Designing Computer Security Assessments to Reduce Plagiarism*

Rosanne English
University of Strathclyde
Glasgow, Scotland
rosanne.english@strath.ac.uk

ABSTRACT

Plagiarism and other forms of academic dishonesty for computing science assessments is a well documented issue. A common mode of dealing with this is to apply plagiarism detector software to code submissions to check for suspected plagiarism based on how similar submissions are. However, it arguably is less well established how to design computing science specific assessments which aim to reduce the possibility of plagiarism, whilst not disadvantaging students who may struggle with some aspects of an assessment. This paper aims to report on the design and practice of such an assessment within a computer security course.

ACM Reference Format:
Rosanne English. 2019. Designing Computer Security Assessments to Reduce Plagiarism. In *Computing Education Practice (CEP '19), January 9, 2019, Durham, United Kingdom.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3294016.3294020

1 INTRODUCTION

Plagiarism is an aspect of assessment in higher education which we would prefer to avoid. As noted by Sheard *et al.* [10] it can be difficult to formally define plagiarism or cheating. The authors proposed 18 different scenarios of inappropriate behaviour such as "two students collaborating on an assignment meant to be completed individually" and "hiring someone to sit an exam for you". Whilst these are obviously not the only methods of questionable behaviour, students often report a range of reasons for such academic dishonesty including time pressures, work being too challenging and the desire to help a friend ([4],[2]).

The matter is further complicated for lecturers in dealing with such behaviour after the fact as this typically includes tasks such as determining how sure the lecturer is that it is indeed plagiarism, and how to report it [6]. Often it may be easier to turn a blind eye to the suspected plagiarism.

However, in all this it is possible to consider a positive side. Is there a way to construct an assessment which discourages plagiarism, whilst simultaneously not disadvantaging students who are unable to complete part of it? This paper aims to present such an approach for a computing science security assessment. The contributions of this paper are as follows:

(1) defines a practice for designing computing science coursework to minimise potential academic dishonesty but not disadvantage students who may struggle with such technical aspects
(2) discusses the experiences and challenges of offering such a practice

2 BACKGROUND

As noted by Harris, plagiarism is a difficult topic to address as it can elicit emotional responses when raised. Harris argues that students not only cheat themselves of an important aspect of their education, but also artificially inflate grades resulting in students who conform to academic integrity rules being disadvantaged by an apparently lower result [5]. In order to address academic dishonesty, the lecturer must first be highly confident that it has occurred.

Roberts argues cheating in computing science is particularly prevalent due to the availability of existing solutions such as those available online [7]. Students can adapt this code, and manipulate it to fit the assessment. Alternatively there are websites which facilitate employing coders to complete an assignment for students for a fee - for example rent-acoder.com.

The particular issue in computing science disciplines is further evidenced by Sheard and Dick who reported the results of a survey of computing science students where students were asked to self-identify scenarios of cheating they had employed, and how acceptable they deemed this [9]. The survey was delivered twice, once in 2000 and once again in 2010. The results showed that in 2000 78% of students reported employing at least one form of the cheating behaviours identified, and in 2010 this reduced to 63%. Whilst this appears to be a substantial decrease, a rate of 63% is higher than one would like.

Another example of students self-identifying academically dishonest behaviours is presented by Selwyn. In Selwyn's work, a questionnaire was issued to 1222 students in the UK, mostly from the University of Cardiff, with 4% of these students in a computer sciences or mathematical discipline. Selwyn's analysis of the results determined that those in a computing sciences or mathematical discipline were significantly more likely to copy a few sentences or paragraphs for an assignment [8]. Selwyn goes on to argue that this is likely due to the nature of such assessments being coding assessments which are easily copied from existing online sources.

Vamplew *et al.* propose that approaches to reducing plagiarism can be seen as belonging to one of two categories - plagiarism prevention and plagiarism detection.In a similar argument, Dick

*Produces the permission block, and copyright information

CEP '19, January 9, 2019, Durham, United Kingdom

ACM ISBN 978-1-4503-6631-1/19/01…$15.00
https://doi.org/10.1145/3294016.3294020

et al. identifies the first stage of plagiarism from an academic's standpoint is to preempt it [3]. Dick *et al.* also suggest a range of approaches to assist an academic in tackling plagiarism. One such approach is to provide a different task to each team in a class. This has an advantage that each team is doing something different and so there's less chance of between-team collusion. However, this can be a significant amount of effort to detail a potentially large number of alternate assessments.

