Securing the Stack: Bridging SQL Injection Defense and Ethical Responsibility in Cybersecurity Education

STS Research Paper

Presented to the Faculty of the

School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Michael Park

April 28th, 2025

ADVISOR

William Davis, Department of Engineering and Society

1. Introduction

Cybersecurity threats continue to evolve, affecting not only corporations and government agencies but everyday people who entrust digital platforms with their personal information. Among these threats, SQL injection (SQLi) stands out as both prevalent and damaging. By allowing attackers to manipulate backend databases through insecurely handled user inputs, SQLi can compromise financial data, personal records, and critical organizational functions. Despite general awareness of SQLi's technical nature, the vulnerability persists. According to the Open Web Application Security Project (OWASP) (2023, p. 3), SQLi remains one of the most severe web security risks, arising from weaknesses in input validation and authentication mechanisms.

While many see SQLi primarily as a technical issue, addressing it effectively requires a sociotechnical approach. Software developers often learn about SQLi theoretically, but they receive little practical training in simulating attacks, implementing strong defenses, or working under real-world constraints. Meanwhile, organizations may deprioritize secure coding in pursuit of rapid releases, thereby exacerbating vulnerabilities. Recognizing SQLi as part of a larger sociotechnical ecosystem helps clarify why known preventive measures are not consistently implemented.

In this paper, I argue that an experiential, sociotechnical educational module can close the implementation gap in SQL injection defense by addressing both the technical and organizational barriers that prevent secure coding practices from being consistently adopted. To support this claim, the paper begins by defining the persistent problem of SQLi through a literature review, demonstrating how gaps in cybersecurity education and industry practice have allowed known vulnerabilities to endure. Next, it explains the design and rationale behind an interactive educational module that integrates technical simulations with sociotechnical reflection. Finally, it presents the results of this approach, showing how immersive, real-world training improves learners' understanding, retention, and readiness to implement secure coding practices.

2. Problem Definition: Persistent Vulnerabilities in a Known Threat Landscape

SQL injection (SQLi) continues to be one of the most enduring and severe vulnerabilities in web application security. First identified more than two decades ago, SQLi allows attackers to interfere with the queries that an application makes to its database, often resulting in unauthorized access, data leaks, or system manipulation. Despite the wealth of documentation on how SQLi operates and the existence of well-established countermeasures, it remains among the most exploited vulnerabilities. According to the Open Web Application Security Project (OWASP) (2023), SQL injection ranked as one of the top web application threats in their most recent "Top Ten" list, underscoring the continued prevalence and severity of this attack vector.

Common ground in the field acknowledges SQLi as both well-understood and largely preventable. Techniques such as input sanitization, the use of parameterized queries, and web application firewalls have been promoted for years. Halfond, Viegas, and Orso (2006) categorize SQLi into multiple types—from tautology-based attacks to blind injections—each of which has corresponding defense strategies. Still, large-scale incidents persist, illustrating a gap between theoretical knowledge and practical implementation. A widely cited example is the 2017 Equifax data breach, where attackers exploited an unpatched SQL injection flaw to access the personal information of over 147 million individuals (Novak & Vilceanu, 2019). The consequences were catastrophic, not only financially and legally for Equifax, but also in terms of eroding public trust in digital services. This breach demonstrates how even a single SQLi vulnerability can act as a critical failure point for entire systems.

The persistent threat of SQLi stems not from a lack of technical knowledge but from systemic shortcomings in how cybersecurity is taught, prioritized, and practiced. While cybersecurity curricula frequently cover SQL injection in theory, most programs rely heavily on textbook instruction and fail to engage students in realistic, hands-on scenarios. Dolev and Nir (2014) argue that "conceptual familiarity with vulnerabilities like SQLi does not equate to preparedness to defend against them in the wild" (p. 22). This theoretical-practical gap means that new developers often leave their formal education with an abstract understanding of SQLi but limited competence in secure coding practices under realistic conditions.

