Follow-up Study

ANONYMOUS AUTHOR(S)

The logical underpinning of the awake lock absence vulnerability is not complicate; essentially, it stems from the programmer's omission of a specific security function that should have been invoked within some interfaces. However, why do such vulnerabilities appear to be relatively common in privacy-sensitive applications where security should be paramount? In order to investigate the causes of the vulnerability, we conducted a semi-structured interview study with technical experts in application development.

1 METHOD

We have sent interview requests to the vendors of the vulnerable applications, but regrettably, we have not received any response. To address this, we resorted to snowball sampling to make our best to recruit qualified technical experts as alternatives. We first interviewed experts we knew and then asked them to recommend additional experts. In total, we recruited 21 experts from nine different organizations, as shown in Table 1. Note that 14 of them were involved in developing applications with the awake lock function. The experts' job type included architect (n=2), development engineer (n=14), quality assurance (n=3), and independent engineer (n=2). All of the recruited experts had considerable experience in mobile application development and testing, with an average of 10.2 years (SD=4.6) of professional experience. Seventeen of the experts came from Fortune 500 companies.

We conducted face-to-face interviews with 17 experts, while the remaining four were interviewed over the phone due to scheduling constraints. All interviews were recorded for reference. The interviews were conducted in a semi-structured manner, using a set of basic questions tailored to each expert's background and responses.

Initially, we investigated the experts' understanding of awake lock vulnerabilities. Subsequently, we delved into the potential causes of such vulnerabilities, with a focus on the design, development, and testing phases of the application. Finally, we requested the experts to provide technical recommendations for mitigating these vulnerabilities based on their professional expertise. Each expert interview spanned around 20 minutes, and as a token of appreciation, we provide a $5 souvenir. The study was approved by our university's IRB.

The interviews were transcribed, and a thematic analysis approach was employed to identify common patterns and themes within the responses. To ensure unbiasedness, two researchers independently analyzed the data. Any disagreements in the identified themes were resolved through group discussions involving another two researchers.

2 RESULTS

2.1 Awareness of the Vulnerability and Its Implication

Among the experts, 19 individuals demonstrated an understanding of awake lock mechanisms. Only two experts had no prior knowledge of the awake lock. Surprisingly, a majority of the experts were unaware of the existence of awake lock vulnerabilities. After demonstrating the detected vulnerabilities to the experts, 17 of them admitted that they had not been aware of such vulnerabilities before.

> "*To be honest, I didn't really realize the problem.*" -P9

When questioned about the implications of awake lock absence, 13 experts immediately acknowledged the significant impact of the vulnerability.

Table 1. Expert Demographics.

ID	Job Type	Organization Type	Years of Experience	ID	Job Type	Organization Type	Years of Experience	ID	Job Type	Organization Type	Years of Experience
P1	Architect	Large Hardware and Software	14	P8	Development Engineer	Large Internet Technology	12	P15	Development Engineer	Large Internet Technology	7
P2	Architect	Large Technology Infrastructure	18	P9	Development Engineer	Large Internet Technology	8	P16	Quality Assurance	Large Internet Technology	13
P3	Development Engineer	Large Internet Technology	14	P10	Development Engineer	Large Internet Technology	5	P17	Quality Assurance	Large Internet Technology	5
P4	Development Engineer	Large Internet Technology	14	P11	Development Engineer	Large Internet Technology	8	P18	Quality Assurance	Large Internet Technology	14
P5	Development Engineer	Large Internet Technology	20	P12	Development Engineer	Large Internet Technology	11	P19	Development Engineer	Small Software	8
P6	Development Engineer	Large Internet Technology	13	P13	Development Engineer	Large Internet Technology	5	P20	Independent Engineer	Independent Engineer	5
P7	Development Engineer	Large Internet Technology	9	P14	Development Engineer	Large Internet Technology	7	P21	Independent Engineer	Independent Engineer	4

> "*My phone may be lent to others for temporary use, and they may inadvertently or deliberately access apps with awake lock. In the absence of awake lock, they could potentially access the app without my authorization, send random messages, or even compromise my data.*" -P21

Some interviewees even believed that the lack of awake lock could lead to financial losses.

> "*Without awake lock protection, important information, including property, could be compromised.*" -P5

However, initially, eight experts stated that the absence of awake lock posed minimal harm. Upon further discussion, we discovered that the main reason was their oversight of scenarios involving phone sharing.

> "*Phones are personal devices that we carry with us, so it wouldn't have a significant impact.*" -P8

After explaining the protective function of awake lock during phone sharing, all the eight experts also acknowledged the potential risks associated with its absence.

