Investigate How Developers and Managers View Security Design in Software

Asif Imran[a]

Department of Computer Science, California State University San Marcos,
333 S Twin Oaks Valley Rd, San Marcos, CA 92096, U.S.A.

Abstract: Software security requirements have been traditionally considered as a non-functional attribute of the software. However, as more software started to provide services online, existing mechanisms of using firewalls and other hardware to secure software have lost their applicability. At the same time, under the current world circumstances, the increase of cyber-attacks on software is ever increasing. As a result, it is important to consider the security requirements of software during its design. To design security in the software, it is important to obtain the views of the developers and managers of the software. Also, it is important to evaluate if their viewpoints match or differ regarding the security. Conducting this communication through a specific model will enable the developers and managers to eliminate any doubts on security design and adopt an effective strategy to build security into the software. In this paper, we analyzed the viewpoints of developers and managers regarding their views on security design. We interviewed a team of 7 developers and 2 managers, who worked in two teams to build a real-life software product that was recently compromised by a cyber-attack. We obtained their views on the reasons for the successful attack by the malware and took their recommendations on the important aspects to consider regarding security. Based on their feedback, we coded their open-ended responses into 4 codes, which we recommended using for other real-life software as well.

1 INTRODUCTION

The recent trend in world events has made cyber threats an issue of ever-increasing importance. Traditionally, software engineers have not focused on building security into their systems. The main concern has been to make data and networks secure through peripheral security measures. Under the current circumstances, software companies are striving to include quality attributes such as security in the software design and analysis phase. The concept of *"building security into the software"* is a talked about topic in the industry, however, we do not see any actionable steps from the companies on how to make their existing and new systems adhere to the security requirements.

Sparked by the requirements of the industry, recent studies have shown an increase in cyber threats for real-life software (Ulsch, 2014). Researchers identify defensive mechanisms at the hardware and operating systems level to counter malware threats

(Lawson and Middleton, 2019). Software industries which produce in-house software have focused on deploying firewalls, whereas increasing number of software services are being provided online using the internet (Kumar et al., 2008). Security needs to be ensured for real-life software at the design and development stages to make them cyber-attack proof (Beznosov and Chess, 2008). Despite the requirement to make the software more secure, what current challenges prevent software engineers from building security into their systems need to be explored in greater detail.

In this paper, we described our approach and findings based on our investigation of how software teams looked into the current challenges which prevented the security of software applications. A survey participated by 9 respondents of a software company who were responsible for designing software that recently got compromised by a malware attack was conducted, and we aimed to determine what factors which prevented the secured design of the software. The 9 respondents were divided into two teams, who worked

[a] https://orcid.org/0000-0002-1780-0296

693

on different components of the compromised project. For purpose of anonymity, we cannot publish the names of the respondents, product or the company. We tried to identify if the developers and the managers shared the same views related to software security challenges, or do their views differed. We further explored what recommendations in terms of collaboration and design challenges suggested by developers and managers which would enable mitigation of the current challenges.

Under the current circumstances, it is important to identify the viewpoints of software developers and managers regarding security vulnerabilities in software. Motivated by the research in (Storey et al., 2022), we explored the perspectives of developers and managers on the reasons for malware vulnerability in software.

The survey of managers and developers highlighted that managers and developers agreed on certain criteria of security, however, they differed in their views on other criteria. We explored why being in the same cohort they had differing views on certain aspects of building security into their systems. Our findings showed that both developers and managers agreed that software can become vulnerable to malware attacks due to the *lack of skills* of the project team members who designed it, *negligence or ignorance* by end users, and *trade-off* between security and key aspects.

Developers and managers could not reach an agreement if *work environment and well-being* of the project team had a role to play in getting the team more motivated and designing security into the software systems. It was seen that although many developers stated the role of team cohesiveness towards designing secure software, managers emphasized more on building skills of developers.

The major contributions of this paper include the following:

- Survey of software developers and managers who worked in two groups to design a commercial software that was recently compromised by malware.

- An empirical evaluation of the common and differing views of developers and managers on what caused the malware attack to be successful on their software.