Despite this apparent proficiency in computing science plagiarism, Barrett *et. al* [1] note that much plagiarism literature in education is particularly relevant to essays and research projects. They argue that there are aspects which are specific to computing science and produce a guide for staff aiming to reduce plagiarism and collusion on computing science course assessments. This includes aspects such as having a variable which can be changed from year to year, and allowing students some choice in their direction as well as incorporating a reflective aspect of the assessment. This could take the form of a presentation or essay to reduce collusion in a large class with many students completing the same assessment [1].

However, these approaches are often reported with a lack of specific associated practice examples, and often minimise the effort required on the lecturer's part to employ such techniques. This paper aims to address this imbalance by presenting the practice of designing an assessment for a computer security class which employs many of these suggestions to minimise the possibility of academic misconduct, whilst also designing for effectual re-use in future assessment.

3 CONTEXT

This assessment was designed for a 4th year honours class in Computer Security, which currently has 98 students enrolled. The class is mandatory for over 90% of students in attendance due to British Computer Society accreditation requirements which require computer security as part of the curriculum. Students participating in the class have three or four prior years experience in programming. Those with four years have completed their first three years in the University before completing a one year industrial placement. In their studies, little emphasis has been placed on computer security until this class. The class has been designed with active learning opportunities in mind. Students are provided with short mini-lecture videos covering the class content. Students are expected to view a number of these videos prior to a class and are then asked to complete problem sheets and problem-based learning tasks in class. The problem sheets are intended to be completed individually and focus on ensuring students have understood the material.

The problem-based learning tasks are scenario based, and students are asked to work in self-assigned assessment teams of 3 or 4 to discuss the scenario and propose solutions for the suggested problem. By asking students to work in their teams, it is hoped to encourage the group to progress through the developmental sequence of small teams proposed by Tuckman [11] (forming, storming, norming, performing) during class, so they can better perform as a team in the assessment.

4 STRUCTURE

The assessment is structured as a treasure hunt, where students are presented with clues and have to solve the clue by writing a program which explores an aspect of computer security such as steganography (hiding information in plain sight, such as a digital image) or cryptography. Each clue, once solved, provides the clue to the next stage. There are three stages in total. The first is a digital image which has a text file within it. The text within the file is encrypted with a traditional cipher. Once solved, the cipher directs students to e-mail the lecturer to get the details of the final stage. The final stage involves students researching how to use the Java cryptography extension and Java cryptography architecture to decrypt a message encrypted with RSA. They are provided a new text file which has been encrypted it with the lecturer's private key. They are also provided a copy of the lecturer's corresponding public key. Students must write a Java program which constructs a PublicKey object to be used with RSA to decrypt the encrypted text file. The final output is a location, which indicates the location of the fictional buried treasure. Each stage is submitted incrementally. Upon submission of a stage, both the process and the output are assessed. That is a student can gain 50% of their marks for that part of the assessment if they get the correct output and the remaining 50% is based on the process they followed. This means it is possible for students to not get the correct output, but still have followed a correct process. For example with the steganography stage this could be exemplified by a group following the least significant bit algorithm for extraction, but resulting in the wrong file by perhaps putting the bits back together as bytes in an incorrect order. Alternatively, a group could get the correct output by a flawed process such as using Strings to put together the bytes rather than bit operations. If a team does not successfully complete a stage, it is important that they are not disadvantaged for the remaining stages. The output of a stage is released to those who are not successful shortly after submission. Also included is feedback on how the process could have been completed successfully.

From a lecturer's perspective, the output of each stage is comparatively undemanding to mark. Take for instance the first stage, the output of which is a text file. The file needs to be compared byte by byte to ensure it is precisely the same as the expected output. This can be achieved using hashing software or even by simple visual inspection of the size and content of the text file.

The second aspect to be considered for marking is the process itself. This can be achieved by examining the submitted code to ensure students have followed the expected process. For example with the steganography exercise the lecturer can look for methods which extract the least significant bit, ignoring header information in the image file, and for code which combines the least significant bits into bytes to be written out as a file.

Note that the code itself does not need to be compiled and run unless the lecturer is unconvinced of its correctness. From experience, running exercises where the lecturer does have to compile and run such code introduces an increased time element in marking as students can misinterpret instructions, submit the wrong version of code which doesn't compile, use packages or similar structures which makes running it on a different environment less straight-forward.

Each stage has the same marking rubric which is released to students when the exercise specification is released. This is shown in Table 1. This has the benefit of making the assessment criteria clear to students, and also being consistent, and efficient to mark for lecturers.

Each stage is worth 4% of the final class mark, thus totalling 12% for all three stages. The final part of the assessment takes the form of a video presentation which reflects on the processes and lessons learned during the treasure hunt. This is used as a tool to determine students ability to understand the material and reflect upon it. This is worth the largest percentage of the overall coursework, totalling 18%.

