2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #51950

WIP: Using Gamification to Introduce DevSecOps Concepts to Software Engineering Students

Dr. Walter W Schilling Jr., Milwaukee School of Engineering

Dr. Walter Schilling is a Professor and Program Director for Cybersecurity Systems in the Diercks School of Advanced Computing at the Milwaukee School of Engineering (MSOE). He teaches courses in both the Cybersecurity Systems and Software Engineering programs, specializing in software reliability, static analysis, cybersecurity, embedded systems, software verification, and software security. Before joining the MSOE faculty, Dr. Schilling served as a graduate researcher at NASA's Glenn Research Center and worked as a software product design engineer for both Visteon Corporation and Ford Motor Company. He currently serves as an editor for the Computers in Education Journal and is a commissioner with the ABET Engineering Accreditation Commission (EAC), supporting quality assurance in engineering education. Dr. Schilling's contributions to engineering education have been recognized with several honors, including the Ohio Space Grant Consortium Doctoral Fellowship, the ASEE New Engineering Educators Distinguished Service Award, the Merl K. Miller Award from the Computers in Education Journal, and the ASEE Computers in Education Distinguished Service Award.

WIP: Using Gamification to Introduce DevSecOps Concepts to Software Engineering Students

Keywords
DevSecOps, Game-Based Learning, Cybersecurity Education, Agile Software Development, Serious Games

Abstract
DevSecOps is a paradigm that integrates security practices throughout the DevOps pipeline, fostering shared responsibility among development, operations, and security teams. While essential in modern software engineering, these concepts—along with Agile methodologies and risk management—are often abstract and difficult for undergraduate students to grasp. This paper introduces BuildOps, a gamified learning tool designed to engage students with DevSecOps principles through interactive play. Extending the Agile App Security Game, BuildOps addresses the limitations of fixed scenarios by incorporating stochastic sprint outcomes, dynamic attack events, and prioritization trade-offs between feature delivery and security debt. The activity simulates real-world development, requiring players to balance sprint planning and risk mitigation while working within resource constraints. We detail the evolution of BuildOps from prototyping to pilots in a cybersecurity summer camp, a junior-level DevSecOps course, and a general software engineering course. Preliminary results indicate that gamification significantly enhances engagement and comprehension. Future work will explore industry validation, scalability, and formal assessment strategies to quantify learning gains.

1. Introduction

DevSecOps has emerged as a critical paradigm within Software Engineering, fundamentally integrating security practices throughout the DevOps pipeline. By dissolving the traditional barriers between development, operations, and security teams, it fosters a culture of shared responsibility. As modern software organizations increasingly expect graduates to utilize these tools and practices immediately upon hiring, it is imperative that software engineering students internalize security as a continuous habit rather than a post-deployment verification step. These concepts are often challenging for undergraduate students to grasp early in their studies.

Historically, DevOps emerged to break down silos between development and operations, combining cultural philosophies with automation to increase delivery speed and reliability [1]. While DevOps focuses on the flow of delivering reliable functionality, DevSecOps explicitly elevates security to a first-class concern. It ensures that security practices are integrated throughout the pipeline, transforming security from a late-stage gate into a shared responsibility [2].

Current industry guidance emphasizes "shifting security left", automating testing and controls within CI/CD, and continuously generating compliance artifacts, allowing teams to maintain velocity without sacrificing protection [3]. This approach is now codified by major standards. The IEEE 2675-2021 DevOps standard aligns these principles, including "mission first,"

"continuous everything", and “systems thinking”, with established lifecycle process models [4]. Similarly, the NIST DevSecOps guidance, SP 1800-- 44, maps secure development practices to the Secure Software Development Framework and zero-trust patterns, underscoring the rigorous regulatory context students will face in the workforce [5].

Beyond compliance, DevSecOps plays a vital pedagogical role by supporting systems thinking and architectural depth. It pushes students beyond isolated coding tasks, requiring them to engage with microservices, cloud architecture, and Infrastructure as Code (IaC). Furthermore, DevSecOps instills a necessary understanding of ethical responsibility and societal impact. By integrating security early, students learn that reducing vulnerabilities is not merely a technical requirement but a commitment to public safety, welfare, and trust.

2. Background and Related Work

Gamification is a widely adopted pedagogical strategy within the engineering community. In software engineering, core concepts are often abstract and complex, making them difficult to grasp through traditional lectures alone; they are more readily understood through active learning paradigms.

