Teaching Security Notions in Entry-Level Programming Courses

Vahab Pournaghshband
Computer Science Department
University of San Francisco
San Francisco, USA
vahab.p@usfca.edu

Hassan Pournaghshband
Software Engineering Department
Kennesaw State University
Atlanta, USA
hpournag@kennesaw.edu

***Abstract*—Nowadays, educating computer science students on security issues is a reality. Offering computer security related courses has significantly increased during the past decade. In fact, offering these courses has become a standard part of the curriculum for many computing disciplines. While there are many proposals suggesting addition of this appealing topic into the CS introductory courses, many faculties do not support the idea. They rightfully claim that these courses are already packed with basic programming concepts leaving not much room for other topics. In this study, we show how basic security concepts can be incorporated into these courses without increasing the normally required efforts needed by students as well as the instructor. The idea behind our approach is to utilize a programming assignment with this matter in mind. While students will work on the assignment as usual, utilizing their understanding and knowledge of conventional computing concepts, they will also be implicitly introduced to major security concepts and practices through the process, at different steps. In this paper, we discuss the necessary steps for our approach, and explore what security concepts should be contemplated, along with how to utilize these concepts during the process.**

***Index Terms*—computer security, security mindset, CS1**

I. Introduction

While software has become a crucial element in our lives in recent decades, hackers, in parallel, have also become more influential and capable of interrupting these technological connections by breaching software security. For instance, In the past year, over 500,000 Zoom account credentials were hacked and made available on the Dark Web [1]. In another instance, in the same year, a critical security flaw in WhatsApp was exploited, enabling hackers to install surveillance software on users' smartphones. This incident may have impacted WhatsApp's 1.5 billion users [2]. Furthermore, more recently, the Colonial Pipeline Company, the largest petroleum pipeline in the US, was breached which consequently led to the preventive shutdown and halting of its services to millions of users for nearly a week [3]. More generally, on average, an attack was reported to a connected computer globally for every 39 seconds in 2017 [4]. In 2019, a data breach cost around $3.92 million on average and it has been forecasted that worldwide spending on cybersecurity will reach $134 billion in 2022 [5], [6]. New threats are further emerging as computers become more embedded and play more intimately a big role into our environment and daily lives, as for the recent security vulnerabilities found in mobile medical devices [7]. For these reasons, in our current technological world, the inherent problems of computer security are becoming increasingly important. Hence, it is necessary that computing students gain the necessary knowledge for handling security problems, early in their academic programs [8].

978-1-6654-3687-8/21$31.00

The need for incorporating computer security education into the computing curriculum is widely recognized [8]–[12]. Researchers and educators believe computer security topics should be tightly integrated into computing education [13], [14], rather than being an afterthought [15]. In fact, the Computer Science Curriculum review taskforce identified the emergence of security as a major area of concern. The taskforce further reported that it has received substantial attention to security matters to be compulsory for all computing graduates [16].

Proper integration of computer security into introductory programming courses can motivate and engage students, and build excitement and enthusiasm for computing discipline, as well as accomplishing anticipated student learning outcomes. Most of the existing research in this area is based on pioneering exploratory projects by faculty members with expertise in computer security. For general faculty members teaching introductory CS courses, few of which have computer security backgrounds, it can be challenging to take advantage of these results. Additionally, these instructors often complain that the courses are already packed with general programming concepts and adding other topics is not viable. Our proposed approach is specifically addressing these concerns. We show how elementary security concepts can be taught in these courses, and yet keep the course load manageable for both, students and instructors. This is done by presenting a step-by-step process to develop a self-contained programming project for the course. While students apply their basic programming concepts working on the project, they also become aware of computer security-related programming issues and concepts.

We have successfully employed our approach in an introductory one-semester course that was developed and taught over a two year period at our institution. As presented in Section 5, we are pleased with the results of our approach and have found it promising.

This paper is organized as follows: Section II presents the underlying principles, followed by desirable characteristics of the project in Section III. Methodology, assessments, and conclusions are presented in Sections IV, V, and VI, respectively.