The rest of the paper is organized as follows: Section 2 explains the methodology of the research. Section 3 presents the survey design and selected responses. Section 4 highlights the analysis and findings of the survey and states important recommendations regarding developers and manager views of software security design. Section 5 discusses the related

work in this area, Section 6 identifies the threats to validity of this research, and Section 7 concludes the paper.

2 METHODOLOGY AND SURVEY SETUP

To determine the views of developers and managers on the cause of security vulnerabilities in software, we surveyed professionals from a large software company who worked on a product that recently suffered from cyber attacks. Our research questions were as follows:

1. How developers and managers define reasons why software suffer from malware attacks?

2. Are developer's view of security threats to a software aligned with each other and views of managers?

Our goal is to identify the views of developers and managers regarding software security. We try to find the recommendations of the developers and managers regarding mitigating security vulnerabilities in software design based on their real-life experience.

3 SURVEY DESIGN

To ensure that the survey questions were accurate, we worked closely with two developers and one manager who helped us to recursively design the questionnaire. We conducted the survey recursively and the team of software professionals helped us pilot it.

The survey was made available to the respondents in March 2022. Based on our prior association with the software company, we were able to contact the respondents via email. We contacted the developers and managers of two small teams within a large company, who combinedly designed and developed one product that faced cyber attacks on January 2022 and was compromised.

We advertised the survey to them as a general survey to obtain their views and recommendations for software security design. The survey was kept available for 20 days, and 2 managers and 7 developers responded. Our initial results showed that both developers and their managers had a clear concept of how each perceived software security in terms of user behavior, skills, and trade-off. However, they did not seem to agree on the perspective of integrating security into software design and its relationship to the work environment of the team.
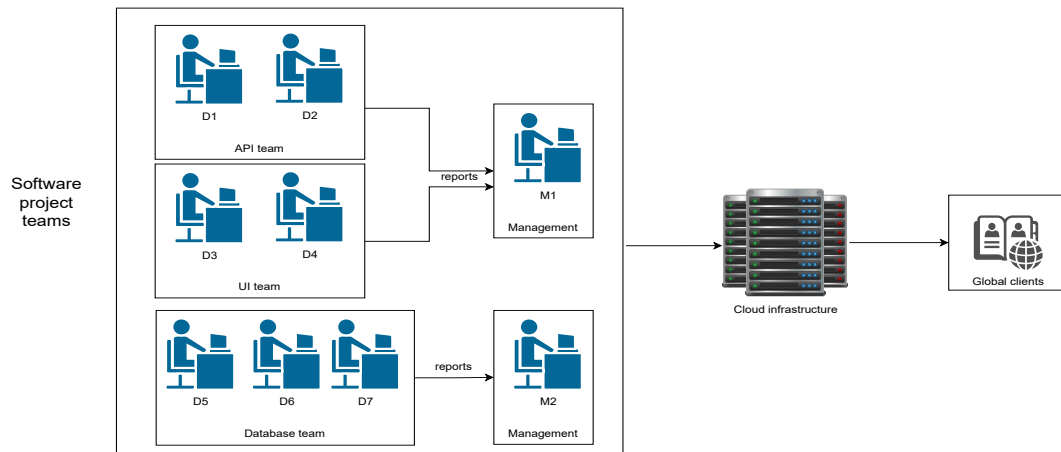
Figure 1: Project team structure for the surveyed software.

3.1 Survey Structure

We identified the respondents based on our prior association with the company. The respondents worked in the same software team that designed and implemented an *application interface (api)* that was used by the core product of the company which was a *Database Management System (DBMS)*.

The *api* acted as a middleware used by the DBMS to launch itself in the Windows platform. The team structure and roles of the 9 respondents were highlighted in Figure 1. This was a large software company, that served many energy and storage industries in the USA and abroad. The surveyed company and respondents have been kept anonymous for the purpose of security. We surveyed 2 managers who were leading the team which designed and developed the *api* that acted as a middleware between DB and OS. We surveyed 7 developers who were working under the managers, we tried to gather their views on software security design of the compromised *api*.

Based on our earlier relationship with the company we learned that the *api* that was designed had been compromised by *Azorult* malware that lurked in the host operating system of a client, thereby it successfully obtained access to the dashboard of the DBMS. *Azorult* malware was used by the attackers to infect the target software. This malware was first discovered in 2017 (Rendell, 2019).