Gamification offers several advantages in software engineering education. First, it consistently increases student engagement and motivation. When abstract concepts are presented through interactive, game-informed tasks, learners demonstrate higher participation and persistence, and courses more readily foster active learning [6]. Second, gamification can lead to improved learning outcomes. By embedding mechanics into exercises and simulations, students gain hands-on practice and show better retention of complex topics, including Agile practices, software testing, and quality assurance. Empirical reviews report positive effects on motivation, engagement, and performance across multiple knowledge areas, though they note variability in study rigor [7]. Third, many gamified designs encourage collaboration and teamwork, often through cooperative or competitive structures that mirror real-world development environments, thereby reinforcing professional skills central to engineering practice. A further benefit is individualized and self-paced learning. Game structures allow learners to progress through challenges at suitable difficulty levels, reducing anxiety and supporting mastery before advancement. It is important to note, however, that Gamification should be used as a supplemental tool rather than a complete replacement for traditional teaching techniques. [8]

While gamification offers clear benefits, several challenges have been identified. A misalignment between game mechanics and learning objectives can lead students to prioritize game mechanics, such as points and badges, over deep conceptual mastery of the material [6], [7]. Overemphasis on extrinsic rewards may reduce intrinsic motivation and increase stress, especially in competitive formats [6], [7]. Many studies rely on self-reported engagement rather than validated learning assessments, limiting evidence of effectiveness [7], [9]. Structural gamification—adding points without meaningful integration—risks shallow learning unless paired with authentic tasks and reflection [9]. Inclusivity and accessibility issues arise when

game formats favor certain learner profiles, and scalability can strain instructor resources [1], [9]. To mitigate these risks, literature recommends aligning game elements with learning goals, balancing intrinsic and extrinsic motivators, and using rigorous evaluation strategies.

In software engineering education, gamification has frequently been applied to teach Agile methodologies, particularly Scrum and Kanban. Many classroom implementations leverage physical or board-game mechanics to make abstract processes tangible.

For example, Marek and Martyniuk-Sienkiewicz [10] introduced a drawing-based game in which students use Post-it notes to represent Product Backlog Items (PBIs) and engage in collaborative sketching to simulate Scrum ceremonies. Lopez and Fernandes [11] employ a LEGO-based board game to teach Scrum practices, incorporating sticky notes for Kanban-style planning of PBIs and User Stories, thereby reinforcing iterative development and visual workflow management. Shore and Belshee [12] developed the Agile Fluency Game, a Euro-style board game designed for professional workshops. This game immerses participants in simulated team environments, requiring them to make strategic trade-offs between feature delivery and investment in Agile practices over multiple iterations. Collectively, these examples illustrate how gamification—through physical and board-game formats—can effectively convey Agile principles, foster collaboration, and provide experiential learning opportunities that mirror real-world software development dynamics.

Gamification is also equally prevalent in cybersecurity education at both the K-12 and collegiate levels. Weitl-Harms et al [13] provide a systematic mapping of 74 studies where gamification was used in the teaching of cybersecurity, highlighting formats such as board games, simulations, and Capture the Flag (CTF) competitions as common strategies. Matovu et al. [14] demonstrate the effectiveness of non-digital serious games for improving threat recognition and secure practices among undergraduates. Williams et al. [15] emphasize CTF-based learning, which boosts motivation and fosters interdisciplinary collaboration. Li [16] explore AI-powered mobile mini-games that adapt difficulty to enhance engagement and retention.

Board and tabletop games are especially popular in cybersecurity. Shostack and Associates [17] catalog over 90 unique card-based or tabletop games designed for security education. Many of these games focus on incident response (IR) workflows, emphasizing detection, containment, and recovery phases, as well as defensive controls, team coordination, and escalation playbooks. Examples include *Backdoors & Breaches*, *Roles and Responders*, and *Malware & Monsters*. Other games teach offensive techniques and model attacker behavior, incorporating concepts such as MITRE ATT&CK tactics, lateral movement, reconnaissance, and exploitation. Representative titles include *Cyber Attack Chain*, *Breach Chain*, and *Pen Test Blitz*. A third category trains players to identify threats, analyze attack surfaces, and to think like adversaries, using frameworks such as STRIDE, attack trees, and misuse cases. Games like *Elevation of Privilege*, *OWASP Cornucopia*, and the *Social Engineering Requirements Game* exemplify this approach.

For software engineering students specifically, games emphasizing Application Security (AppSec) are particularly relevant. Unlike general cyber games, these tools help developers understand secure coding practices and design trade-offs within the Secure SDLC. Examples such as *Attackers and Friends* and *Credential Shuffle* reinforce secure design principles in an interactive, team-based format, directly supporting the transition to DevSecOps.