II. Underlying Principles

In this section we discuss the significance of teaching security notions in foundational courses, and yet keeping the primary focus on teaching basic programming concepts.

1) An early exposure to security issues is essential to a students' foundational appreciation and understanding of computer security. This helps students later when they take upper level courses covering security at a large stage.
2) Many students outside of the CS major are required to take only foundational CS courses, and quite possibly will never take any other CS courses beyond the introductory courses. Therefore, it might be their only chance in their study to learn about computer security. Even if these students never pursue a programming career, this would be a valuable opportunity for them to learn about the technical aspects of computer security through understanding the root cause of known security problems covered in this course. For instance, students will understand the importance of choosing a strong password once they learn how powerful brute force attacks (included in our example project) are.
3) It is shown that real-world programming projects play a significant role in actively engaging and motivating students. Incorporating security issues into the project would make it even more interesting for them [17].

III. Desirable Characteristics of the Project

1) Due to the fact that institutions use different language choice incorporated in their CS introductory courses, the project should be language independent so it can be adoptable to any of those CS courses.
2) The rubric of the project should be easy to follow for easy grading purpose. This is to ensure that it is scalable to large classroom settings.
3) The project should be self-contained and accessible to all students. That is, it should provide necessary background in computer security, describing security concepts in their simplest terms. In other words, it should not assume or require any prior specialized knowledge of computer security. This is particularly important, in regard to minimizing the instructor's intervention. The assignment specification should provide additional resources and materials for those who are interested to learn more about the topic.

In addition to the characteristics mentioned above, our proposed project should have essential characteristics of any CS assignments including:

4. The project should provide students with first-rate homework assignments. This is integral to the success of any course, and nowhere is this relationship so pronounced as in CS1/CS2 [18].
5. The project should incorporate advanced CS topics into an introductory course so it can provide motivation. Yet care must be taken to avoid overwhelming students with more than what they can normally accomplish.
6. The project must engage students with their learning. With the instructors taking the care to make assignments enticing and interesting, students look forward to working on their programs and eager to get their code to work. As a result, students achieve far greater results than they otherwise would [19].

IV. Methodology — The Process

In this section we present a five-step process that instructors can follow to develop a fitting project if they wish to include basic security concepts in their CS introductory courses. Below we have listed those five steps followed by an explanation of each step.

1) Stimulate the students' interest in security through a simple attack on a security-sensitive system.
2) Present elementary terminologies and definitions involved in the attack scenario given in Step 1.
3) Explain the significance and root causes of the problem (i.e., the inherent vulnerability of the system to certain attacks) and lead the students to potential defensive solutions to eliminate or mitigate such attacks.
4) Present students with the project specification to implement a defense mechanism.
5) Offer students with potential programming mistakes that typically can be made at this point, which will make the defense solution ineffective.

To further clarify each of the above steps, here, we use examples in relation to a project specification sample that we have used in our introductory courses.

1) The students are presented with a simple authentication system (login program). In the attack-the-system phase, we present the students the password cracker, where they are asked to implement a program that guesses the correct password and measures the time it takes to guess the password in the worst-case scenario by enumerating every single password combination. The password in this example is an integer type, so the password space is relatively small. Hence, the correct password is expected to be identified within seconds in a typical machine. Figure 1 lays out the details of the attack component of the project, as presented to the students in the project specification.
2) The students are introduced to terminologies and definitions related to Step 1. In the case of the login program, they are brute force attacks and authentication systems.
3) In the project specification, we explain to the students why it was relatively trivial to attack this system. This is due to the password space being trivially small. Therefore, a natural way to increase the size of the

Fig. 1. Password Cracker as presented in the project specification.

Password Cracker

In this component of the project, you are to write a password cracker for login program. Your password cracker performs the brute force attack to guess the correct password, hence trying all possible combinations. In the process, we will also measure how long it takes for the password cracker to find the correct password, as well as measuring how long it takes to try all possible combinations. We will do all the following steps for each of the three types: `unsigned short`, `unsigned int`, and `unsigned long`. The steps below, however, uses only `unsigned int` as an example.

