Editors: Eric Bodden, eric.bodden@uni-paderborn.de | Fabio Massacci, fabio.massacci@ieee.org | Antonino Sabetta, antonino.sabetta@sap.com

Securing Code

A View on the Cultural Aspects of Resilience

Tamara Lopez | The Open University

Developers navigate diverse security cultures and face pressures from production demands, leading to circumventions of resilient software development practices that are not always secure. Effective security depends on open communication, valuing developer input, and aligning organizational priorities with security goals.

"I believe that as software gets more complex, you're going to see an equilibrium which is determined more or less by sociocultural and economic factors rather than by the kind of tools that people have for vulnerability finding…"

—*Ross Anderson, Silver Bullet Talk*[1]

As foreseen by Anderson almost two decades ago, securing software using a set of tools is no longer possible. In today's encompassing digital operational space, performance is brittle. Vulnerabilities in sociotechnical software systems flow within a more or less constant stream—affecting people as they take decisions about what to buy, where to eat, how to move from here to there, and how to perform their work. Anderson realized that we can no longer fix the problem of vulnerabilities in code, but we must identify processes for managing them. The first step in this direction is for companies to create a working climate that supports resilient practice in software development.

Resilience in software engineering typically refers to the ability of technical systems to remain operational in the event of component failure. A sociotechnical view on resilience holds that systems are resilient when they are able to perform in the face of disruptions and changes that have social and technical aspects. Within software development contexts, individuals and teams contribute to resilient performance by adapting to meet situational demands and by compensating to address imperfect circumstances.[2] These adjustments in behavior are constrained by capacity, policies, and the cultural norms within organizations and professions.

Security Engagement

When we think about the need for security in code, we often think of engineering tasks in which developers enact security policies for their companies and clients through design and architectural decisions that have a technical focus. In these cases, security is seen as a primary concern within the development task, and developers are perceived to play a dominant role in setting project directions that will produce secure code.

However, it is much more common for developers to encounter security through constraints on their activity. The security policies of companies are often enforced through security mechanisms in tools. In these cases, developers are users and security compliance is mandatory, achieved through technical control.[3] Alternatively, the mechanisms are warnings in security tools indicating the presence of vulnerabilities. In both cases, behaving securely is often tangential to the task being performed, but the perceived need to comply is high.

Developers must also comply with security policies when they integrate their work within technical infrastructures. As with tool use, in integration tasks, access is the primary security mechanism employed, and there is an impetus to navigate, address, or ignore the security

Digital Object Identifier 10.1109/MSEC.2024.3500113

 Copublished by the IEEE Computer and Reliability Societies

constraints as quickly as possible so that teams can continue to make progress. This raises a tension between the drive to be productive and the priority to behave securely.[4]

Contrary to findings that developers consider security to be "burdensome," [5] our research indicates that developers perceive a "duty of care" to manage users' data as they want their own to be managed.[6] This sense of duty is coupled with loyalty toward hiring companies and suggests a willingness and commitment to behave securely in development tasks, qualities that are often associated with security culture.

Security Culture(s)

Security culture can be identified through levels of engagement, specifically how employees comply with security policies and meet acceptable standards for behavior. In addition to the qualities of loyalty and commitment mentioned before, acceptable standards include values like responsibility, trust, communication, and cooperation.[7] In software engineering contexts, there is an interest in improving culture within engineering departments and teams.[5] The broad expectation is that by strengthening the engagement of the people who make software, behavior will improve, commitment will deepen, and software will be more secure.

Though the culture within teams undoubtedly influences security outcomes in code, it is also helpful to examine perceptions toward security that have built up over time in the broader sofware development environment. Developers take part in multiple security cultures. They perform tasks in an atmosphere of mechanized constraints and social attitudes toward security held within their teams but also within their departments, employing companies, and those of customers and clients. Very often, the position toward security that developers encounter within these cultures has an origin that predates an individual task, project, or period of employment and that is deeply embedded within the technical infrastructure: in libraries, tools including database management software, and hardware.

