How Do Students Feel About Automated Security Static Analysis Exercises?

Akond Rahman* Hossain Shahriar† Dibyendu Brinto Bose‡
*Department of Computer Science, Tennessee Technological University, Cookeville, TN, USA
†College of Computing and Software Engineering, Kennessaw State University, Kennessaw, Georgia, USA
‡Reeve Systems, Dhaka, Bangladesh
Email: *arahman@tntech.edu †hshahria2012@gmail.com ‡brintodibyendu@gmail.com

***Abstract*—This Innovative Practice, work in progress (WIP) paper presents our experience related to two exercises that focus on automated security static analysis, a practice used to integrate security into development and operations (DevOps). The concept has gained popularity amongst information technology (IT) organizations. However, security-related concerns, such as security weaknesses in DevOps artifacts can cause serious consequences. Our preliminary findings indicate that (i) students positively perceive the introduced exercises; and (ii) the students perform well if they are provided necessary background on the exercises. Our WIP paper lays the groundwork to build course materials that will facilitate development, deployment, and dissemination of DevOps-related education materials that also incorporate cybersecurity concepts.**

***Index Terms*—computer science, cybersecurity, devops, devsecops, education, exercise, experience report, student perception**

I. Introduction

The concept of development operations (DevOps) have gained a lot of interest amongst information technology (IT) organizations. DevOps advocates for automation for both development and operations teams so that typical software development activities, such as development, deployment, and testing of software can be done seamlessly without involving manual efforts [1]. Practitioners use a wide range of practices, such as automated deployment [2], automated testing [2], and infrastructure as code (IaC) [3] to implement DevOps in their organizations. Adoption of DevOps has resulted in benefits for IT organizations. For example according to an Atlassian survey [1], 61% of the surveyed 500 practitioners, reported DevOps to help "*them produce higher quality deliverables*". As another example, a Longitude survey [2] with 400 IT professionals reports that DevOps have helped them to reduce the number of system outages.

Despite reported benefits, security weaknesses remain a concern for DevOps implementation. A survey conducted by the SANS institute [3] found 46% of practitioners to neglect cybersecurity in their implementation of DevOps, which can introduce security weaknesses in artifacts used for DevOps. Researchers have documented the presence of security weakness in artifacts used to implement DevOps, for example, Rahman et al. [4], [5] reported that IaC scripts that are used to automate configuration management are susceptible to security weaknesses, such as including hard-coded password and using weak cryptography algorithms. Rahman et al. [4] advocated for increasing cybersecurity-related awareness amongst practitioners who use IaC so that security weaknesses are mitigated early in the development stage. Industry experts too have discussed the importance of integrating cybersecurity into DevOps, and advocated for a 'shift left policy', which will integrate cybersecurity concepts early into the development process [4].

Our above-mentioned discussion highlights the need for educating students about integrating cybersecurity concepts, such as automated security static analysis into the software engineering curriculum. The first author of the paper instructed a graduate course titled 'Secure Software Development', where two exercises were dedicated to showcase the value of automated security static analysis in the context of DevOps. We present an experience report of conducting the two exercises in this work in progress (WIP) paper. The exercises are related to automated security static analysis, a practice used by IT organizations to integrate security into DevOps [2]. Our reported experience can be helpful for other educators who want to adopt automated security static analysis into prospective courses. Furthermore, our WIP paper can provide clues for researchers on how to better integrate DevOps-related concepts into the curriculum of software engineering and cybersecurity.

We answer the following research questions:

- ***RQ1**: What is the performance of students in exercises related to automated security static analysis?*
- ***RQ2**: What are the perceptions of students for exercises related to automated security static analysis?*

We conduct an empirical study to evaluate the effectiveness of automated security static analysis exercises. We answer the research questions by analyzing grade books and survey results collected from a graduate course titled 'Secure Software Development', which was introduced for the first time at Tennessee Tech University (TnTech). To synthesize students' perceptions we apply open coding [6], a qualitative analysis

[1]https://www.atlassian.com/whitepapers/devops-survey-2020
[2]https://www.businesswire.com/news/home/20200630005260/en/New-Relic-Research-Uncovers-Gaps-Software-Leaders
[3]https://www.sans.org/reading-room/whitepapers/analyst/membership/38690
[4]https://cloud.google.com/solutions/devops/devops-tech-shifting-left-on-security

TABLE I: Students' Experience in Cybersecurity and Software Engineering

Experience	Respondents (Cyber)	Respondents (Soft. Eng.)
< 1 year	8	4
$1 - 2$ years	2	1
$3 - 4$ years	2	4
> 4 years	0	3

technique to generate high-level categories from text input. Prior to conducting the survey and analysis we obtain Internal Review Board (IRB) approval from TnTech.

