity courses to feature Jeopardy-style CTF challenges. The framework takes the form of a virtual machine (VM) containing vulnerable services and challenges devised by the course staff; they can be revealed gradually to students as the relevant cybersecurity topics are taught in lectures, allowing individual continuously-assessed exercises to feature particular challenges. Each student runs the VM locally and attempts to solve each challenge inside the VM as they are made available by the course staff. The successful completion of a challenge reveals a flag to the student, which can be submitted to an automated system controlled by the course staff for credit in the exercise; flags are unique to a particular instance of the VM, allowing for the detection of collusion between students.

This framework provides the benefits of CTFs to cybersecurity students while avoiding the drawbacks listed above: students run the virtualised framework on their own hardware (e.g., their laptops), so course staff need not invest time continually maintaining a centralised infrastructure to operate the CTF, and malicious traffic is only routed inside a particular student's virtual network, eradicating the impact that an inexperienced student's actions may have on the university network.

The students are not given the root password for the VM by the course staff, but since they have control of the virtual disk image they can gain root access and sidestep some of the challenges in order to obtain flags. Since it is impossible to prevent this from happening, in each exercise we aimed to make obtaining a flag by such methods more difficult than simply completing the exercise as intended, e.g. by placing the flag in obfuscated, compiled code.

Our VM-based framework has been used in three iterations of our introductory cybersecurity courses, both at graduate and undergraduate level. All of the continuous assessment for these courses was based on our framework; we therefore expected students to interact with the framework for 3–4 hours per week over 11 weeks. The introduction of VM-based CTF-style exercises was popular with students, as indicated by a significant increase in student satisfaction levels reported in the end-of-course feedback questionnaires.

Students were required to submit not only flags, but also written descriptions akin to traditional formative assessment (e.g., detailed descriptions of the steps they took to recover a flag, suggested fixes, discussion of the security issues, etc). These written answers were marked, and feedback provided. The submitted flags were only used by markers as a sanity check for the students' written descriptions; however, we can compare the overall marks the students achieved in their continuously-assessed work with the number of flags they managed to find, and therefore we can test whether the ability to acquire flags in CTF-style challenges correlates well with the marks students would achieve in traditionally-marked cybersecurity exercises.

We found that for students obtaining low and medium marks there was a very strong correlation between the number of flags found and marks achieved. However, for higher-attaining students the relationship between the number of flags found and marks achieved was weaker. After further analysis, it seems that the ability to acquire flags in CTF-style challenges is an excellent indicator of students' basic knowledge and technical ability, but does not accurately assess whether students have a deeper understanding of the underlying cybersecurity issues, which can be assessed with written answers. This means that CTF-style challenges are an excellent complement to the formative assessment of a taught cybersecurity course: they are better than no formative assessment at all, but should not replace written assessment altogether.

In the following section we present the VM-based framework we have used in our courses, then in Section 3 we compare students' performance on the CTF-style challenges with the marks they received on their written assessment. We conclude in Section 4. The VM and example exercise sheets are available from our website[2].

2 A VM-Based Framework for CTF-Style Challenges

Our design goal was to create a VM-based framework in which students could complete Jeopardy-style CTF challenges for credit, in a fun and accessible way. While the use of VM-based exercises to support cybersecurity education is common (e.g., [2, 7]), our framework includes exercises that assess the full range of cybersecurity topics taught in the course and gives students flags for completing the exercises. Our other goals were that:

- Students should submit flags online, and their validity should be automatically and instantly verified. To avoid duplicate submissions, flags for a single challenge should be different for every student.
- Students should be able to set up the VM on their own hardware and attempt to solve the challenges offline. We did not want to require students to

[2]A CTF-Style VM Framework: `http://www.cs.bham.ac.uk/~tpc/SecEduVM/`

have an Internet connection at all times and we have also previously observed students encountering many problems with virtual networking setups; we did not want to have to dedicate staff time to providing support for these issues.