3. Overview of the Agile App Security Game

BuildOps is founded upon the Agile App Security Game [18], a collaborative, team-based activity designed to initiate structured conversations about application security and prioritization. The game is lightweight to deploy; participants download, print, and cut the story cards, review the brief instructions, and can complete a full session in approximately 60–90 minutes. This duration allows sufficient time for a facilitator-led debrief to consolidate learning. Because multiple tables can operate in parallel, the activity scales to classroom or workshop environments with several teams (up to ~30 participants) while preserving an emphasis on discussion and reflection on secure design trade-offs.

Gameplay centers on the fictional "MoneyZoom" money-management app (Figure 1), which has gone unexpectedly viral despite having minimal built-in security. Teams act as Agile developers tasked with prioritizing and implementing security enhancements across four rounds, each representing a two-week sprint.

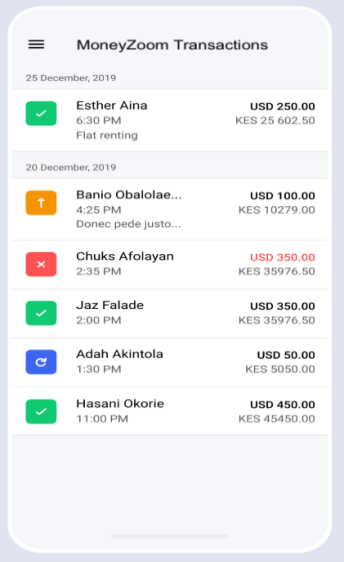

Figure 1 The money Zoom App. [18]

In every sprint, players receive a set of playing cards (Figure 2) representing potential actions. These cards span security features, testing actions (including fixes arising from tests), and research tasks that may unlock new options in subsequent iterations. Teams must choose which stories to implement within a fixed story-point limit, balancing risk reduction against effort constraints.

Figure 2 Agile App Security Game sample playing card. [18]

After each sprint, the facilitator provides feedback on attacks observed against the previous release and on whether those attacks were mitigated by the selected features. Prior decisions can also open new stories, reinforcing iterative learning. The activity models real-world Agile rhythms—planning, release, and retrospective—while highlighting secure-by-design principles, evolving threats, and the need for team coordination when security work competes for priority in the backlog [11], [12].

While the Agile App Security Game is a valuable exercise for professional workshops, it presents limitations for undergraduate education. From a pedagogical standpoint, the game relies on a static attack schedule (Figure 3). Each time the game is played, the attacks occur in the exact same order. This determinism hampers replayability and reduces opportunities for exploration and problem-solving. This fixed nature works well for a one-off workshop but diminishes the gamification value in a semester-long course, as the experience can quickly turn into rote execution rather than strategic thinking.

Developer Security Essentials: Agile App Security Game: Attacks

Attack	Mitigations	Card Round	Happens after sprint
Scanning Kiddie – gets to server by accident – Malice – pwn – deletes all data.	• Backup Server (BS), or • Server Patches Up to Date (PT)	1 1	1
DoSing Kiddie – gets to server by accident - DDOS attacks on other servers	• Server Patches Up to Date (PT)	1	1
Hacking Kiddie - gets to server by accident – downloads and publishes what he finds.	• Server Patches Up to Date (PT)	1	2
Phishing Kiddie – sends spam email, gets subscribers to enter credentials in spoof website.	• Contingency Plan (CP), or • Two Factor Authentication (2F)	2 2	2
Detective or journalist hacker – wants account information for specific users – Server hack – publish information or get individual complaint.	• Server Pen Testing, plus Encrypt and Hide Data on Server (PT, EH*)	1, PT	2
MITM kiddie – Spoof wifi access point in airport – Gains credentials – randomly hacks accounts.	• HTTPS in Protocol (HP), or • Two Factor Authentication (2F)	1 2	3
Aggrieved hacker – Gains access to server – downloads or modifies server data – publicizes.	• Server Pen Testing, and Encrypt and Hide Data on Server (PT, EH*)	1, PT	3
MITM Mafia – Spoof wifi access point in airport – use dodgy root certificates to validate all banking services - Gains credentials – steals small amount from each – and identity theft from some users.	• Forced Upgrades and SSL Pinning (FU*, SS*), or • Two Factor Authentication (2F)	1, RA 2	3
Mafia Team – Gain access to office, get passwords from server logs, small theft from tens of thousands of accounts.	• Prevent Access to Office (PA) or • Filtered Logging for Server (FL*) or • Two Factor Authentication (2F)	2 PT 2	4
Mafia Advanced Programming Team – gain access to server though zero day exploit, installed hacked version, transfer money out of accounts.	• Network Monitoring for Server (NM*)	PT	4
Mafia APT - Malware on device via email – sends back credentials found in logs to command and control server – used to clean out all compromised accounts.	• Forced Upgrades and Sanitize App Logging (FU*, SA*), else: • Two Factor Authentication (2F)	RA, RA 2	4