Our contribution is a list of student perceptions related to automated security static analysis exercises.

II. Overview of the Course and Exercises

The course is titled 'Secure Software Development', which was introduced in the graduate curriculum in the Department of Computer Science (CS) at TnTech for the first time in Fall 2020. The pre-requisite of this course for students was to be enrolled as a graduate student in the Department of CS. Prior to conducting the course, the syllabus was shared amongst all graduate students through e-mails in April 2020. A total of 12 students enrolled in the course. The instructor of the course conducted an initial survey of students' experience in software engineering and cybersecurity. The students' reported academic and professional experience in software engineering and cybersecurity is presented in Table I. The course included three components: class lectures, exercises, and a semester long project assigned individually to each student.

The course included two exercises related to automated security static analysis. The first exercise focused on identifying security smells. The second exercise focused on how Git hooks can be used to automatically conduct security static analysis. Each of the exercises maps to a knowledge unit (KU) recommended by the U.S. National Center of Academic Excellence in Cyber Defense Education (CAE-CD) [7]. KUs are CS-related topics deemed essential or recommended by the U.S. National Center of Academic Excellence to develop a curriculum related to cyber defense education. Before assigning each exercise necessary theoretical concepts were covered by the instructor. We describe each exercise as follows:

Exercise#1 - Security Smells: The purpose of this exercise was to allow students to apply their knowledge related to security smells gathered in the lecture and apply it to SaltStack [5] scripts. Security smells are recurring coding patterns that are indicative of security weaknesses [4], [5]. A hard-coded password, such as `password => ''v23zj59an''` is an example of a security smell [5]. SaltStack scripts are used to implement the practice of infrastructure as code (IaC), the practice of managing system configuration automatically using dedicated programming languages and by applying recommended software engineering best practices [8]. As part of this exercise the students were asked to perform two tasks: *first*, the students were asked to manually inspect three SaltStack scripts to identify security smells. *Second*, the students were asked to build an automated program to detect the identified security smell instances. This exercise maps to the CAE-CD KU called 'Secure Programming Practices'.

Exercise#2 - Git Hooks for Automated Security Static Analysis: The purpose of this exercise was to help students learn how to integrate security using a single example of Git hook [6]. Automated security static analysis is considered as a good practice to integrate security into software development workflows. If a software repository uses Git, then using Git-based utilities, such as Git Hooks, automated security static analysis can be performed. As part of this exercise, students were asked to learn about Git hooks, and how to create a Git hook so that upon committing a file, a security static analysis tool can run and scan all files in the repository. To perform security static analysis the students used `cppcheck`, a security static analysis tool for C/C++ code. This exercise maps to the CAE-CD KU called 'Secure Programming Practices'.

III. RQ1: Student Performance in Exercises

In this section, we provide the methodology and results for **RQ1:** ***What is the performance of students in exercises related to automated security static analysis?***

A. Methodology to Answer RQ1

We answer RQ1 by using information related to the percentage of task completed obtained from the course gradebook. Once the deadline for each exercise passed the instructor inspected and graded the submission materials. The instructor also documented the percentage of task completed for each exercise.

B. Answer to RQ1

We answer RQ1 by reporting summary statistics for (i) percentage of task completion for each exercise, and (ii) grades obtained for each exercise. The summary statistics for grades and completion amount is provided in Table II. From the statistics presented in Table II, we observe students to perform overall well in both exercises related to automated security static analysis.

IV. RQ2: Perceptions of Exercises

In this section, we provide the methodology and results for **RQ2:** ***What are the perceptions of students for exercises related to automated security static analysis?***

A. Methodology to Answer RQ2

For each exercise the students were required to participate in a survey that asked two questions:

- Survey(Q1): What are the positive aspects of the exercise?
- Survey(Q2): What are the negative aspects of the exercise?

We use the answers provided by the students for Survey_{Q1} and Survey_{Q2} to answer RQ2. We apply open coding [6] to determine categories that express positive and negative aspects of the students for the two exercises.

[5]https://www.saltstack.com/

[6]https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks

TABLE II: Summary Statistics of Grades and Completion Amount For Two Exercises

Exercise Name	**Grades** (Min., Median, Max.)	**Completion** (Min., Median, Max.)
Security Smells	(45%, 65%, 100%)	(45%, 61%, 100%)
Git Hooks for Automated Security Static Analysis	(30%, 100%, 100%)	(40%, 100%, 100%)

B. Answer to RQ2

We answer RQ2 by providing positive and negative perceptions for the two exercises in the following subsections:

1) Positive Perceptions of Exercises: We identify four categories of positive perceptions. A mapping between each identified category and each exercise is presented in Table III. The number of students who have reported the category for an exercise is presented in parenthesis. For example according to Table III, skill set development was mentioned by six students for the exercise related to security smells. We describe each identified perception below:

Positive Perception#1 - Skill Set Development: For multiple exercises the students mentioned that the assigned exercises helped them to learn new tools and techniques needed in software engineering. For the security smell exercise one student stated exercises of this nature "*is highly appreciated as it helps to get a diverse skill set*". The idea of using Git hooks for secure software development came as a pleasant surprise for one student "*Very cool to learn about git hooks and realize how useful it could be for software projects. I was not aware that git provided this feature prior*".