- The exercises should slowly introduce students to Linux (with which many had no previous experience); beyond this, only cybersecurity knowledge taught in the course should be needed to complete the exercises. Cheung et al. [3] identify the lack of knowledge as a major barrier to competing in CTFs, so exercises should be made accessible to students.
- CTFs are inherently competitive, but exercises should not be perceived as a "competition" between students. Weaker students should not be publicly perceived as "losing" in a taught course; instead, each student should be able to work on exercises at their own pace, and students submitting shortly before a deadline should receive the same credit as students completing exercises well ahead of it.
- Finding flags by sidestepping the challenges (e.g., by disassembling binaries, or by mounting the virtual disk image and manually searching the file system for flags) should be more time-consuming than solving the exercise as intended.

2.1 The Framework

Our framework is based on a single VM that students download at the start of the course and import into the open-source VirtualBox virtualisation software[3]. This VM runs a Linux operating system containing several services, such as a web server, database server, and daemons running purpose-built insecure protocols, as well as many user accounts and complex, flawed access control configurations.

When the VM boots for the first time, a program runs that generates the flags. These flags are written to particular locations (e.g., into the source code of a service which is then compiled, or into a password-protected MySQL database). The startup program sets appropriate permissions on the generated files and binaries, and deletes the source code it compiled before deleting itself.

The startup program generates a random *VM identifier*, intended to be unique for each VM. To generate a flag, this VM identifier is combined with *exercise number* and *question number identifiers* and is then encrypted with a static 128-bit AES key chosen by the course staff; the resulting ciphertext can be represented as a string of 32 hexademical characters, and these are the flags the students are required to submit. Students submit these flags by logging into the flag submission server — a website operated by the course staff — using their University username and password and pasting the 32-character string into the input box for the appropriate exercise and question.

The flag submission server decrypts the ciphertext and verifies that it is a valid flag for the given question by checking the identifiers contained within the resulting plaintext. The server records details of the flag submission for the markers and instantly informs the student whether the submission was successful. Students are allowed an unlimited number of flag submission attempts for a particular question. To avoid the CTF-style challenges being perceived as competitions, students cannot check whether other students have submitted flags for a particular question.

An administrative page on the flag submission server shows the course staff details of all flag submissions for a given exercise and question, including the identity of the student submitting the flag, when the submission was made, and the identifier of the VM from which the flag originated. Students sometimes complete exercises using multiple copies of the VM (e.g., one on a laptop and one on a desktop computer), so we occasionally detect multiple VM identifiers for a particular student. However, students are explicitly instructed not to share their VMs with other students, so two students submitting flags with the same VM identifier is a possible indicator of plagiarism. The flag submission server does not reveal the results of this check to the student, but instead alerts the course staff of any irregularities on the administrative page; the course staff then follow this up with the implicated students individually (as we discuss in Section 3).

It would be possible for good students to download a fresh copy of the VM and reverse-engineer the flag generation program, giving them the ability to generate their own flags. However, we note that doing this would be harder than completing any of the exercises, in particular the existing reverse-engineering exercises. Additionally, we require students to submit written work describing how they solved each exercise, which would be hard to do convincingly without solving the exercise as intended. A larger concern is students finding a way to gain root privileges on the VM and then finding information that makes the exercises trivial. To counter this, we have designed the exercises to, wherever possible, not rely on the secrecy of particular files (e.g., in the web security exercise, the website's PHP code is not considered secret, and students are given access to it along with the exercise sheet).

[3]Oracle VM VirtualBox: `https://www.virtualbox.org`

2.2 Course Mechanics

We used our framework in introductory cybersecurity courses with a particular focus on technical skills and understanding. One version of the course was run for master's degree students, who were expected to have a good first degree in computer science but not necessarily any previous knowledge of cybersecurity; another version of the course was run for second-year undergraduate students with basic programming and networking skills. The latter version of the course used the same basic framework as the former, but with fewer and simpler exercises.