Figure 3 Agile App Security Game Attacks

4. Design Goals and Target Audience for BuildOps

BuildOps was designed to extend the Agile App Security Game by rendering gameplay more dynamic and decision rich. While retaining the approachable board-game structure, BuildOps intentionally incorporates variability to reflect real-world engineering uncertainty. Rather than following a deterministic sequence, the game introduces branching paths and visible consequences, allowing players to experience how their choices directly shape security posture, delivery velocity, and risk.

The tool was designed to scale across two distinct target audiences. First, it serves high school seniors attending enrichment camps at the Milwaukee School of Engineering. For these novices, the game was designed to be accessible with only a brief overview of high-level concepts (sprints, user stories, and backlogs). Second, it targets undergraduate students in a junior-level DevSecOps course [19]. These students possess a foundational understanding of the software process and are expected to navigate deeper trade-offs.

5. Gameplay Mechanics

To increase variability, we introduced three core mechanical changes to the original game design:

- *Stochastic Sprint Outcomes*
- *Balancing Features vs. Security*
- *Probabilistic Threat Modeling*

In real-world Agile development, sprint success is rarely guaranteed; unforeseen complexity often prevents teams from meeting the "Definition of Done." BuildOps simulates this uncertainty.

Unlike the original game, where cards played are automatically completed, BuildOps introduces mechanisms in which PBIs may remain incomplete or require scope adjustments. This forces teams to manage "carry-over" work and re-prioritize their backlog dynamically during the Sprint Review.

A significant limitation of the original *Agile App Security Game* was its exclusive focus on security tasks. Because players only held "security cards," the friction between delivering customer value and maintaining security was lost. BuildOps introduces "Feature PBIs" cards representing new system functionality. This forces players to make realistic trade-offs: prioritizing a revenue-generating feature often means delaying a critical security patch, thereby accumulating technical debt.

Finally, we revised the attack surface to be probabilistic rather than deterministic. While production systems face constant threats, they do not experience a successful breach in every sprint. BuildOps includes the potential for "quiet" sprints in which no attacks occur, as well as sprints in which attacks are attempted but may be thwarted by prior preventive investments. This

variability reinforces risk-based thinking, demonstrating that effective security is proactive rather than merely reactive.

As a secondary design goal, BuildOps served as an experiment in leveraging Generative AI. We utilized Microsoft Copilot to assist in the asset creation process. Copilot generated visual assets and thematic illustrations for the physical board and assisted in drafting and refining Product Backlog Items (PBIs). This integration aimed to evaluate the efficacy of AI tools in accelerating the creative and technical workflows of educational game development.

6. Game Mechanics and Implementation

The BuildOps game is designed to simulate multiple DevSecOps sprints within a condensed timeframe, typically lasting about one hour. It is most effective when played by a group of three to four participants, who collectively assume the role of a development team.

The primary activity is conducted using the game board (Figure 4), which organizes four distinct sets of cards: Customer PBIs, Security PBIs, Sprint Results, and Attacks.

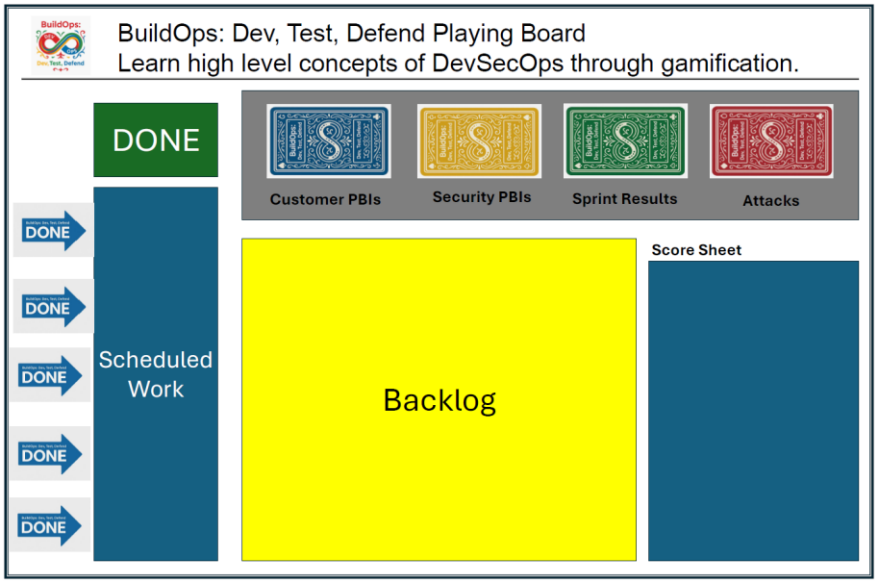

Figure 4 The game board.