Brittle Security

Recent incidents suggest that large-scale software intensive systems are increasingly brittle, and demonstrate that technical mechanisms alone do not keep them resilient.[8] The problem of security in software provides another example: The number of breaches in software remains high, and it has proven difficult to identify a set of development practices that uniformly and consistently produce secure systems. As in studies that examine safety-critical systems,[9] the ways in which environments discourage adaptive, proactive work practices also gives insight into what *is not* working within secure software development.

Security outcomes are often constrained by demands for production set within product development—the features or fixes that clients and product teams request and prioritize. Engineers tailor their individual behavior to respond to situational needs. When security is a secondary concern[4] to product teams and clients, developers do not perceive an ability to influence security outcomes. By contrast, when product teams and clients engage, developers are able to recommend and direct security activity.

Forms of employment also affect security in software. Consultants are expected to deliver agreed-to outputs but are treated within hiring organizations as outsiders, subject to additional access constraints and left out of conversations and decision-making.[10] Similarly, freelance developers write code that includes security implications, but the responsibility for security tasks is ambiguous or assumed, and security knowledge and skill development are left to the developers' own time and resource.[11]

Finally, production demands and the complexity of technical environments can lead to informal agreements made among developers to open ports, bypass automated builds, or give unauthorized access to devices. Actions taken in the shadows between machines keep work moving, but are not secure. The impact goes beyond outputs in the code. Our findings suggest that informal practices that have negative security implications raise concern among developers that they are not meeting professional engineering standards.[10]

To restate, through a recent developer survey suggests that developers do not know how to ask for security training or where to find it,[12] a richer understanding of the state of security practice in environments requires additional detail: Whether, for example, security mechanisms and policies were perceived by participants to be vaguely defined within companies,[13] the situations in which developers were encouraged to take action and listened to,[14] and if they were able to push back within their teams or with clients when they felt more security was needed.[10,11]

Recommendations

Sociotechnical software systems are kept resilient through the work performed by software developers, and security is fostered in work climates that encourage adaptive and proactive performance. To support this, industry can improve the work climate surrounding secure software development practices in three ways.

First, industry must acknowledge the power product development has in

setting priorities for production and the effect that this has on security outputs in code. Just as the priorities held by product stakeholders subsume the priorities of developers and teams, the demand for task completion influences individual developer behavior. Production priorities held by product stakeholders have a direct impact on engineering choices and can inhibit engineers from speaking out or take action to improve security in a product.

Second, companies must facilitate open communication between developers, product owners, and external stakeholders. The effort to integrate security and engineering teams is a positive step in this direction. Companies should provide freelance and consultant software developers with opportunities to develop security knowledge. Above all, it is necessary for companies to recognise that developers' expertise and knowledge are vital to securing products.

Third, organizations must take responsibility for fostering a working environment that supports good engineering practice. Establishing a "no blame" culture is not enough. In an analysis of survey data taken by software security groups within 211 companies over 12 years, Weir et al. identified that security activity is driven by the priorities of companies: toward compliance, knowledge distribution, or by placing the effort for security on engineers.[15] For security priorities to succeed, companies must also take responsibility for ensuring that the atmosphere supports resilient performance among development teams.

Writing code that is secure and maintaining security within environments depends on developers of all kinds being nurtured and included: permitted to disagree, direct, and advocate for security activity alongside production requirements. ■