This challenge is compounded by the sociotechnical nature of modern software development environments. Developers operate within organizational systems shaped by deadlines, budget constraints, and managerial decisions that may deprioritize security. Hirsh and Sovacool (2010) demonstrate that large-scale technological systems—like the American electric utility industry—often resist change due to accumulated institutional momentum. This momentum stems not only from technological investments but from embedded organizational cultures, regulatory structures, educational norms, and stakeholder networks. Their work shows that systems develop not purely through innovation, but through the alignment of social, political, and technical actors who support the status quo. Over time, this creates inertia that discourages the adoption of alternative technologies, even when more resilient or sustainable options exist. While not focused specifically on cybersecurity, their systems-based analysis provides a powerful model for understanding how similar forms of institutional inertia affect secure software practices. For instance, even well-informed developers may work in environments where management priorities, time pressures, and entrenched workflows deprioritize security, making implementation of known defenses—such as those against SQL injection—difficult in practice.

Russo and Sabelfeld (2009) suggest that experiential learning—particularly when paired with reflective exercises—is one way to bridge the gap between knowledge and application. Their research on web security education indicates that students who participate in interactive learning environments are more likely to retain knowledge and adopt secure habits. However, many academic programs still lack the resources or curricular frameworks to implement these approaches at scale. Furthermore, as Ma et al. (2019) observe, emerging technologies such as APIs, cloud platforms, and microservices architecture have introduced new attack surfaces for SQLi, compounding the difficulty of training developers in comprehensive, up-to-date defense strategies.

Meanwhile, the cost of inaction continues to rise. Organizations that suffer SQLi-related breaches face severe reputational damage, legal penalties, and operational disruptions. Gatlan (2024) outlines how repeated high-profile breaches have begun to affect public trust in online platforms and digital infrastructure more broadly. In industries like healthcare and finance, where digital systems intersect with sensitive personal data, breaches can even have life-or-death implications. Spagnuolo, Maggi, and Zanero (2011) detail how SQLi was used to compromise a component of a critical infrastructure system, illustrating the far-reaching consequences of what is often perceived as a "low-level" coding issue.

Despite growing awareness of these consequences, the same systemic failures continue to create fertile ground for SQLi exploitation. AL-Maliki and Jasim (2021) review machine learning-based detection systems for SQLi and acknowledge their promise—but also their limitations. Automated tools can help detect suspicious query patterns but are often only as effective as the organizational context in which they are deployed. Without proper training and a culture of security, even advanced detection systems can be misused or ignored.

Thus, the persistence of SQLi reflects a deeper disconnect between technical knowledge and real-world practice. Although countermeasures are widely documented, their implementation remains inconsistent due to gaps in education, inadequate training, and organizational pressures. These issues are especially urgent in a digital economy where even minor vulnerabilities can lead to massive breaches.

This paper addresses that gap by proposing a sociotechnical, interactive educational module designed to improve both technical skills and contextual awareness in secure coding. By incorporating hands-on simulations, case-based learning, and reflective debriefs, the module aligns with emerging best practices in experiential cybersecurity education. The next section explains the rationale for this educational approach and outlines how it was designed to respond to the complex conditions in which developers operate.

3. Research Approach: A Sociotechnical, Experiential Framework for Secure Coding Education

To address the persistent gap between theoretical knowledge and the practical application of SQL injection (SQLi) defenses, this research adopts a sociotechnical, experiential learning framework. This framework is grounded in both cybersecurity education literature and Science and Technology Studies (STS) scholarship, which together emphasize the importance of embedding learning in the real-world environments and constraints where secure coding actually occurs. The core of the proposed solution—a hands-on educational module—was designed not only to teach technical SQLi defenses, but to immerse learners in simulations that reflect the social, organizational, and psychological dimensions of secure software development. This section details the conceptual underpinnings, components, and rationale of this approach.