Topics covered included cryptography, access control, network and protocol security, web security, and — for the master's degree students — buffer overflows and reverse-engineering of binaries. Both versions of the course ran for 11 weeks with 2 hours of lectures and 2 hours of lab sessions per week, and students were given 2–3 weeks to complete each exercise. Exercises were marked out of 100; grade boundaries were positioned at intervals of 10 marks between 40 and 70, with a mark above 70 representing high-quality work and a mark below 40 representing failure.

The difficulty of the exercises was set based on experience gained from teaching previous courses and knowledge of the other courses that the students had taken. Generally, a third of the marks were awarded for a basic solution (e.g., a working attack on a protocol), another third of the marks were awarded for an optimal solution (e.g., a protocol attack with no unnecessary steps), and the final third were awarded for showing a clear understanding of the problem (e.g., a description of each line of the protocol attack, making it clear what each part of the message does, and why the attack works). Students were informed in advance of how marking would take place and were told what would be expected to receive top marks. Complete marking guides for all of our exercises are available on request. Marks were not scaled, but — as shown by the figures below — naturally ended up distributed across all grades.

2.3 The Exercises

For all exercises, students were required to submit written answers describing the steps they took to recover flags from the VM, and — where appropriate — a description of what the vulnerabilities were and how they worked, and an explanation of how they could be fixed. Not all questions required the recovery of flags from the VM, and flags were not used to unilaterally prove that the student completed the exercise, but were instead intended to give the marker some degree of assurance that the students' written answers were dependable. There were minor variations to the questions in the exercises for each course, both to make the second-year undergraduate version of the course easier than the master's degree course, and to prevent students from simply reusing solutions from a previous iteration of the course. Examples of the full exercise sheets are available on our website.

Basic encryption. The first exercise familiarises students with the VM, Linux (since many begin the course with little or no experience of it), and performing cryptographic operations in Java. The students are given the passwords to two user accounts on the VM, `alice` and `bob`. `bob`'s home directory contains incomplete Java source code for an encryption application; the supplied code contains methods for encrypting files using AES (in either CTR or CCM mode) and RSA using a user-defined key. The CTF-style challenges for this exercise involve decrypting AES/RSA-encrypted files in `bob`'s home directory that each contain a flag; students are required to write the corresponding decryption methods for the application and use their compiled code to recover the flags from the encrypted files.

Additionally, the startup program encrypts the message "Pay Tom 1000 pounds" using AES in CTR mode and writes it to a file in `bob`'s home directory; students are not given the encryption key, but they must manually edit the bits of the ciphertext (e.g., with a hex editor) so that, when decrypted, the corresponding plaintext reads "Pay Bob 9999 pounds". Students are also required to set up PGP for their email account, upload their public key to a keyserver, ask their friends to sign their key, and send a signed and encrypted email to a bot operated by the course staff; the bot automatically checks that the student has done this correctly and replies with feedback.

Access control. The VM contains a number of files protected with flawed access controls. One flag is stored in a file hidden in an obscure location in the file system rather than given appropriately restrictive permissions; others are stored in files with a variety of insecure permissions that allow the file to be read (e.g., via a chained confused deputy attack against two insecure programs with the `setuid` bit set). The challenges in this exercise require students to bypass these weak access controls and recover the flag contained within each file.

Protocol analysis. Students are given two key-agreement protocols in the standard "Alice and Bob" security protocol notation, Java source code of servers implementing these protocols, and pcap traces of clients communicating with these servers using these protocols. The protocols are vulnerable to a man-in-the-middle attack and a replay attack respectively. Students must discover and describe the attacks, implement them, and run them against the servers running on the VM; successful exploitation of each server reveals a flag.