> "*When lending our phones to others temporarily, it is indeed necessary to have lock protection for personal photo albums and finance-related applications; otherwise, it may result in personal privacy leakage.*" -P8

Digest. Many application developers have a limited understanding of awake lock absence and its potential risks. The majority of experts (17/21, 81%) were oblivious to the presence of the vulnerability in applications. Even after demonstrating the identified vulnerabilities, about one-third of experts remained unaware of the harm caused by the vulnerability. However, after explicitly explaining the scenario of phone sharing, all of them recognized the potential significant risks involved.

2.2 Causes of the Vulnerability

The main objective of our interview study was to explore the causes that lead to awake lock vulnerabilities. The interviewees attributed the vulnerability to issues in requirement analysis and development specification.

Incomplete Requirement. All interviewees acknowledged the significant relationship between the vulnerability and incomplete application requirement design, which resulted in the omission of awake lock implementation requirements in certain interfaces.

> "*As long as it is clearly stated in the requirements, it is impossible for the development team not to implement it.*" -P2

Furthermore, nine experts highlighted that incomplete requirement were primarily attributed to insufficient security awareness among analysts who overlooked security-related needs.

> "*Product design typically prioritize core business functionalities, and the significance of security-related features may not be emphasized to the same extent.*" -P5

> "*Since requirements are determined by individuals, a lack of understanding regarding awake lock vulnerabilities naturally leads to the omission of relevant requirements during the development of certain application modules.*" -P3

Besides, six experts identified incomplete requirements as stemming from the neglect of specific application usage scenarios, which they referred to as "scenario omission" within their development practices.

> "*When an architect decomposes requirements, it is crucial to comprehensively outline all access scenarios to privacy interfaces. Then, developers can align their development efforts with the various usage scenarios. If a scenario is inadvertently overlooked, it may result in the omission of certain functionalities during the development process.*" -P1

Unclear Development Specifications. Unclear development specifications were identified as a cause for the vulnerability by six experts. They highlighted that, even in large leading IT enterprises, the software engineering standards are not strictly followed during the application development due to time constraints and other factors. Furthermore, verbal communication is heavily relied upon in developing, leading to misalignment with requirements.

> "*Verbally or dictated requirements frequently occur without clear documentation. This documentation deficiency can result in information gaps in application development.*" -P2

The experts further noted that this problem is particularly prevalent during software upgrading, as developers may not adhere to initial development specifications. The vulnerability found in the QZone interface of QQ corroborates this point. We found that it does not exist in the old version of QZone. This implies that the upgrade process of QQ lacks a clear and unified specification.

> "*The initial product specifications may have been clear, but as more features were added, some modules may not have adhered to the previous design. As awake locks are not considered a critical feature and are often overlooked, bugs can occur.*" -P20

Digest. The main reason for the vulnerability is incomplete requirement. Security features, were not given enough attention compared to business functionalities, particularly when there was insufficient awareness of the vulnerability. Additionally, real-world application development does not always conform to rigorous development processes, and lacks clear specifications. Some developing practices still depend on human communication. This can increase the likelihood of failing to fully meet certain security requirements.

2.3 Reasons for Missing the Vulnerabilities During Testing

Experts argue that the vulnerability exhibits clear manifestations, and the testing should be able to discover such a vulnerability before releasing a product. The interviewees attributed the missing of the vulnerability to two main factors.

Incomplete Test Case Coverage. Thirteen experts argue that the fundamental reason for failing to detect the vulnerability lies in poor test case coverage. Seven among them further pointed out that incomplete requirement analysis directly leads to the absence of necessary cases for testing.

> "*The phenomenon of awake lock omissions is quite apparent and easily reproducible. It should not be missed as long as proper testing is conducted. Therefore, the test suit lacks specific evaluations targeting awake locks for certain interfaces.*" -P6

> "*Our current workflow involves conducting requirement analysis first, followed by testing analysis. However, it is possible that some scenarios were overlooked during the development phase, resulting in testers not receiving information about them and subsequently excluding them from the tests. The vulnerabilities are likely to go unnoticed.*" -P7

Testing Costs. Fourteen experts highlighted the significant testing workload and complexity of the target application.

> "*A typical reason is the substantial workload involved. Due to the complex navigation paths of the application, such as jumping from the home screen, task scheduler, or notification bar messages, there are multiple entry points and covering all paths poses a considerable challenge. The workload becomes overwhelming.*" -P4

> "*The existing automated testing tools have limited capabilities, and interface interaction testing still relies on manual efforts. It is easy to overlook certain interfaces, resulting in missed vulnerabilities.*" -P3

Digest. The vulnerabilities can be discovered with targeted testing. However, the test cases may not be comprehensive enough, which is often due to incomplete requirements. Furthermore, the complexity of applications also adds to the significant workload of testing, and manual efforts still play a critical role in addressing such complexities.