Customer PBIs (Product Backlog Items), illustrated in Figure 5, represent new features requested by the customer. Each PBI is written in a user story format, though for simplicity, detailed acceptance criteria are omitted. Every PBI includes:

- Description – A narrative description of the feature in User Story form.
- Story Points – A value representing the complexity of the item. Higher numbers indicate a more difficult PBI.
- Customer Value – A numeric score representing the benefit of the feature. Higher numbers indicate greater importance or advantages to the business.

Figure 5 Sample Customer PBI Card

Security PBIs, shown in Figure 6, address technical debt and mitigate vulnerabilities that could lead to system compromise. These items generally offer lower customer value because they prioritize stability and compliance over visible functionality. Like customer PBIs, security PBIs are assigned story points. Some cards appear only in later sprints to reflect evolving priorities and prior decisions.

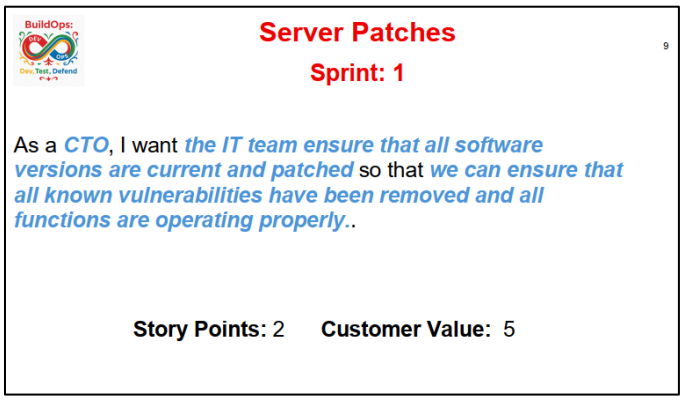

Figure 6 Sample security PBI.

Sprint result cards, an example of which is shown in Figure 7, simulate the variability inherent in agile development. Outcomes vary, just as in real agile processes:

- **Ideal case:** All planned PBIs are completed, achieving the sprint goal.
- **Typical case:** Most PBIs are finished, with maybe a few carried over due to complexity or shifting priorities.
- **Exceptional case:** The team finishes early, allowing extra work or backlog refinement.
- **Challenging case:** Significant impediments reduce output, signaling process improvements are needed.

These variations mirror real-world unpredictability and emphasize the importance of adaptive planning.

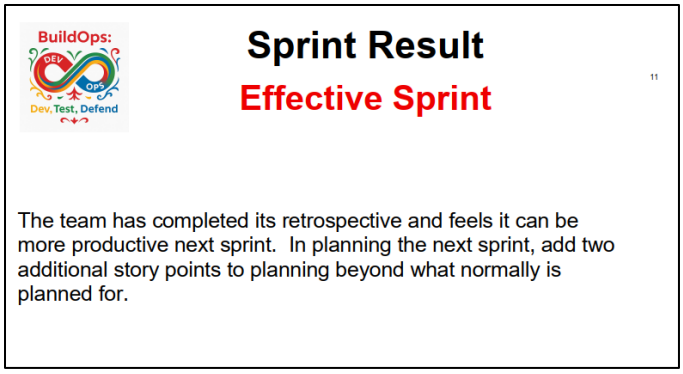

Figure 7 Sample sprint completion card.

Attack cards, such as the example shown in Figure 8, introduce stochastic events representing cyberattacks. While attacks can theoretically occur at any time, the game simplifies this dynamic by evaluating attacks only at the conclusion of each sprint. Each card specifies an attack type, its impact score, and the mitigation strategies that would have prevented it. If the team has implemented the relevant security measures, the attack is considered unsuccessful; otherwise, the corresponding penalty is applied to the team's score.