References

1. G. McGraw and R. Anderson, "Silver bullet talks with Ross Anderson," *IEEE Security Privacy*, vol. 5, no. 4, pp. 10–13, Jul./Aug. 2007, doi: 10.1109/MSP.2007.94.
2. T. Lopez et al., "Accounting for socio-technical resilience in software engineering," In *Proc. IEEE/ACM 16th Int. Conf. Cooperative Hum. Aspects Softw. Eng. (CHASE)*, Piscataway, NJ USA: IEEE, 2023, pp. 31–36, doi: 10.1109/CHASE58964.2023.00012.
3. A. Beautement, I. Becker, S. Parkin, K. Krol, and A. Sasse, "Productive security: A scalable methodology for analysing employee security behaviours," in *Proc. 12th Symp. Usable Privacy Security (SOUPS)*, Denver, CO, USA: USENIX Association, 2016, pp. 253–270.
4. S. L. Pfleeger, M. A. Sasse, and A. Furnham, "From weakest link to security hero: Transforming staff security behavior," *J. Homeland Secur. Emergency Manage.*, vol. 11, no. 4, pp. 489–510, 2014, doi: 10.1515/jhsem-2014-0035.
5. H. Assal and S. Chiasson, "Security in the software development lifecycle," in *Proc. 14th Symp. Usable Privacy Security (SOUPS)*, 2018, pp. 281–296.
6. C. Posey, T. L. Roberts, P. B. Lowry, and R. T. Hightower, "Bridging the divide: A qualitative comparison of information security thought patterns between information security professionals and ordinary organizational insiders," *Inf. Manage.*, vol. 51, no. 5, pp. 551–567, 2014, doi: 10.1016/j.im.2014.03.009.
7. M. Beyer et al., "Awareness is only the first step: *A framework for progressive engagement of staff in cyber security*," Hewlett Packard, Business White Paper, 2015. [Online]. Available: https://www.slideshare.net/slideshow/awareness-is-only-the-first-step/55894537
8. B. Raghaven and B. Schneier. "The CrowdStrike outage and market-driven brittleness." Lawfare. Accessed: Sep. 15, 2024. [Online]. Available: https://www.lawfaremedia.org/article/the-crowdstrike-outage-and-market-driven-brittleness
9. E. Hollnagel, *Safety-II in Practice: Developing the Resilience Potentials*. New York, NY, USA: Taylor & Francis, 2017.
10. T. Lopez, H. Sharp, A. Bandara, T. Tun, M. Levine, and B. Nuseibeh, "Security responses in software development," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 3, pp. 1–29, 2023, doi: 10.1145/3563211.
11. I. Rauf, M. Petre, T. Tun, T. Lopez, and B. Nuseibeh, "Security thinking in online freelance software development," In *Proc. IEEE/ACM 45th Int. Conf. Softw. Eng. Softw. Eng. Soc. (ICSE-SEIS)*, Piscataway, NJ USA: IEEE, 2023, pp. 13–24, doi: 10.1109/ICSE-SEIS58686.2023.00008.
12. "The Linux Foundation and OpenSSF release report on the state of education in secure software development." The Linux Foundation. Accessed: Sep. 14, 2024. [Online]. Available: https://www.linuxfoundation.org/press/linux-foundation-and-openssf-release-report-on-the-state-of-education-in-secure-software-development
13. I. Hadar et al., "Privacy by designers: Software developers' privacy mindset," *Empirical Softw. Eng.*, vol. 23, no. 1, pp. 259–289, 2018, doi: 10.1007/s10664-017-9517-1.
14. D. Ashenden and D. Lawrence, "Security dialogues: Building better relationships between security and business," *IEEE Security Privacy*, vol. 14, no. 3, pp. 82–87, May./Jun. 2016, doi: 10.1109/MSP.2016.57.
15. C. Weir, S. Migues, and L. Williams, "Exploring the shift in security responsibility," *IEEE Security Privacy*, vol. 20, no. 6, pp. 8–17, Nov./Dec. 2022, doi: 10.1109/MSEC.2022.3150238.

Tamara Lopez is a lecturer of computing and member of the Software Engineering and Design research

group at The Open University, MK7 6AA Milton Keynes, U.K. Her research interests include professional development in software engineering, sociotechnical systems engineering, and security and privacy. Lopez received a Ph.D. in computing from the Open University. Contact her at tamara.lopez@open.ac.uk.

Actions taken in the shadows between machines keep work moving, but are not secure.

Companies should provide freelance and consultant software developers with opportunities to develop security knowledge.

When security is a secondary concern to product teams and clients, developers do not perceive an ability to influence security outcomes.

For security priorities to succeed, companies must also take responsibility for ensuring that the atmosphere supports resilient performance among development teams.