1) Declare a `const` variable, `unsigned int correct_password` to store the correct password and initialize it to your choice.
2) In order to check all combinations we need a loop that iterates through all possible values. Our `for` loop requires a lower bound and an upper bound for the range of values it will iterate through, incrementing the loop counter at each step.
   - What is the lower bound for this type?
   - What is the upper bound for this type? In other words, what is the largest possible value an `unsigned int` can hold?
   - Your loop should stop as soon as the correct password is found.
3) In order to measure how long it takes for our program to guess the password correctly (or to measure the time it takes to execute any block of code!), we will use the `clock()` function defined in `time.h`. Here is how to use the `clock()` function in C:

```
#include <time.h>
clock_t start, end;

start = clock();
/* Your code goes here */
end = clock();

float seconds = (float)(end - start) / CLOCKS_PER_SEC;
printf ("It took %f to complete this.", seconds);
```

4) Lastly, consider your current program:
   - For what password, your program takes the longest to guess?
   - Set the correct password to this password, and measure how long it takes for your password cracker to find the correct password. This is what we call the *worst-case time*, or *upper bound*.

password space is to enforce the passwords to be a combination of various characters as opposed to be only containing digits. At this point, the students, since they measure the time to crack weak passwords only containing digits, understand that increasing the password space will require more resources (e.g., time) for the attacker to crack it.

4) In this step, the students are asked to implement a defense mechanism to defend or mitigate brute force attacks in authentication systems. In this project example, the students are asked to implement a password strength meter that requires the users to choose “strong" passwords, which must be sufficiently long and must be combinations of other certain characters. Additionally, the students are also asked to implement a default password generator that generates a temporary strong random password when the user sent requests to reset their password. The project specification explains to students why default passwords must also be strong. That is because the system should not assume the users will change the default passwords very soon, or change them at all. Leaving the system with weak default passwords creates an attack window of opportunity.
5) We explained (through specific examples) to students what potential mistakes in implementing the password strength meter correctly, would mean in a security context. For instance, a failure to properly check if the password is at least 8 characters long, leads to accepting one-character long passwords as strong, making cracking such passwords a trivial task. It is crucial to emphasize the security implications of programming mistakes in writing security-sensitive applications.

V. Assessments

We have employed our approach in this course at our institution for three semesters with a total of 83 students. We

TABLE I
ILLUSTRATION OF THE PERCENTAGE OF STUDENTS WHOSE RESPONSES CHANGED FROM "NO" TO "YES" FOR EACH QUESTION (Q) IN EACH CONSTRUCT (C) WHERE THE NUMBER OF STUDENTS IS 83.

C: Students retention in CS	
Q: Are you planning to take CS2?	%7
C: Students' interest in pursuing subject matter beyond the project	
Q: Do you know what a cryptographic one-way hash function is?†	%13
Q: Are you willing to take a computer security class as your upper-division elective?	%24
C: Impact outside of classroom	
Q: Have you changed your primary e-mail or social network website account password in the past two months?	%16

† Included in the project specification, we provided advanced relevant security topics as additional resources labeled as *optional readings for interested readers*. In this additional optional reading resources for interested readers, we provided information on how and why only the hashed digest of passwords are stored. By asking if the students know what cryptographic hash functions are (without prior knowledge prior to taking the course), we determine if the student was interested in pursuing the security topics further.

have assessed the students' responses to our several Yes/No questions, and found the results promising. These results are summarized in Table I.

We plan, for our future work, to have a robust assessment of our approach measuring parameters such as the amount learned by our students from the project, and if in fact, completing it truly thrilled them.

VI. CONCLUSIONS

Using security-focused assignments in foundational programming classes has allowed instructors to increase student awareness about security issues at the beginning of their education experience rather than introducing it during specialized courses at higher levels. In this paper, we presented a project-based approach for effectively achieving this goal. We showed that if instructors wishing to include these security concepts into their introductory courses follow our step-by-step process, they can attain this goal without putting additional burden, than is normally required, on their students as well as own shoulders. We presented a self-contained approach that is scalable and adoptable to any CS introductory sequence course in any institution. Our preliminary assessments, while making them security-conscious, shows an increase in students engagement and excitement in completing assignments in an introductory course.