Azorult is primarily used to confiscate user information on communication platforms like Discord. It is also used by the attacker to steal the browser history, cookies, and cryptocurrency keys of the victims. Finally, the data is sent back to the attacker via a zipped file. Recently, malware has been found that *Azorult* can load other malware and the recent versions have improved delivery mechanisms compared

to earlier ones.

Prior to data collection, we consulted industry experts in survey design and asked them to answer the survey as a test pilot. The survey included 16 questions that were conducted in a semi-structured format. We iteratively tried to determine developers and manager views. The responses were coded by the second and third authors and assigned to the the pre-specified codes. Any ties in coding were dissolved by the author. It took 9.10 minutes on average to complete which was an acceptable completion time, thus not consuming the excess time of the respondents.

The respondents included both developers and the manager of the *api* project. We aimed to determine the views of the developers and managers regarding any issues related to software engineering practices that led to a faulty design thus resulting in the malware compromise. We aimed to identify if the views of developers and managers differed in this regard and tried to determine *"why"* the difference existed.

Additionally, any recommendations by the respondents based on their experience during that process on how to securely design similar software for DBMS to thwart such attacks were discussed here. The survey instrument is shown in Figure 2. The questionnaire precisely asked the respondents to relate the cause of the detected cyber attack compromise to the reasons like *skills of the development team, user behavior, trade-off, and work environment*. The respondents provided their responses which show-cased their different viewpoints. We recorded all the responses and mapped those to the software attributes described above. The data were compiled and presented in the following section.

Survey questionnaire: obtainment of developer and manager views on software security challenges		
general (presented to both developers and managers)	Q1	What is your position in the current company?
	Q2	How long have you been working in the software industry?
	Q3	How many software teams (approximately) does the company have?
	Q4	Identify [1-3] security issue in terms of malware attacks which your software faced in the last 12 months?
	Q5	Do you think the short time to finish the project lead to some loopholes which the malware exploited? Please include your comments
	Q6	Do you think having to trade security with other aspects of design such as user-friendliness resulted in this vulnerability? Explain your views
	Q7	Do you think the malware impacted your system because of the end-user's ignorance or negligence? Please explain
	Q8	Do you think your working environment affects the way you build security into your code? Please explain
manager-specific	Q9	How many teams do you currently manage?
	Q10	How many engineers do you manage in your current teams?
	Q11	Do you think there were certain lack of skills in you or your team's part which resulted in this malware attack? Please include your comments
developer-specific	Q12	Do you think there were certain lack of skills in you or your team member's part which resulted in the malware attack? Please include your comments

Figure 2: Survey instrument to obtain viewpoints of developers and manager on software security challenges.

3.2 Survey Takeaways

We found that developers and managers agreed to some extent regarding the cause of malware attacks, however, they seemed not to agree on the steps which were required in software design to make sure the malware cannot attack in the first place. The open-ended responses were coded by us in an inductive manner, and initial codes namely *Skills (S), User behavior (U), Trade-off (T), and Work environment (E) (SUTE)* were refined after further discussion. Following are the codes for developers and the manager. Regarding skills, the following managerial feedback can be considered.

The two managers have been identified as *M1* and *M2* respectively, and the seven developers are identified as *D1, D2,...,D7* respectively. Regarding the code *S*, manager *M1* stated: *"yes, I personally think virus writers are better programmers than some of my team members, the issue is when we make a team, we select people who are good software engineers, however, to address security we should have a policy where all teams need cyber security experts"*. The response provided by *M2* is similar and notable, *"we design and develop large database apps that communicate with the host OS. Now, I have a big team of engineers working on our project and someone is always a weak link"*.

Regarding skills and expertise, developer *D1* stated that *"Malware developers continuously develop new ways to get around design to which is deployed to prevent malware. On the other hand, our skills are mostly applicable for after-attack situations.* Another developer, *D2* stated that *"we react to attacks, rather than focusing on predicting and preventing the malware attack in the first place"*. *D3* mentioned *"our teams only constituted of domain experts and software developers, we definitely lacked the inputs of a cyber security expert, this is something we are planning to include in our future projects"*. An-

other developer, *D6* noted that *"I primarily focused on finishing the different functional requirements of the api, which included incorporating the different features, never really got time to check if the features meet security guidelines"*.