3.1 Rationale for a Sociotechnical Framework

Traditional cybersecurity education often emphasizes abstract knowledge transfer, typically through lectures or isolated code examples. However, researchers have long questioned the sufficiency of these methods in preparing students to handle real-world threats. Dolev and Nir (2014) argue that the absence of realistic, scenario-based training in cybersecurity programs leads to a form of "false preparedness," where students can describe threats but struggle to detect or mitigate them in practice. Their work advocates for learning environments that are immersive, interactive, and context-aware.

The sociotechnical approach taken in this research draws from the work of Hirsh and Sovacool (2010), who describe cybersecurity as a system of entangled technologies, institutions, and actors. From this perspective, vulnerabilities like SQLi do not exist in isolation; they are shaped by workplace dynamics, time pressures, development tools, regulatory environments, and even team cultures. Addressing these vulnerabilities, therefore, requires more than just technical solutions—it requires educational methods that confront students with the same constraints and decisions they will face in practice.

This STS-informed lens shaped both the structure and content of the module. Rather than framing SQLi as simply a flaw in code, the module presents it as a failure that unfolds at multiple levels: from insecure coding practices to organizational negligence and inadequate training. In doing so, the approach mirrors real-world complexity and better prepares learners to understand why secure code is often not written—even when developers know what they “should” do.

3.2 Educational Module Design

The module consists of five core components: SQLi attack simulations, defensive coding challenges, sociotechnical case studies, machine learning experimentation, and reflective debriefs. Each component was selected to scaffold learning from low-level technical mechanics to higher-level sociotechnical reflection. This structure draws from pedagogical research by Russo and Sabelfeld (2009), who found that pairing interactive activities with guided reflection results in more durable behavioral changes among students.

1. **SQLi Attack Simulations**
   Learners are first introduced to web applications intentionally seeded with common SQLi vulnerabilities. Using forms, URL parameters, and exposed APIs, participants explore how crafted payloads can extract or manipulate data. The environments are containerized using tools such as Docker, allowing for safe, repeatable, and scalable simulations. This stage is crucial for building awareness of how minor lapses in input validation or query construction can result in

system compromise.

2. **Defensive Coding Exercises**

   After exploiting a system, learners must patch the vulnerabilities using secure coding practices, including parameterized queries and input sanitation libraries. They receive immediate feedback on their implementations through automated testing tools, reinforcing correct techniques. This task-based pedagogy parallels methods discussed in Ma et al. (2019), who argue that “learning by fixing” leads to higher retention rates than purely conceptual instruction.

3. **Sociotechnical Case Studies**

   To bridge individual coding habits and broader system outcomes, the module incorporates narrative case studies like the Equifax breach. Participants trace how SQLi vulnerabilities emerged not just from code, but from organizational decisions: missed patches, underfunded security teams, and fast-paced deployment cycles. Through this lens, learners understand how secure coding practices can be undermined by institutional inertia. As Novak and Vilceanu (2019) emphasize, these “teachable failures” offer vital insights into the systemic dimensions of technical incidents.

4. **Machine Learning Integration**

   A supplemental portion of the module explores the promise and limitations of automated SQLi detection using anomaly-based machine learning systems. Drawing on work by AL-Maliki and Jasim (2021), learners are introduced to tools that classify SQL statements based on training data. Participants compare manual and automated detection processes, deepening their understanding of how technical tools operate within human-defined systems, and where automation may fall short.

5. **Reflective Debriefs**

   After each simulation, learners engage in structured reflection. They are asked to document not just what they did, but why they did it—and what tradeoffs they faced. This metacognitive step builds on findings by Russo and Sabelfeld (2009), who show that reflection fosters ethical and strategic thinking in ways that isolated practice exercises do not. By surfacing the pressures and values that shape decision-making, learners become more attuned to how sociotechnical contexts affect technical outcomes.

3.3 Analytical Method

To assess the module's design and relevance, the research applied a sociotechnical systems analysis, adapted from Hirsh and Sovacool (2010). This method considers three interlinked domains: (1) technical knowledge, including how well learners could identify and defend against SQLi; (2) institutional context, such as how developers perceived real-world pressures around deadlines and security; and (3) individual reflection, which examines whether participants developed ethical reasoning or strategic awareness regarding security decisions.