Positive Perception#2 - Lecture Reinforcement: The conducted exercises provided students the opportunity to get a better understanding of what was being taught in the class lectures. The exercises complemented the class lectures by providing students clarity, as noticed by one student for the security smell exercise "*[it] was nice to actually use what we learned in class and reinforce the material*".

Positive Perception#3 - Practicality: Both exercises were perceived as practical by the students. One student stated "*I have never used git hooks before so I didn't know how easy they were. I will definitely be using them in the future. I also want to run cppcheck on some of my own repositories. I think this workshop was pretty easy, but very practical.*". For the security smells exercise one student also mentioned practicality and stated "*Practical knowledge of inspecting a configuration file and detecting ICPs.*".

Positive Perception#4 - Sense of Accomplishment: The exercises helped students to gain a sense of accomplishment. For the security smells exercise one student felt satisfied as the student was able to have working code that finds security smells: "*Taking the rules and turning them into working, simplistic but working, code was very satisfying. The hands on experience and exposure to short files that contained somewhat hidden ICPs was also valuable*".

2) Negative Perceptions of Exercises: We identify three categories of negative perceptions expressed by students for exercises. A complete mapping between the identified categories and the applicable exercise is provided in Table IV. In Table IV, the 'Reported Negative Perception' column states the negative perception category names and the count of students who stated the category enclosed within parenthesis. For example, for the security smell exercise, the category 'lack of background' is expressed by two students. 'None' indicates that no negative perceptions were reported by students for a certain exercise. We describe each of the categories below:

TABLE III: Positive Perceptions for the Exercises.

Exercise Topic	**Reported Positive Perception**
Security Smells	Skill Set Development (6), Lecture Reinforcement (3), Practicality (5), Sense of Accomplishment (1)
Git Hooks for Automated Security Static Analysis	Skill Set Development (8), Practicality (7)

Negative Perception#1 - Lack of Background: Despite detailed written instructions, we observe students to express a lack of background for both exercises. For example, while identifying and detecting security smells in SaltStack scripts one student found comprehension of SaltStack scripts to be difficult: "*I think SaltStack scripts are hard to look through especially if your not familiar ... I spent a lot of time trying to look up and research how to get the scripts to parse*". Another student expressed similar views and identified learning about SaltStack as a "*barrier created by unfamiliar language which made it harder to focus*". Even though the instructions on how to use the Docker image were given for each exercise, the students faced challenges: "*I didn't know that 'exiting' from the shell will destroy the running image, and when I rerun the Docker image all my work was gone*".

Negative Perception#2 - Artifact Management: All artifacts i.e., datasets and scripts for each exercise was shared using a Docker image. The Docker image was available using the instructor's DockerHub account, which included all necessary dependencies to run certain programs needed to complete each exercise. While downloading the Docker images the students provided negative feedback stating the "*Took a long time to setup docker and run the code*". Another student commented: "*seems unnecessary to download a docker image of some 800+ MB to work on a small python file*". Transfer of files back and forth between the Docker image and the development environment also created negative experience for one student: "*dev environment is in Windows ... Docker is in a virtual machine ... passing files back and forth is tedious*".

TABLE IV: Negative Perceptions for the Exercises.

Exercise Topic	**Reported Negative Perception**
Security Smells	Lack of Background (2), Artifact Management (1)
Git Hooks for Automated Security Static Analysis	None

V. Related Work

Our paper is closely related with prior publications related to cybersecurity education. Mountrouidou et al. [9] described their experience in integrating cybersecurity concepts into the general curriculum of a liberal arts degree and reported that if cybersecurity modules are flexible, then they can be incorporated into a general education curriculum. Olano et al. [10] reported their experience of introducing 'SecurityEmpire' in an undergraduate course to teach cybersecurity concepts to students. They [10] reported SecurityEmpire to help increase awareness and engagement about cybersecurity amongst students. Parrish et al. [11] synthesized educator experience related to teaching cybersecurity concepts to students, and advocated to treat cybersecurity as a meta-discipline because applications of cybersecurity concepts are applicable for a diverse set of domains, such as computer science, law, and business. Celeda et al. [12] reported their experience of using 'KYPO4INDUSTRY', a testbed to teach cybersecurity of industrial control systems. They [12] observed that while 'KYPTO4INDUSTRY' shows promise, physical processes that are related with industrial control systems make the proposed testbed generalizable.