Web security. The VM runs a web server that hosts what at first sight appears to be an online furniture store; however, hidden inside this furniture store is a black market website. Students must investigate this website, find the black market interface and carry out a number of attacks. An SQL injection attack allows them to view all products stored in the backend MySQL database, including the black market products and a flag. By analysing the cookies set by the furniture store, they can discover how the hidden website authenticates its users and gain access to it. The site displays a flag from a protected file that the students can submit to show that they accessed the hidden website. A file upload attack allows them to recover the MySQL server login details used by the black market website; logging into the MySQL server using these credentials reveals another database containing a flag. A shell injection attack provides access to a shell running as the `www-data` user, revealing a final flag in a protected file in the file system. Non-flag questions in this exercise additionally cover XSS and CSRF attacks.

Reverse-engineering. For this exercise, students must reverse-engineer four programs: two written in heavily-obfuscated Java and compiled into JDK bytecode, and two written in C and compiled into native code. All of these binaries behave as servers, and are listening on ports on the VM. Students must reverse-engineer the first three programs to recover a password expected to be entered into the program which, when entered into the copy of the program running as a server on the VM, causes the program to output a flag. The final program is an x86 binary for managing PGP keys, which also contains a backdoor; this binary listens on a port on the VM and runs as the `root` user. Students must find the backdoor by examining the binary in the free edition of the IDA disassembly tool[4] and use it to gain access to a root shell on the VM; a final flag is stored in a file in the `/root` directory. We also intended to place a buffer overflow vulnerability in one of the native-code binaries and require students to exploit it to recover another flag; however, we ultimately felt that buffer overflow exploits would be beyond the scope of either course.

As an unofficial (and unassessed) sixth exercise, curious students were encouraged to discover how the VM framework works. Using the skills they learned in the course, the top 10% or so were able to deduce how the flags must have been generated, and then re-downloaded the VM and reverse-engineered the startup program.

[4]IDA: Freeware Version: `https://www.hex-rays.com/products/ida/support/download_freeware.shtml`

2.4 Student Perception of the Courses

At the end of each course, students filled in detailed end-of-course feedback questionnaires. Students stated that they spent an average of 6 hours per week working on these courses and rated them as difficult (on average the fifth most difficult of the 35 courses offered by the School); this is supported by comparing the final marks that second-year undergraduates achieved on the Spring 2014 iteration of our course with the average final marks for the other courses they took (Figure 1): the average for our course is slightly lower than the overall average. However, students also rated our courses as either the most or second-most worthwhile course they took, rated themselves as very happy with the course overall, and rated each iteration of the course as one of the top courses offered by the School.

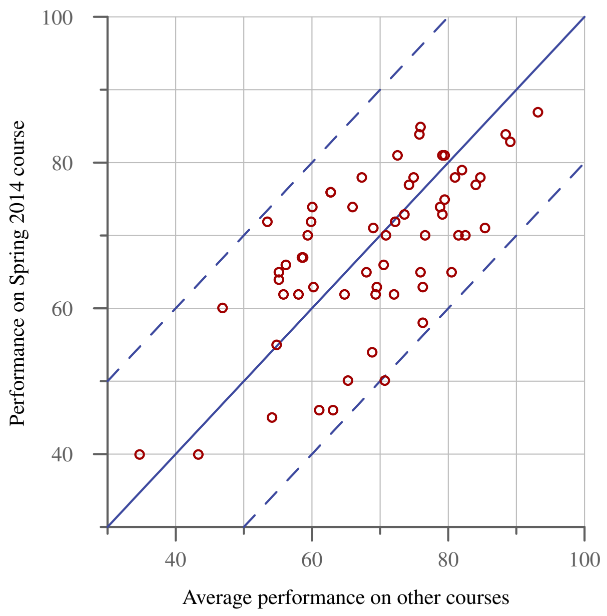
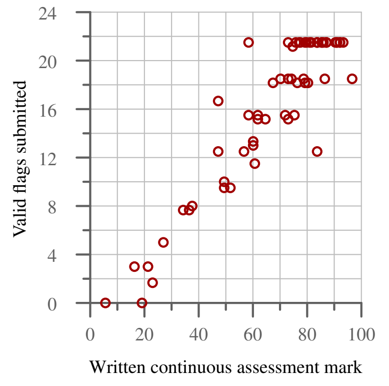