A key aspect of the design is that each team is given a different set of clues, though they are all structured in a similar fashion. For the first stage, each team receives a different stegoimage, within which a different secret message is hidden using a different substitution cipher. To ensure all teams had a message of the same length, the message was altered so that each team had a code word to send to the lecturer by e-mail. All code words were of the same length, meaning all resulting payloads were the same size. For example one such message was "send rose an email with your team name and the code word spiral".

This means that if the lecturer receives an email from a team which doesn't use their assigned word, then they are likely to have plagiarised. This is of course not fool proof, it is possible another team provides their code to extract the text file, or crack the code and this could potentially work for the other team but this adds in an additional element of distinction for the students work.

The production of different stegoimages and substitution ciphers is convenient to automate using purpose built code, only marginally adapted from constructing one such exercise.

By adjusting the clues for each team, this introduced an element of variability, which is adapted slightly from Dick *et al.*'s suggestion of a different task for each team [3] and implements one of Barrett *et al.*'s proposal of the same task with different variables [1].

It should be noted that the overall process for solving each stage will be similar irrespective of which team the student is in. However by having a different clue at each stage you introduce an element of distinction between teams.

5 EXPERIENCE

Delivering this assessment with a class of 98 4th year students it was clear the approach introduced an element of uncertainty with the students as to whether they had different clues to solve. This was evidenced by a number of conversations with students around this topic.

Students were not initially told what form of steganography had been used, nor what the content of the text file included. This caused a degree of confusion and frustration in students. There were many questions around the topic asking if the bits of the payload had been hidden randomly, or in a different order to the typical top left to bottom right.

Students also got side-tracked by the concept of the payload being a text file, with students asking about the encoding of the text in the file. They were encouraged not to think of it as a text file, but as a stream of bits.

There was a desire from many teams to know the structure of the text so they could be certain they had the right output before final submission. It is believed from speaking to students that these concerns arose from the fact that there is no easily adapted existing code for steganography extraction online. This appears to cause a degree of anxiety in students, as it goes against their experience of coursework to this point.

Whilst this may be uncomfortable for some students, the task itself is achievable by implementing the algorithm using basic Java and bit manipulation which is presented in a mini-lecture video. In a class of 98 students, the video was accessed 2761 times in the two weeks that the first stage of the assessment was completed in. This could additionally support the idea that an answer for this is not readily available to students from existing web sources.

In an attempt to reduce student anxiety it was made clear that the least significant bit algorithm had been used, and the content of the text file was identified as being English language letters. Despite this, some teams still struggled with this aspect with 6 out of 26 teams e-mailing for last minute help in the working day before the assessment was due.

In the second and third stage, there was a little less anxiety as the overall process became more familiar to students. The second stage is also something for which there is more code commonly available online. However, for the maximum available marks for process students must fully automate detection of a the correct decrypted (English) text. A number of techniques are examined in the mini lectures to explore this, but a limited number were employed in the submissions which instead focused on more brute force approaches with manual intervention to detect an appropriate decryption.

The final submission of video presentations contributed the majority of the marks for the assessment. The criteria examined here include the understanding of security techniques presented, reflections on lessons learned, presentation quality, and team performance.

The use of a narrative in addition to the technical output of the stages resulted in a deeper insight into students understanding of the material. Students who had made small adjustments to code until it eventually worked clearly demonstrated a limited understanding of the material which would not have been apparent from the technical part alone. In contrast those who ensured they understood the material before designing the solution demonstrated a deeper understanding.

Due to the structure of the assessment, even if students managed to plagiarise for the technical parts, the larger proportion is for the reflective presentation which is more difficult to do particularly if the technical parts had been plagiarised.

6 DISCUSSION

The assessment of this class has changed considerably from previous years where individual students were presented with one or two large, unconnected technical exercises to complete. These exercises were typically performed well, and their structure was familiar to students but it was apparent from subsequent assessment in exams that students had managed to 'hack' at the code until they got something which was working instead of developing a good understanding of the material. It was suspected that this was due

Criterion	0 points	1 point	2 points	3 points
Stage Output	No output submitted	The output submission has small elements reflective of the expected output, but there are minor errors in the output.	The output submission is nearly precisely what is expected, perhaps one or two minor errors.	The output submitted perfectly matches the expected output. No errors at all.
Process followed to achieve output	No attempt	The process followed was satisfactory, it was a somewhat appropriate approach to solving this part of the treasure hunt. It is likely to have a number of errors in implementation and/or in the approach itself	The process followed was good, it was an appropriate approach to solving this part of the treasure hunt but was perhaps limited in places, or implemented incorrectly	The process followed was excellent, it was an entirely appropriate approach to solving this part of the treasure hunt

Table 1

to students getting existing code snippets from online sources, and from over collusion.