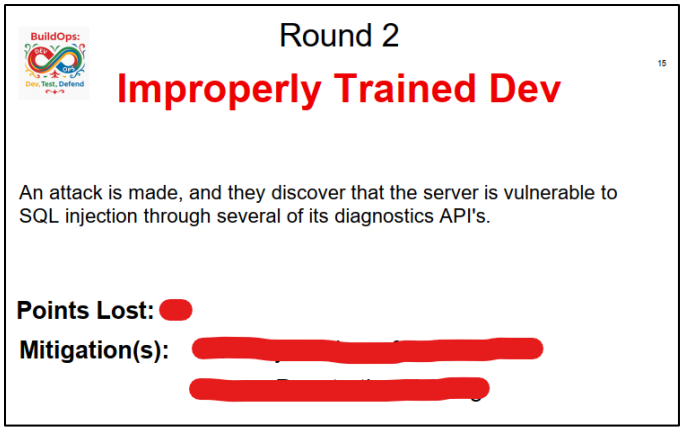

Figure 8 Sample attack card.

Gameplay begins with setting up the board and placing all Security PBI cards for the initial sprint in the backlog. Two customer PBIs are drawn from the stack, and the team collaborates to select PBIs for inclusion in the first sprint. The primary objective is to maximize customer value while managing security risk. Each sprint is constrained by an 11-story-point budget, requiring strategic prioritization. Selected PBIs are placed in the Scheduled Work area and arranged by priority.

Once planning concludes, the game advances to simulated sprint execution. Because this is a simulation, development occurs instantly. The team draws a Sprint Result Card that determines the sprint's actual output (e.g., whether all planned points were completed). Completed PBIs contribute their Customer Value to the score, while incomplete items remain in the backlog.

Finally, two attack cards are drawn. The team checks if they have previously completed the specific Security PBIs required to mitigate these attacks. If the mitigation is in place, the attack fails and is discarded. If the mitigation is missing, the team suffers the penalty points. Crucially, the unmitigated attack card is returned to the deck and reshuffled. This simulates persistent risk: a system remains vulnerable to the same attack vector until the root cause is patched.

This iterative process continues through seven sprints, simulating the complex interplay between feature development, security investment, and risk management in a modern Agile environment.

Game scoring is done throughout using a scorecard, shown in Figure 9. Essentially, the score for each sprint is the sum of the customer value added by the sprint minus the score for unmitigated attacks. A sprint score can be positive or negative depending on the game flow.

BuildOps: DEV OPS Dev, Test, Defend

Score Card

Sprint Number	Customer Values of PBI's completed	Points for Attack Cards	Score for the Sprint
1	PBI 1: _____ PBI 2: _____ PBI 3: _____ PBI 4: _____ PBI 5: _____ PBI 6: _____	Attack 1: _____ Attack 2: _____	Customer Value (Total of PBI Points)_____ - Attack Card Scores:_____ Sprint Score: _____
2	PBI 1: _____ PBI 2: _____ PBI 3: _____ PBI 4: _____ PBI 5: _____ PBI 6: _____	Attack 1: _____ Attack 2: _____	Customer Value (Total of PBI Points)_____ - Attack Card Scores:_____ Sprint Score: _____
3	PBI 1: _____ PBI 2: _____ PBI 3: _____ PBI 4: _____ PBI 5: _____ PBI 6: _____	Attack 1: _____ Attack 2: _____	Customer Value (Total of PBI Points)_____ - Attack Card Scores:_____ Sprint Score: _____
4	PBI 1: _____ PBI 2: _____ PBI 3: _____ PBI 4: _____ PBI 5: _____ PBI 6: _____	Attack 1: _____ Attack 2: _____	Customer Value (Total of PBI Points)_____ - Attack Card Scores:_____ Sprint Score: _____
5	PBI 1: _____ PBI 2: _____ PBI 3: _____ PBI 4: _____ PBI 5: _____ PBI 6: _____	Attack 1: _____ Attack 2: _____	Customer Value (Total of PBI Points)_____ - Attack Card Scores:_____ Sprint Score: _____
6	PBI 1: _____ PBI 2: _____ PBI 3: _____ PBI 4: _____ PBI 5: _____ PBI 6: _____	Attack 1: _____ Attack 2: _____	Customer Value (Total of PBI Points)_____ - Attack Card Scores:_____ Sprint Score: _____
7	PBI 1: _____ PBI 2: _____ PBI 3: _____ PBI 4: _____ PBI 5: _____ PBI 6: _____	Attack 1: _____ Attack 2: _____	Customer Value (Total of PBI Points)_____ - Attack Card Scores:_____ Sprint Score: _____
Total Score (Sum of all Sprints)			

Figure 9 Game scorecard.

7. Classroom and Program Integration

The development of BuildOps followed an iterative process driven by classroom feedback. The initial pilot utilized the *Agile App Security Game* in its original form. During this implementation, several limitations became apparent. While students were engaged in the activity, mechanics such as deterministic outcome determination and scoring did not fully align with our instructional objectives, thereby reducing the exercise's overall effectiveness.