Regarding user behavior, *M1* stated, *"Most malware got into our software through the OS because of the user's mistakes like clicking on suspicious links that allowed the Azorult to infect the system and ultimately gain access to our api. M2* mentioned that, *Users are "click robots", they are asked to click something, and they click it"*. From the developer perspective, we got feedback from *D5* such as *"users do not keep their systems up to date, because they are not IT people, they do not have IT people"*.

Also, another developer, *D7* responded by stating, *"They open attachments from people they do not know, or from people who pretend like someone they know"*. Developer *D1* mentioned *"The weakest vulnerability of our api was probably the human element. User behavior is the chief reason for creation of access points which were exploited by the Azorult"*. *D2* stated that *"I think besides developers and managers, general users with minimal technical knowledge should be trained about system health, awareness about suspicious behaviors, and creating strong passwords"*.

Tradeoff is another important consideration for security. According to manager *M2*, *"inevitably some decisions are made in favor of usability, rather than security, so there was a tradeoff between usability and security.* The reason for the tradeoff was mentioned as *otherwise people will not buy and use our solution"*. Developers also highlighted that quality requirements such as security was traded off for this project.

Developer *D4* mentioned that *"it is a cat and mouse game, think of building a house, why cannot architects and contractors build homes and business buildings that are impervious to burglars"*? *D3* responded *"amount of time I can spend doing actual*

development compared to attending team meetings, I had to tradeoff my productivity and time with attending meetings, some of which were not really useful".

Developer *D2* mentioned that *"we wanted to ensure least privileges in all aspects of the software, however, user-friendliness often came in the way and we had to make the software more easily accessible by the users, which was definitely a wrong decision in my opinion".* It was visible that tradeoff happened due to usability, ensuring least privileges, and attending team meetings in a large institution not directly related to the project.

Regarding environmental condition and personal well-being, manager *M1* noted, *"personally I think work environment does not impact security aspects of the software rather the focus is more on building developer skills".* Another response from manager *M2* was such that *"developers tend to focus on how many artifacts they produce in a sprint whereas I tell them to get the job done efficiently and preserve highest quality, we always do that in good team spirit so I think team environment is not a concern".* It can be inferred that both the managers believe that their teams have good working environments and there is no issue in this regard. We analyze the responses obtained from the developers in the following.

The developers had a different opinion regarding the personal well-being and work environment and their impact on including quality attributes in software design. Feedback from developer *D1* included *"When I feel engaged in the work I am doing, I do not feel held back and try to implement new security designs in the component I was developing".* Additionally, we obtained feedback from developer *D2* such as *"when I am truly engrossed in a task, I felt that I worried more about non-functional requirements like security.*

Developer *D3* stated, *"My manager seemed to avoid many questions I had regarding the api during the design and development stages, after some time, I lost a lot of motivation and tried my best to obtain the answers to my questions using the web. If my manager provided timely feedbacks and appreciated my efforts, I think we would have worked in a more cohesive environment".* *D4* mentioned that *"I personally was not impressed on how our team communication was maintained, we did not get feedback in a timely manner from other members, including developers and managers, so it was a 9 to 5 job where I was concerned with finishing my tasks to meet functional requirements only".* Finally *D7* said, *"I focus on building real software and gaining new knowledge, and this can only happen when I am working in a place where people care and provide timely an-*

swers to my queries". As a result, multiple respondents mentioned the requirement of a supportive team to build secured software.

4 ANALYSIS OF FINDINGS

The thematic analysis was conducted by one of the authors of the paper based on the responses submitted by survey participants during the interview. The survey questionnaire was prepared in an online form that was provided to the respondents. The data was exported in an excel file and the responses were recorded that were used to review and complete the findings. The exporting of the data into an excel file helped to harness the power of the survey tool that we used and it helped to leverage the design of our survey for analysis. Responses to each question were analyzed on a per-question method, which resulted in finding similarities across all responses and coding the results. The codes were provided in Table 1.