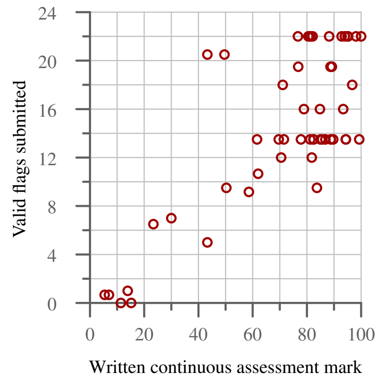
Figure 1: Second-year undergraduate students' final marks in the Spring 2014 iteration of the course vs. their average marks in other courses taken that year

3 Analysis of Flag Submissions and Marks

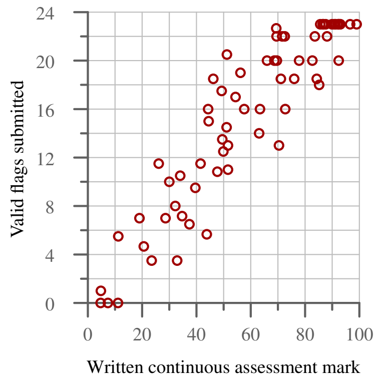
The submission of flags to a centralised course staff-controlled flag submission server allowed us to track the identities of students submitting flags, the unique identifiers of VMs deployed on students' hardware, and the frequency of flag submissions over time, across all iterations of the courses. Using this data, we wish to measure



(a) Spring 2014



(b) Autumn 2014



(c) Spring 2015

Figure 2: Scatter plots of students' written continuous assessment marks and the number of valid flags they submitted

how well a student's ability to find flags in CTF-style challenges corresponds with their performance in traditional written formative assessment, and also whether it could be possible to replace manual marking of exercises with automatic marking based solely on flag submissions. The design of our framework allows us to perform these analyses.

Figure 2 shows scatter plots of each student's written continuous assessment mark and the number of valid flags they submitted, across all three iterations of the courses. Each point represents a single student; the number of valid flags submitted (on the *y* axis) is weighted for difficulty.

The correlation between the number of valid flags submitted and written continuous assessment mark is high, as shown by Table 1. A correlation coefficient greater than 0.9 indicates a very strong relationship. However, notice that this correlation is not uniform across the entire mark range: if we divide the students into groups of those who achieved the highest grade (i.e., those with a mark of at least 70) and those who achieved lower grades, we see that the correlation is much weaker amongst students achieving the highest grade.

After reviewing students' written submissions, we found that all students who had submitted a valid flag for a question received a good mark in their written submission for that question. The difference between students receiving good and outstanding marks was mainly due to higher-attaining students showing a deeper understanding of the cybersecurity issues involved in the question. For instance, the protocol analysis exercise requires the student to suggest fixes for the protocols, as well as exploiting their implementation and recovering the flags; only students who understood the protocols very well were able to describe adequate fixes, whether or not they had recovered the flags.

	Iteration of course		
	Spring 2014	Autumn 2014	Spring 2015
All marks	0.92	0.84	0.93
Marks ≥ 70	0.37	0.22	0.68
Marks < 70	0.91	0.82	0.90

Table 1: The correlation between the number of valid flags submitted by a student and their written continuous assessment mark for the course

We found that the widest range of marks occurred with the reverse-engineering exercise. The written submissions of all students who submitted valid flags for this exercise showed that they understood the basic concepts of reverse-engineering and were competent with the disassembly tools. The higher-attaining students, however, showed a clear and complete understanding of the operation of the binaries and the reverse-engineering process, whereas the weaker students found vulnerabilities by trial and error. Some of the best answers were provided by students who did not submit flags for all of the other questions in the exercise, meaning that students with fewer valid flag submissions sometimes scored better overall.