A subsequent iteration of the classroom activity introduced modifications to address these issues. Changes included using a spinner to determine certain outcomes and adjusting the scoring system to better reflect security-related decisions. While these modifications improved the flow of the game and increased student engagement, fundamental limitations persisted. Specifically, the game struggled to convey the depth and complexity of security integration in real-world agile environments, suggesting the need for a more robust and adaptable framework.

The current version of BuildOps has been deployed in three distinct educational settings to validate its adaptability and impact.

An early version of BuildOps was implemented in a summer enrichment camp for high school seniors. The activity was conducted on the first day, prior to any formal instruction on cybersecurity topics. Despite a lack of prior domain knowledge, students responded positively. They reported that the game provided an accessible introduction to the challenges of securing systems, specifically the interplay between attacks, mitigations, and resource constraints. Importantly, students indicated that the activity helped them appreciate the trade-offs required to balance functionality, cost, and security—a foundational theme that resonated throughout the remainder of the camp.

BuildOps has also been incorporated into a junior-level college DevSecOps course as an introductory laboratory activity. This exercise occurs during the first week of the semester and serves as an “organizer" for the curriculum. The intent is to provide a high-level overview of course content and contextualize the importance of security in modern pipelines.

Student feedback indicates the activity successfully achieved these goals. While these students had prior experience with Agile sprint planning through a prerequisite software development sequence, their prior exposure lacked security and deployment considerations. Participants reported gaining a preliminary understanding of risk analysis and the implications of security decisions during sprint planning, with the game serving as a valuable bridge between their existing software engineering knowledge and the new focus on secure practices.

Beyond specialized security courses, the game was deployed in a general software engineering course at a second institution. In this context, the activity introduced computer science students to DevSecOps principles within the broader software lifecycle framework. This implementation underscores the adaptability of game-based learning for diverse curricular settings. While the depth of coverage was necessarily limited compared to the specialized course, the activity

provided an engaging entry point for discussions on security integration and lifecycle management.

Across these varied implementations, several themes emerge. First, gamification effectively engages students and fosters an appreciation for the complexity of secure software development. Second, timing matters: introducing the game early in a course primes students for deeper learning as formal instruction progresses. Finally, while game mechanics can be modified to improve usability, striking the right balance between simulation realism and playability is critical. Future work will explore iterative refinements to the design and assess learning outcomes through formal, quantitative evaluation methods.

8. Conclusions and Future Work

For its initial form, BuildOps has met its objectives in introducing students to the key concepts of DevSecOps. Student feedback has been positive about the experience, and anecdotal evidence indicates that students appreciate the need for DevSecOps processes to ensure secure software delivery.

The iterative refinement of the game has been a consistent theme across all implementations. Each deployment has revealed opportunities for improvement, leading to incremental changes in both content and mechanics. Early modifications primarily addressed superficial issues such as clarity improvements and inconsistencies within card decks. However, subsequent revisions have focused on deeper structural adjustments, including recalibrating scoring systems and enhancing card descriptions. These changes were motivated by the desire to increase the strategic depth of gameplay and ensure that decision-making reflects realistic trade-offs encountered in secure software development. While the frequency of major revisions has decreased over time, ongoing adjustments remain necessary to maintain pedagogical effectiveness and relevance.

As the fields of cybersecurity and DevSecOps continue to evolve, the game must adapt to reflect emerging practices and threats. Future iterations will incorporate new attack vectors as they become prevalent, while outdated threats will be retired to prevent obsolescence. Similarly, the domain context of the simulated system may be revisited. Although the current domain is simple for instructional purposes, it also offers opportunities to further enhance engagement among engineering students. Introducing more compelling and relatable domains—such as IoT systems, cloud-native applications, robotics, or critical infrastructure—could enhance student motivation and contextual understanding. Simulating these high-stakes environments would allow students to grapple with safety-critical constraints that differ from those in financial applications, broadening their understanding of security contexts.

Another critical area for future work involves validation through industry engagement. To date, the game has primarily been tested in academic environments with students at varying levels of expertise. However, feedback from experienced practitioners is essential to assess the realism and applicability of the scenarios. Preliminary discussions have begun with a corporation that

hosts an annual software development conference to pilot the game in a breakout session. This initiative aims to gather insights from seasoned developers regarding the authenticity of the challenges presented and the appropriateness of the scoring mechanisms. Such industry testing is imperative, as graduates of software engineering programs are preparing to enter professional environments where security considerations are integral to development workflows. Positive endorsement from industry professionals would not only validate the game's design but also enhance its credibility and appeal among students.