The survey had open-ended questions as seen in Figure 2. It was important to code the open-ended question to obtain correct takeaways from those. We proceeded to code the open-ended responses recursively and inductively. Two authors initially coded all the open-ended responses. Next, we conducted a dedicated session where all three authors analyzed the coded responses and reached an agreement. After multiple rounds of coding, 4 codes were determined to be applicable to analyze both developer and manager views. We proceeded to analyze the coded responses to determine any similarities or differences in the perspectives of developers and managers.

The four main codes that resulted from coding the productivity definition question are as follows: skills of the developers and managers, user behavior when using the software, trade-off between security and usability, and developer satisfaction and work environment. We noted that these four codes from recursive coding practice are in-line with the dimensions provided by the SPACE framework (Forsgren et al., 2021).

- *S*: both the developers and managers agree that there is a lacking in the current skill set of engineers and recommended new upcoming engineers to *"think out of the box and focus on developing malware threat prediction-based action skills"* rather than *"reactive skills"*. In this regard, the software teams should include cyber security experts in them.

- *U*: 100% of the respondents stated that user negligence has a role to play in software getting impacted by malware. One thing to remember is that

Table 1: Description of the codes.

Code	Description of code
S: skills	do not possess the skills required to remove all bugs which may result in security vulnerabilities
U: end user behavior	ignorant and negligent activities from end users result in malware attacks succeeding
T: trade-off	trade-off between completion of user stories to security
E: environment	impact of personal well-being and team cohesiveness towards designing a more secure software

malware primarily does not attack the software, it attacks OS first in most cases. How users use the OS is very important in this regard.

- *T*: over 78% of respondents identified trading security for multiple reasons that are highlighted in Figure 3. *Limited budget, exhaustive testing of security features, lack of collaboration with cyber security experts, strict time to market, company culture,* and *user requirements* are a few attributes that forces the team to tradeoff security for user-friendliness. Around 22% of respondents stated that they will not tradeoff security for usability as they stated *"software which follows secured design with the highest importance is likely to achieve higher user satisfaction in the long run than software that looks to reduce security for usability"*. The respondents also mentioned the importance of collaboration between development team and cyber security experts to achieve highly secured software.

- *E*: although developers and managers agree on the S, U, and T aspects of security design into their systems, they have different views when it comes to work environment. Developers addressed the relationship of team cohesive, caring work environment and building security into their systems. Developers defined *"building secured software"* as related to their work environment satisfaction whereas managers focused more on improving the skillset of developers to achieve this goal.

Developers and managers addressed software security challenges in terms of *SUT* codes, however, they seem to differ in their viewpoints regarding a cohesive work atmosphere. More specifically, developers and managers, as cohorts, were not in-line with their viewpoints regarding the impacts of a cohesive work environment and designing security into the software. Managers tend to view security design as dependent on the *SUT* codes rather than the *E* code. On the other hand, developers, emphasized the need for *E* code, which is a friendly work environment, as a critical aspect if addressing security in software design.
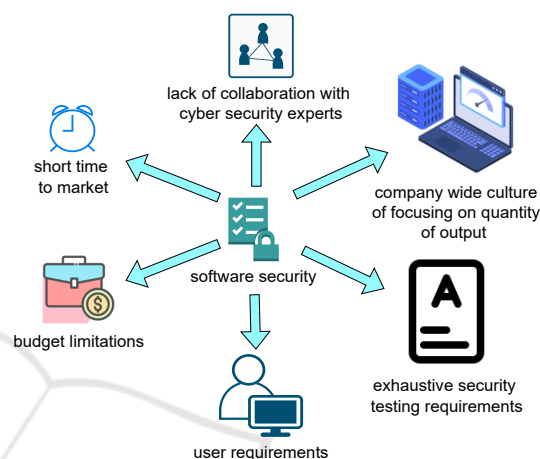

Figure 3: Identification of the different reasons according to developers and managers which lead to trading off security in software design.