The above-mentioned description shows the prevalence of experience reports related to a wide range of cybersecurity education concepts, such as gaming and industrial control systems. However, we observe a lack of research that discusses the experience of conducting a course related to secure software development, which we address in this paper.

VI. Future Work and Conclusion

One limitation of our findings is that it is limited to the sample size, i.e., 12 students. We plan to build upon our preliminary work to collect and analyze more student data from courses related to secure software development taught at TnTech. We also plan to add exercises related to security smells for other types of artifacts, such as Kubernetes, a container orchestration tool used in DevOps [13]. Recent work by Shamim et al. [13] shows that open source development of Kubernetes manifest can include security defects [14], which necessitates integration of education materials related to Kubernetes security into the course curriculum.

As DevOps has become mainstream amongst IT organizations, the need for integrating cybersecurity has become pivotal to ensure the software development and deployment process is secure. As part of the Secure Software Development course taught at TnTech, the first author has developed exercises to teach students how cybersecurity concepts can be integrated into DevOps. From our preliminary results we observe students to perceive the exercises positively. Furthermore, we observe majority of the students to perform well into the exercises. Our WIP paper shows promise on how automated security static analysis can be integrated into CS curriculum.

Acknowledgment

We thank the PASER group at Tennessee Technological University for their valuable feedback. This research was partially funded by the National Science Foundation (NSF) award # 2026869.

References

[1] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*, 1st ed. Addison-Wesley Professional, 2010.

[2] A. A. U. Rahman and L. Williams, "Software security in devops: Synthesizing practitioners' perceptions and practices," in *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*, 2016, pp. 70–76.

[3] C. Parnin, E. Helms, C. Atlee, H. Boughton, M. Ghattas, A. Glover, J. Holman, J. Micco, B. Murphy, T. Savor, M. Stumm, S. Whitaker, and L. Williams, "The top 10 adages in continuous deployment," *IEEE Software*, vol. 34, no. 3, pp. 86–95, 2017.

[4] A. Rahman, M. R. Rahman, C. Parnin, and L. Williams, "Security smells in ansible and chef scripts: A replication study," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 1, Jan. 2021. [Online]. Available: https://doi.org/10.1145/3408897

[5] A. Rahman, C. Parnin, and L. Williams, "The seven sins: Security smells in infrastructure as code scripts," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE '19. Piscataway, NJ, USA: IEEE Press, 2019, pp. 164–175. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00033

[6] J. Saldana, *The coding manual for qualitative researchers*, 2nd ed. Los Angeles, CA, USA: Sage, 2015.

[7] NIETP, "NIETP About CAE Program," https://www.iad.gov/nietp/CAERequirements.cfm, 2020, [Online; accessed 18-Dec-2020].

[8] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Information and Software Technology*, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584918302507

[9] X. Mountrouidou, X. Li, and Q. Burke, "Cybersecurity in liberal arts general education curriculum," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 182–187. [Online]. Available: https://doi.org/10.1145/3197091.3197110

[10] M. Olano, A. Sherman, L. Oliva, R. Cox, D. Firestone, O. Kubik, M. Patil, J. Seymour, I. Sohn, and D. Thomas, "Securityempire: Development and evaluation of a digital game to promote cybersecurity education," in *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*. San Diego, CA: USENIX Association, Aug. 2014. [Online]. Available: https://www.usenix.org/conference/3gse14/summit-program/presentation/olano

[11] A. Parrish, J. Impagliazzo, R. K. Raj, H. Santos, M. R. Asghar, A. Jøsang, T. Pereira, and E. Stavrou, "Global perspectives on cybersecurity education for 2030: A case for a meta-discipline," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE 2018 Companion. New York, NY, USA: Association for Computing Machinery, 2018, p. 36–54. [Online]. Available: https://doi.org/10.1145/3293881.3295778

[12] P. Čeleda, J. Vykopal, V. Švábenský, and K. Slavíček, "Kypo4industry: A testbed for teaching cybersecurity of industrial control systems," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1026–1032. [Online]. Available: https://doi.org/10.1145/3328778.3366908

[13] M. I. Shamim, F. A. Bhuiyan, and A. Rahman, "Xi commandments of kubernetes security: A systematization of knowledge related to kubernetes security practices," in *2020 IEEE Secure Development (SecDev)*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2020, pp. 58–64. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SecDev45635.2020.00025

[14] D. B. Bose, A. Rahman, and S. I. Shamim, "'under-reported' secuirty defects in kubernetes manifests," in *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering Workshops*, ser. ICSEW'21, 2021, to appear, pre-print: https://akondrahman.github.io/papers/k8s-encycris2021.pdf.