The Spring 2015 iteration of the course was simpler than the other iterations: it did not include the reverse-engineering exercise, and questions did not require students to suggest fixes for the vulnerabilities. For this iteration of the course we see a closer correlation between marks awarded for students' written submissions and the number of valid flags they submitted, although the marks of the higher-attaining students still show a lower correlation than those that did not perform as well.

3.1 Discussion

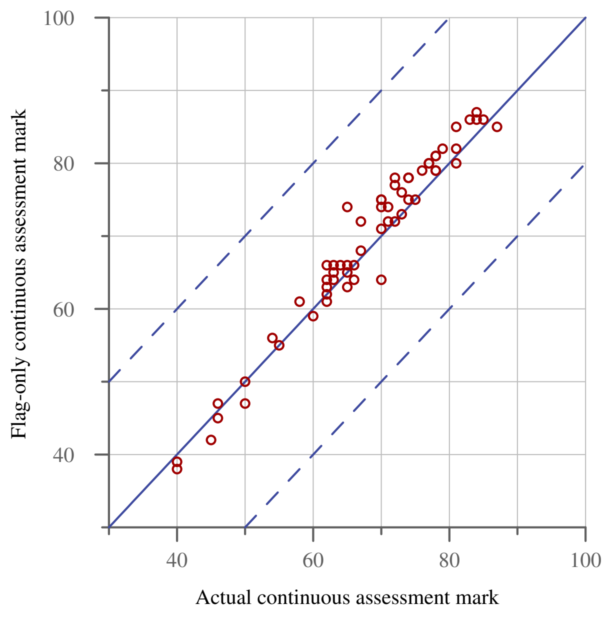


Figure 3: Comparison of Spring 2014 second-year undergraduate students' actual continuous assessment marks and the marks they would have been awarded had they only been assessed on their valid flag submissions

Figure 3 shows a scatter plot of the marks that second-year undergraduate students achieved for the Spring 2014 iteration of the course versus the marks they would have been awarded had they only been assessed on their valid flag submissions. The mean difference in mark is 1.5, and the standard deviation is 2.5. The grade boundaries are positioned at 10-mark intervals between 40 to 70, therefore most students would have received the same overall grade; however, there are some exceptions, particularly around the highest grade boundary, with the maximum difference being 9 marks. The examinations for the other two iterations of the course have not yet taken place.

In summary, our analysis shows a high correlation between a student's ability to successfully complete CTF-style challenges and their continuous assessment marks. This suggests that flag acquisition is a useful assessment technique for academic cybersecurity courses. However, our data also suggests that assessing a student's ability to complete such challenges in isolation is not an appropriate method of assessing whether they possess a deep understanding of cybersecurity topics.

Pedagogical value of written submissions and feedback. An issue that our analysis does not address is the educational value to students of writing down detailed answers to questions and receiving feedback on the answers they give; this process requires students to reflect on what they have done. Higgins et al. [5] provide evidence that students are driven by more than just obtaining high marks, and benefit from feedback that will help them engage with their subject in a deeper way. Flag-only marking would also deprive students of personalised feedback on their work: in every iteration of these courses, either the lecturer or a teaching associate would give detailed feedback on every written submission, something that students indicated was highly valued and a reason for their overall satisfaction with the course (in the end-of-course feedback questionnaires, students rated the feedback they received on average as the third-best amongst the courses offered by the School). However, after assisting students with the initial deployment of the VM, marking the written submissions and providing this feedback was the most labour-intensive aspect of the courses.

Plagiarism. Another (unfortunate) issue raised by flag-only marking is the potential for plagiarism. Across all three iterations of the course, we encountered three cases where groups of students submitted flags containing identical or inconsistent VM identifiers (which are intended to remain constant across the entire life of a single instance of the VM). In the first case, a group of four students submitted the same flag and very similar written answers for a particular question. When interviewed, they admitted copying from each other; they were given a warning and were awarded 0 marks for the entire exercise. In the second case, two students submitted the same flag but different written answers; when interviewed, it was found that they were both using low-spec tablet PCs to attempt the challenges and were sharing a single instance of the VM on an external hard drive. Both students were given words of guidance. In the final case, it emerged that, upon being the first to solve one of the more difficult challenges, a student posted a screenshot of his flag on Facebook to show off to his friends, from where one of his "friends" duly transcribed the flag and submitted it to the flag submission server before him. The students were penalised for their lack of foresight and lack of ethics respectively.