5 RELATED WORK

The importance of methods, processes, and tools for improving the security of software created by a wide range of developers is emphasized in existing research (Beznosov and Chess, 2008). Modern software works mostly in a connected and collaborative manner, therefore they need flawless access to other systems in different geographic locations (Imran et al., 2013) (Imran et al., 2016). Security needs of modern-day software need to be addressed from the perspective of software design, developer skills, user behavior, and legislation governing software engineering (Kołodziej and Xhafa, 2011). At the same time, software engineers need to collect security requirements during the software requirements elicitation phase.

Security is mostly considered an afterthought and incorporated only after the software has been developed (Khan and Zulkernine, 2008). It was stated that there is a lack of processes to quantify the security of a *Software Development Life Cycle (SDLC)* artifact. As a result, the number of vulnerabilities was detected to calculate a vulnerability index of an *SDLC* artifact which acted as an indicator of the current vulnerabil-

ities. However, a mechanism to code the different areas of human factors which lead to the vulnerabilities in design is required.

Activities to solve software security issues are mainly considered after completion of the coding phase (Jürjens, 2005). Security professionals address security concerns by using tools like antivirus, proxies, firewalls, and intrusion prevention systems mainly during the final stages of *SDLC* (Jürjens, 2005). However, addressing security challenges after the coding phase may require more expensive reworks (Khan and Zulkernine, 2008). As a result, security needs to be considered at an early stage of *SDLC* through the incorporation of feedback from the project team.

Existing research focuses on the security of software primarily designed for military, government, and safety-critical systems where budget and time to market have less priority and security is the primary concern (Bressan et al., 2021). However, the focus should be also provided to software which is designed by software industries where cost, time to market, and usability are the main concerns, and security is considered a non-functional requirement.

Software security in earlier publications stated the importance of security practices such as access control, firewalls, encryption, and authentication (G. Kagombe et al., 2021). We agree that all security features are important, however, from a software engineering perspective, the processes and designs followed to build the software are equally important. Therefore, we think further research should be conducted in this regard.

The importance of secured software design was highlighted in (Zhang et al., 2022). The authors mentioned that following a structured security design practice during software development should be given importance. They argued that although redundancy can help during random cyber attacks, focusing on reliability only will not ensure security against well-planned and coordinated cyber-attacks since such attackers are capable of compromising redundant copies of the same system if they can compromise the deployed version.

The importance of collecting security requirements besides regular software requirements is emphasized in (Tondel et al., 2008). The authors stated that correct identification of security requirements will make sure that the software performs the correct operation. Whereas, wrong security requirements will lead to failure of performance by the software, which often leads to finger-pointing by the engineers. Also, if the software team lacks any security experts, then these problems will increase manifold. In addi-

tion to collecting software requirements, the process and time when to collect these requirements should be specified after discussion with developers and managers.

Threat analysis has been proposed as a critical activity to identify and prioritize different threats during software design and development (Ingalsbe et al., 2008). Given the situation where cost, usability, and time to market take greater importance to security, an effective threat analysis mechanism will enable software professionals to detect the most critical threats and resolve those with the given budget and time. However, besides threat analysis, importance should be provided to gathering real-life experiences of both developers and managers to prioritize threats in future software.

6 THREATS TO VALIDITY

We identify the common threats to validity as experienced by similar qualitative research. Generalization of the survey results beyond the scope of the sampled respondents needs to be conducted with care. We interviewed only industry experts and not the clients of the software, also we interviewed the teams related to designing and developing the software, the opinion may vary among teams working on different software projects in the same company. We limited our scope to one software company only, however, the findings may not be reflected across different sized companies at different geographic locations. Furthermore, our number of respondents was small, and hence our results may not be generalized to other software teams. More specifically, the number of managers we interviewed was low.

7 CONCLUSION

We determined that developers and managers had different viewpoints regarding the work environment and its impact on designing security into software. Based on the findings regarding the misalignments between the views of developers and managers about quality, we suggested that the development team and managers should regularly communicate their views on security design and state what the work environment means to them. We suggest, even before obtaining security requirements and starting a security-focused software design process, developers and managers should first communicate what security level they want the software to achieve. At the same time, they should openly discuss the necessary

skills, tools, and work environment which is required to build security into the software.