REFERENCES

[1] Forbes.com, "Hacked zoom accounts given away for free on the dark web," https://www.forbes.com/sites/leemathews/2020/04/13/500000-hacked-zoom-accounts-given-away-for-free-on-the-dark-web/6e67407a58c5, April 2020.

[2] B. Technology, "Whatsapp discovers 'targeted' surveillance attack," https://www.bbc.com/news/technology-48262681, May 2019.

[3] CNN, "Colonial pipeline launches restart after six-day shutdown," https://edition.cnn.com/2021/05/12/business/colonial-pipeline-restart/index.html, May 2021.

[4] T. S. Magazine, "Hackers attack every 39 seconds," https://www.securitymagazine.com/articles/87787-hackers-attack-every-39-seconds, February 2017.

[5] DataStealth, "110 must-know cybersecurity statistics for 2020," https://www.datex.ca/blog/110-must-know-cybersecurity-statistics-for-2020, September 2020.

[6] Statista, "Number of compromised data records in selected data breaches as of january 2021," https://www.statista.com/statistics/290525/cyber-crime-biggest-online-data-breaches-worldwide/, January 2021.

[7] V. Pournaghshband, M. Sarrafzadeh, and P. L. Reiher, "Securing legacy mobile medical devices," in *3rd International Conf. on Wireless Mobile Communication and Healthcare*, ser. MobiHealth '12. Springer, 2012.

[8] V. Pournaghshband, "Teaching the security mindset to CS1 students," in *Proceeding of the 44th ACM technical symposium on computer science education*, ser. SIGCSE '13. ACM, 2013. [Online]. Available: http://doi.acm.org/10.1145/2445196.2445299

[9] V. Švábenský, J. Vykopal, and P. Čeleda, "What are cybersecurity education papers about? a systematic literature review of sigcse and iticse conferences," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20, 2020, p. 2–8.

[10] T. Hynninen, "On the learning activities and outcomes of an information security course," in *Proceedings of the 19th Koli Calling International Conf. on Computing Education Research*, 2019.

[11] T. Dimkov, W. Pieters, and P. Hartel, "Training students to steal: a practical assignment in computer security education," in *Proceedings of the 42nd ACM technical symposium on computer science education*. ACM, 2011, pp. 21–26.

[12] D. Basu, H. K. Kumar, V. K. Lohani, N. D. Barnette, G. Back, D. McPherson, C. J. Ribbens, and P. E. Plassmann, "Integration and evaluation of spiral theory based cybersecurity modules into core computer science and engineering courses," ser. SIGCSE '20. Association for Computing Machinery, 2020, p. 9–15.

[13] S. Peltsverger and O. Karam, "Is teaching with security in mind working?" in *2010 Information Security Curriculum Development Conference*, ser. InfoSecCD '10. Association for Computing Machinery, 2010, p. 15–20.

[14] C. Curricula, "The joint task force on computing curricula," *IEEE Computer Society Association for Computing Machinery*, 2017.

[15] J. McManus, "Security by design: Teaching secure software design and development techniques," *J. Comput. Sci. Coll.*, vol. 33, no. 3, p. 75–82, Jan. 2018.

[16] "Computer science curricula 2013," https://acm.org/binaries/content/assets/education/cs2013_web_final.pdf.

[17] R. L. Fanelli and T. J. O, Connor, "Experiences with practice-focused undergraduate security education," in *Proceedings of the 3rd International Conference on Cyber Security Experimentation and Test*, ser. CSET'10. USA: USENIX Association, 2010, p. 1–8.

[18] T. J. Feldman and J. D. Zelenski, "The quest for excellence in designing cs1/cs2 assignments," in *In Proc. of the 27th Symposium on Computer Science Education*, ser. SIGCSE '96, 1996, p. 319–323.

[19] E. S. Roberts, *The Art and Science of C: A Library Based Introduction to Computer Science*, 1st ed., ser. 10. Addison-Wesley, 9 1994.