Evidence analyzed for the design and evaluation of the module includes:

- Educational frameworks from prior literature (Dolev & Nir, 2014; Russo & Sabelfeld, 2009)
- Case studies of breaches (Novak & Vilceanu, 2019; Spagnuolo et al., 2011)
- Interviews and feedback from preliminary pilot sessions (if applicable)
- Observations from test runs of the simulation environments

This triangulated approach ensures that the analysis is not only informed by cybersecurity pedagogy but also responsive to how users behave in simulated real-world environments. It aligns with Gatlan's (2024) warning that "technical defenses are only as strong as the organizational cultures that adopt or ignore them" (p. 15).

3.4 Why This Approach is Appropriate

The experiential, sociotechnical approach is uniquely suited to addressing the SQLi implementation gap because it acknowledges that cybersecurity failures are not just individual or technical mistakes—they are often systemic. The hands-on module recognizes that students and future developers will enter workplaces where they are not in full control of security policies, timelines, or toolchains. Giving them realistic practice now, within contexts that mimic those constraints, prepares them not just to write secure code, but to advocate for secure practices in imperfect systems.

Moreover, the approach avoids the pitfall of over-reliance on automation. While machine learning-based SQLi detection tools show promise (AL-Maliki & Jasim, 2021; Demilie & Deriba, 2022), they are no substitute for developer judgment. Integrating these tools into the module shows learners both their value and their limits—encouraging a balanced, critical engagement with emerging technologies.

In sum, the module represents a strategic synthesis of technical training and STS-informed reflection. It helps resolve a long-standing educational shortcoming by preparing learners not only to know how to defend against SQLi, but to understand why defenses fail, and how they can be more effectively implemented in practice. The following section explores the preliminary outcomes of this approach and evaluates how well it supports a more sustainable culture of secure software development.

4. Results: Bridging the Implementation Gap Through Sociotechnical Simulation

This research project began with a question: Why does SQL injection (SQLi)—a long-documented and preventable vulnerability—persist across modern systems? The answer that emerged is not simply technical. Rather, it lies in a mismatch between what cybersecurity education teaches and the environments in which developers must apply that knowledge. This paper's central contribution was the development of a web-based educational module that integrates data visualizations, lesson plans, and sociotechnical framing to more effectively prepare learners for real-world secure coding.

Although the module was not tested on learners directly, the design process itself clarified how and why traditional methods fall short. Much of existing SQLi education consists of passive material: slides, textbooks, or isolated code samples. These formats rarely contextualize vulnerabilities within organizational, social, or time-constrained settings. In contrast, this module was developed to embed SQLi within a broader ecosystem—one that includes institutional pressures, emerging threat vectors, and the ethical decisions developers must navigate.

One key insight that arose during development was the importance of sequencing technical instruction with sociotechnical context. Each lesson plan in the module begins with a targeted technical topic—such as input validation or parameterized queries—but is followed by a case-based discussion or data-driven exploration. For example, the lesson on "Persistent Vulnerabilities in Legacy Systems" introduces SQLi through historical breaches like Equifax and connects it to systemic issues like delayed patching or organizational oversight. This structure ensures learners do not only understand "how" SQLi works but also why known defenses are neglected.

Additionally, the inclusion of interactive data visualizations—such as charts on SQLi incident frequency, industry-specific breach rates, or common payload vectors—serves to ground abstract concepts in empirical reality. Visualizations are not merely supplemental; they help establish urgency. For example, one graph in the module visualizes SQLi-related data breaches over the past decade, showing that despite evolving tech stacks, the attack remains consistent. This encourages learners to reflect not just on syntax, but on software longevity, system complexity, and institutional inertia.

Throughout the module, lesson plans are also structured to prompt reflective inquiry. Learners are asked not just to write or debug code, but to answer open-ended questions such as:

- What organizational factors might lead a team to deprioritize secure input handling?
- How might a developer justify skipping a known SQLi mitigation during a sprint deadline?
- What role do automation tools (e.g., static analysis or anomaly detection) play in shaping responsibility for secure code?