By introducing an assessment which had links between them, and a variation on elements of the assessment this encouraged teams to work more independently of existing sources. However, this did take a longer time to construct as an assessment. It also resulted in an element of anxiety for the students, which was then reflected onto the lecturer.

The incorporated aspect of a reflective presentation also helped to better highlight students who had more deeply understood the material as they were clearly able to articulate their approaches, the relevant technical material, and reflect on the process.

If a lecturer wished to adopt this approach I believe it would be possible to adapt by swapping in a range of different tasks for each of the stages, such as attacking an insecure website or a different range of encryption methods.

The code to generate different versions of the clues for each stage could be built over time, with small aspects being the variable such as the cover image, the code word, and the encryption key.

In future I plan to continue with this approach, building up a collection of tasks to use for each stage. This will mean that each year I can alternate the assessment with comparative ease. I would make it clearer to students earlier what the expected output will look like for the first stage beyond being a text file, as this caused a great deal of anxiety for students and tension in the class.

I will also make it clearer to students that the code should all be written using default Java packages as some teams used external packages which made their implementations unnecessarily complex.

Initially I had considered that students would like the assessment as it was structured like a treasure hunt, however the structure forces students to engage in the course much earlier and submit coursework more often than they are typically used to. I was not prepared for the small number of students who found this particularly challenging, and hope that by disseminating my experiences of this another lecturer could minimise the impact of this by providing more scaffolding for those students such as for example 'buying a clue' where teams could take a small reduction in their mark to get a clue which would help them in their completion of the assessment. It could also be possible to change the structure such that students are given the option to submit incrementally, or to do it all at the end of the course. I think this would allow students the opportunity to have more control over the timing of their assessment and could help settle some anxieties over this.

REFERENCES

[1] R Barrett, A L Cox, J Malcolm, and C Lyon. 2006. Plagiarism prevention is discipline specific: a view from Computer Science. *Journal for the Enhancement of Learning and Teaching* https://uh, 1 (2006), 48–56.

[2] Stephen F Davis and H Wayne Ludvigson. 1993. Additional Data on Academic Dishonesty. 1992 (1993), 79–81.

[3] Martin Dick, Judy Sheard, Cathy Bareiss, Janet Carter, Donald Joyce, Trevor Harding, and Cary Laxer. 2003. Addressing student cheating. *ACM SIGCSE Bulletin* 35, 2 (2003), 172. https://doi.org/10.1145/782941.783000

[4] Arlene Franklyn-Stokes and Stephen E. Newstead. 1995. Undergraduate Cheating: Who does what and why? *Studies in Higher Education* 20, 2 (1995), 159–172. https://doi.org/10.1080/03075079512331381673

[5] James K. Harris. 1994. Plagiarism in computer science courses. *Proceedings of the conference on Ethics in the computer age* - (1994), 133–135. https://doi.org/10.1145/199544.199601

[6] Chris Park. 2004. Rebels without a clause: towards an institutional framework for dealing with plagiarism by students. *Journal of Further and Higher Education* 28, 3 (2004), 291–306. https://doi.org/10.1080/0309877042000241760

[7] E Roberts. 2002. Strategies for promoting academic integrity in CS courses. *Frontiers in Education, 2002. FIE 2002. 32nd Annual* 2 (2002), F3G–14–F3G–19 vol.2. https://doi.org/10.1109/FIE.2002.1158209

[8] Neil Selwyn. 2008. 'Not necessarily a bad thing ...': A study of online plagiarism amongst undergraduate students. *Assessment and Evaluation in Higher Education* 33, 5 (2008), 465–479. https://doi.org/10.1080/02602930701563104

[9] Judy Sheard and Martin Dick. 2011. Directions and Dimensions in Managing Cheating and Plagiarism of IT Students. (2011), 177–186.

[10] Judy Sheard, Selby Markham, and Martin Dick. 2003. Investigating Differences in Cheating Behaviours of IT Undergraduate and Graduate Students: The maturity and motivation factors. *Higher Education Research & Development* 22, 1 (2003), 91–108. https://doi.org/10.1080/0729436032000056526

[11] Bruce W. Tuckman. 1965. Developmental sequence in small groups. *Psychological Bulletin* 63, 6 (1965), 384–399. https://doi.org/10.1037/h0022100 arXiv:arXiv:1011.1669v3