We also encountered a case in which two students submitted flags unique to their own instances of the VM but plagiarised their written answers; both students were penalised; this case would not have been caught had flag-only marking been used. Flag-only marking would also massively increase the reward to students for plagiarising; questioned informally, some students said that there was no benefit to plagiarising flags because of the additional requirement to accompany them with written answers. Overall, we saw much lower rates of plagiarism on our courses than on those with similar amounts of continuous assessment contributing toward the final course mark; we speculate that having to submit flags as well as written answers was a deterrent for plagiarism, as students could no longer simply copy and paste a written answer but had to demonstrate understanding of the written answer by acquiring a flag from their own VM instance too.

4 Conclusion

We have shown that Jeopardy-style CTF challenges can be adapted for an offline VM framework and used as part of the formative assessment for an academic cybersecurity course. Exercises developed as part of this framework were considered difficult by students but were also very popular. Automatic marking of flag submissions complemented by manual marking of detailed written answers provided students with instant feedback while progressing through an exercise, provided them with highly-valued detailed feedback after completing the assessment, improved student satisfaction with the course, and helped course staff to monitor and quickly mitigate incidences of plagiarism.

By comparing the number of valid flags submitted by students and the marks awarded for their written answers, we have shown that the ability of students to acquire flags in CTF-style challenges is highly correlated with their overall marks, and that flag-based marking effectively assesses a student's basic skills and understanding of cybersecurity topics. However, acquiring flags is much less well-correlated with a student's deeper understanding of the underlying issues; flag-only marking may also lead to more widespread plagiarism and fail to satisfy students' expectations of feedback. Therefore, we conclude that basing the formative assessment for an academic cybersecurity course on a combination of automatic flag-based marking with manual marking of detailed written answers provides a highly satisfactory student experience.

We are continuing to extend our framework with additional exercises. Schreuders and Ardern [7] have developed a method of generating individual VMs for each student, which may be an alternative to our startup program. We will also continue our analysis of how well acquiring flags in CTF-style challenges corresponds to traditional educational assessment, with the aim of providing more long-term data.

References

[1] Andreea-Ina Radu and Sam L. Thomas. Organising Monkeys or How to Run a Hacking Club. In *Workshop on Cybersecurity Training & Education (VIBRANT15)* (2015).

[2] Bursztein, E., Gourdin, B., Fabry, C., Bau, J., Rydstedt, G., Bojinov, H., Boneh, D., and Mitchell, J. C. Webseclab Security Education Workbench. In *3rd Workshop on Cyber Security Experimentation and Test (CSET '10)* (2010).

[3] Cheung, R. S., Cohen, J. P., Lo, H. Z., Elia, F., and Carrillo-Marquez, V. Effectiveness of Cybersecurity Competitions. In *Proceedings of the International Conference on Security & Management* (2012).

[4] Davis, A., Leek, T., Zhivich, M., Gwinnup, K., and Leonard, W. The Fun and Future of CTF. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE '14)* (2014).

[5] Higgins, R., Hartley, P., and Skelton, A. The Conscientious Consumer: Reconsidering the role of assessment feedback in student learning. In *Studies in Higher Education* (2002), vol. 27.

[6] Mirkovic, J., and Peterson, P. Class Capture-the-Flag Exercises. In *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE '14)* (2014).

[7] 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).

[8] Vigna, G., Borgolte, K., Corbetta, J., Doupé, A., Fratantonio, Y., Invernizzi, L., Kirat, D., and Shoshitaishvili, Y. 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).