In addition to content and validation, future work will explore formal assessment of learning outcomes associated with gameplay. While anecdotal feedback has been encouraging, systematic evaluation using pre- and post-activity measures could provide empirical evidence of the game's effectiveness in improving understanding of DevSecOps principles, risk analysis, and secure design trade-offs. These findings would guide further refinement and support broader dissemination within engineering education.

The game is publicly available for others to use and will continue to evolve as the DevSecOps practice evolves and more student feedback is received.

9. Bibliography

[1] "What Is DevOps?," Atlassian, 2025. [Online]. Available: https://www.atlassian.com/devops. [Accessed 2 1 2026].

[2] "What is DevSecOps?," Microsoft, 2025. [Online]. Available: https://www.microsoft.com/en-us/security/business/security-101/what-is-devsecops. [Accessed 2 1 2026].

[3] DoD, DoD Enterprise DevSecOps Activities & Tools Guidebook, DoD, 2025.

[4] IEEE Standards Committee, "IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: IEEE Standard 2675-2021," IEEE, Piscataway, 2021.

[5] NIST, "NIST SP 1800-44: Secure Software Development, Security, and Operations (DevSecOps) Practices," NIST, Gaithersburg, 2025.

[6] D. López-Fernández, A. Gordillo, J. Pérez and E. Tovar, "Learning and Motivational Impact of Game-Based Learning: Comparing Face-to-Face and Online Formats on Computer Science Education," *IEEE Transactions on Education,* vol. 66, no. 4, pp. 360-368, 2024.

[7] S. Korniienko, P. Zahorodko, A. Striuk, A. Kupin and S. Semerikov, "A systematic review of gamification in software engineering education," in *AREdu 2023: 6th International Workshop on*

Augmented Reality in Education, Kryvyi Rih, Ukraine, 2023.

[8] V. Di Nardo, R. Fino, M. Fiore, G. Mignogna, M. Mongiello and G. Simeone, "Usage of Gamification Techniques in Software Engineering Education and Training: A Systematic Review," *Computers,* vol. 13, no. 8, p. 196, 2024.

[9] S. Tonhão, M. Shigenaga, J. Herculani, A. Medeiros, A. Amaral, W. Silva, T. Colanzi and I. Steinmacher, "Gamification in Software Engineering Education: a Tertiary Study," in *Proceedings of the XXXVII Brazilian Symposium on Software Engineering (SBES '23)*, 2023.

[10] K. Marek and K. Martyniuk-Sienkiewicz , "Drawing Based Game for Teaching Scrum," in *Agile Processes in Software Engineering and Extreme Programming – Workshops*, 2024.

[11] F. Castro Lopes and S. Fernandes, "The Use of Gamification for Learning SCRUM: LEGO-Based Case Study," *Trends in Higher Education,* vol. 3, no. 2, pp. 235-246, 2024.

[12] J. Shore and D. Larsen, "The Agile Fluency® Game," 2022. [Online]. Available: https://www.agilefluency.org/game.php.

[13] S. Weitl-Harms, A. Spanier, J. Hastings and M. Rokusek , "A Systematic Mapping Study on Gamification Applications for Undergraduate Cybersecurity Education," *Journal of Cybersecurity Education, Research and Practice,* vol. 2023, no. 1, 2023.

[14] R. Matovu, J. Nwokeji, T. Holmes and T. Rahman, "Teaching and Learning Cybersecurity Awareness with Gamification in Smaller Universities and Colleges," in *IEEE Frontiers in Education Conference (FIE)*, Uppsala, 2022.

[15] L. Williams, E. Anthi, Y. Cherdantseva and A. Jarved, "Leveraging Gamification and Game-based Learning in Cybersecurity Education," *Journal of the Colloquium for Information Systems Security Education,* vol. 11, no. 1, 2024.

[16] B. Li, "Security Education in Higher Education through AI-Powered Gamification," *Journal of Cybersecurity, Digital Forensics and Jurisprudence,* vol. 1, 2025.

[17] A. Shostack, "Tabletop Security Games + Cards," Shostack and Associates, 2026. [Online]. Available: https://shostack.org/games.html. [Accessed 2 1 2026].

[18] C. Weir, "Secure Development Resources," Secure Development, [Online]. Available: https://www.securedevelopment.org/resources/. [Accessed 2 1 2026].

[19] W. Schilling, "WIP: Integrating Modern Development Practices into a Software Engineering Curriculum," in *2022 ASEE Annual Conference & Exposition*, Minneapolis, 2022.