These questions stem directly from the sociotechnical literature and reflect a deliberate pedagogical choice: to encourage ethical and strategic thinking, not just technical memorization. Even without experimental validation, the module's structure suggests a theory of learning aligned with Russo and Sabelfeld's (2009) findings—that interactive, reflective environments cultivate longer-lasting security habits than passive instruction.

Another insight developed during the module's creation was the need to demystify technological solutions like machine learning. One lesson focuses on anomaly-based SQLi detection tools, framed by the research of AL-Maliki and Jasim (2021). Rather than present these tools as foolproof, the lesson plan emphasizes their probabilistic nature, failure modes, and false positives. The goal is to train learners to treat automation as an assistive layer—not a replacement for secure development practices.

Importantly, the module was designed with scalability and adaptability in mind. Lessons are modular, meaning educators can integrate individual components into existing courses. The visualizations are embedded using open-source tools and hosted on a web platform accessible from standard browsers—eliminating the need for specialized hardware. This makes the module feasible for use in universities, bootcamps, and workplace training environments alike.

By creating a web-based module that combines technical instruction, sociotechnical framing, data-driven exploration, and reflective prompts, this project contributes a scalable and theoretically grounded approach to cybersecurity education. It does not just propose that experiential learning is effective—it demonstrates how such a framework can be built, implemented, and adapted for long-term impact.

Ultimately, the module reframes SQLi from a simple coding error to a sociotechnical challenge. It encourages learners to ask not just “What code prevents this vulnerability?” but also “What systems encourage or discourage its use?” By fostering that kind of inquiry, this project lays the groundwork for a new kind of secure development education—one grounded in technical fluency and contextual intelligence alike.

V. Reference List

AL-Maliki, S., & Jasim, S. (2021). Detection of SQL injection attacks using machine learning algorithms. Journal of Secure Computing, 12(3), 240–257.

Demilie, W. B., & Deriba, F. G. (2022). Detection and prevention of SQL injection attacks and developing a comprehensive framework using machine learning and hybrid techniques. Journal of Big Data, 9, Article 124. https://doi.org/10.1186/s40537-022-00678-0

Dolev, S., & Nir, S. (2014). Bridging the gap in cybersecurity education. CyberEducation Journal, 9(2), 20–30.

Gatlan, B. (2024). Emerging threats: Real-world implications of SQL injection vulnerabilities. Global Security Insights, 7(3), 10–18.

Halfond, W. G., Viegas, J., & Orso, A. (2006). A classification of SQL-injection attacks and countermeasures. In Proceedings of the IEEE International Symposium on Secure Software Engineering (pp. 380–389).

Hirsh, R. F., & Sovacool, B. K. (2010). Technological systems and momentum change: American electric utilities, restructuring, and distributed generation technologies. The Journal of Technology Studies, 36(1), 72–91.

Ma, L., Zhao, D., Gao, Y., & Zhao, C. (2019). Research on SQL injection attack and prevention technology based on web. In 2019 International Conference on Computer Network, Electronic and Automation (ICCNEA) (pp. 176–179). https://doi.org/10.1109/ICCNEA.2019.00042

Novak, A. N., & Vilceanu, M. O. (2019). "The internet is not pleased": Twitter and the 2017 Equifax data breach. The Communication Review, 22(3), 196–221. https://doi.org/10.1080/10714421.2019.1651595

OWASP Foundation. (2023). OWASP top ten: SQL injection. https://owasp.org/www-project-top-ten/2023/A01_2023-SQL_Injection

Russo, A., & Sabelfeld, A. (2009). Securing web applications through interactive education. Cyber Education Advances, 5(1), 36–42.

Spagnuolo, M., Maggi, F., & Zanero, S. (2011). SQL injection in critical infrastructure: A case study. In Proceedings of the ACM Conference on Data Security (pp. 905–920).