We also recommend software companies use our provided coding mechanism called *SUTE* to evaluate the security challenges faced by their software teams. Although the codes have been tested on two teams in one company, we believe they should be piloted across different projects to establish our recommendations. Our coding mechanism reveals a misalignment between developers and managers when it comes to a working environment and security design. Similar findings have been stated in earlier research efforts (França et al., 2018), (Graziotin et al., 2013). Hence, we believe the proposed coding technique can help companies evaluate security challenges in existing projects.

At the same time, our coding mechanism can be used with the popular *SPACE* framework as discussed earlier in the paper. Using such a framework will enable practitioners to identify a wide range of metrics that will enable practitioners to capture the requirements and challenges of building security into the software concerning their company and project. We would also like to extend our research by incorporating feedback from more respondents and increasing the number of project teams. Including a greater number of respondents will enable us to reach more concrete answers to our research questions.

ACKNOWLEDGEMENTS

REFERENCES

Beznosov, K. and Chess, B. (2008). Security for the rest of us: An industry perspective on the secure-software challenge. *IEEE Software*, 25(1):10–12.

Bressan, L., de Oliveira, A. L., Campos, F., and Capilla, R. (2021). A variability modeling and transformation approach for safety-critical systems. In *15th International Working Conference on Variability Modelling of Software-Intensive Systems*, pages 1–7.

Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., and Butler, J. (2021). The space of developer productivity: There's more to it than you think. *Queue*, 19(1):20–48.

França, C., Da Silva, F. Q., and Sharp, H. (2018). Motivation and satisfaction of software engineers. *IEEE Transactions on Software Engineering*, 46(2):118–140.

G. Kagombe, G., Waweru Mwangi, R., and Muliaro Wafula, J. (2021). Achieving standard software security in agile developments. In *2021 The 11th International Conference on Information Communication and Management*, pages 24–33.

Graziotin, D., Wang, X., and Abrahamsson, P. (2013). Are happy developers more productive? In *International Conference on Product Focused Software Process Improvement*, pages 50–64. Springer.

Imran, A., Aljawarneh, S., and Sakib, K. (2016). Web data amalgamation for security engineering: Digital forensic investigation of open source cloud. *J. UCS*, 22(4):494–520.

Imran, A., Ul Gias, A., Rahman, R., and Sakib, K. (2013). Provintsec: a provenance cognition blueprint ensuring integrity and security for real life open source cloud. *International Journal of Information Privacy, Security and Integrity*, 1(4):360–380.

Ingalsbe, J. A., Kunimatsu, L., Baeten, T., and Mead, N. R. (2008). Threat modeling: diving into the deep end. *IEEE software*, 25(1):28–34.

Jürjens, J. (2005). *Secure systems development with UML*. Springer Science & Business Media.

Khan, M. U. A. and Zulkernine, M. (2008). Quantifying security in secure software development phases. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 955–960.

Kołodziej, J. and Xhafa, F. (2011). Meeting security and user behavior requirements in grid scheduling. *Simulation Modelling Practice and Theory*, 19(1):213–226.

Kumar, N., Mohan, K., and Holowczak, R. (2008). Locking the door but leaving the computer vulnerable: Factors inhibiting home users' adoption of software firewalls. *Decision Support Systems*, 46(1):254–264.

Lawson, S. and Middleton, M. K. (2019). Cyber pearl harbor: Analogy, fear, and the framing of cyber security threats in the united states, 1991-2016. *First Monday*.

Rendell, D. (2019). Understanding the evolution of malware. *Computer Fraud & Security*, 2019(1):17–19.

Storey, M.-A., Houck, B., and Zimmermann, T. (2022). How developers and managers define and trade productivity for quality. In *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*, pages 26–35.

Tondel, I. A., Jaatun, M. G., and Meland, P. H. (2008). Security requirements for the rest of us: A survey. *IEEE software*, 25(1):20–27.

Ulsch, M. (2014). *Cyber threat!: how to manage the growing risk of cyber attacks*. Wiley Online Library.

Zhang, Y., Xiao, Y., Kabir, M. M. A., Yao, D., and Meng, N. (2022). Example-based vulnerability detection and repair in java code. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, pages 190–201.