2.4 Potential Mitigation Strategies for Addressing the Vulnerability

The experts provided clear and actionable advices to prevent the vulnerability. Fourteen experts emphasized the importance of ensuring the completeness of requirements.

> "*How do we avoid this? It is crucial to thoroughly analyze all scenarios during the requirements analysis phase. This enables comprehensive developing and testing that covers all aspects of the requirements.*" -P7

Eight experts have expressed that requirement alignment is crucial in achieving security objectives. Several experts also highlighted the significance of frequent face-to-face communication alongside documentation.

> "*First, the product team needs to hold a meeting with the architect to present what they want to do. Then, the architect conducts thorough analyses and provides a feedback presentation to the product team, which is like a handshake. Once the handshake is done, the development department proceeds with detailed design and development. After completion, the architect presents the prototype and test results. The product team then provides us with another round of feedback presentation based on their own understanding, which is like a reverse handshake. This means there are at least two rounds of presentation and feedback presentation, resulting in multiple handshakes.*" -P1

However, experts also pointed out that due to cost reason, the actual development process is often not fully complete.

> "*Many IT companies prioritize rapid iterations and agile development, resulting in deficiencies in the overall workflow, even in prominent companies. The more complex the processes, the higher the costs.*" -P1

Regarding testing, experts specifically point out the need for further improvement in the level of automated interface testing techniques to enhance the capability of interaction testing.

> "*Popular automated interface testing tools like Monkey in the industry struggle to handle testing for complex applications. They still partially rely on manual intervention, lack an understanding of interface semantics, and urgently need to improve their level of intelligence.*"-P3

Digest. To effectively address the vulnerability, it is crucial to prioritize the completeness and alignment of requirements within the development process. However, achieving this in practice requires more processes and face-to-face communication, which can incur higher costs and are not always guaranteed. Additionally, leveraging automated interface testing techniques proves vital in enhancing test coverage and overall efficiency.

3 SUMMARY

The interview study highlights human factors as the primary factor to awake lock absence vulnerabilities, while technical factors played a secondary role. The results reveal that most developers lacked awareness of the awake lock vulnerabilities, resulting in incomplete requirement analysis and a failure to implement essential security features. This problem further extended to the testing phase, where the vulnerability remained undetected. Additionally, ensuring requirement alignment incurred high costs, including inevitable face-to-face interactions, which even top-tier enterprises often struggled to guarantee. Note that we believe the gained insights also hold applicability for other emerging vulnerability types.

Based on the investigation, we conclude the following recommendations for vendors to mitigate awake lock absence vulnerabilities and other newly emerged ones.

- Provide continuous security training for designers and developers, enabling them to stay abreast of evolving vulnerability techniques and enhance their awareness of security concerns. This will effectively prevent the occurrence of incomplete security requirements.
- In the foreseeable future, verbal interactions will continue to be an integral part of software development. Augmenting technical support for interpersonal communication, such as intelligent meetings [1, 5] and automated oral conversation recording and summarization [2, 6], is essential to align requirements across stages.
- Introduce more intelligent automated testing techniques, such as interface testing based on large language models [3] and automation of interactive guidance [4], to reduce manual workload and enhance testing efficiency and coverage.

REFERENCES

[1] Senthil Chandrasegaran, Chris Bryan, Hidekazu Shidara, Tung-Yen Chuang, and Kwan-Liu Ma. 2019. TalkTraces: Real-time capture and visualization of verbal content in meetings. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–14.

[2] Tae Soo Kim, Seungsu Kim, Yoonseo Choi, and Juho Kim. 2021. Winder: linking speech and visual objects to support communication in asynchronous collaboration. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–17.

[3] Zhe Liu, Chunyang Chen, Junjie Wang, Xing Che, Yuekai Huang, Jun Hu, and Qing Wang. 2023. Fill in the Blank: Context-aware Automated Text Input Generation for Mobile GUI Testing. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 1355–1367.

[4] Zhe Liu, Chunyang Chen, Junjie Wang, Yuekai Huang, Jun Hu, and Qing Wang. 2022. Guided Bug Crush: Assist Manual GUI Testing of Android Apps via Hint Moves. In *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022*, Simone D. J. Barbosa, Cliff Lampe, Caroline Appert, David A. Shamma, Steven Mark Drucker, Julie R. Williamson, and Koji Yatani (Eds.). ACM, 557:1–557:14.

[5] Samiha Samrose, Daniel McDuff, Robert Sim, Jina Suh, Kael Rowan, Javier Hernandez, Sean Rintel, Kevin Moynihan, and Mary Czerwinski. 2021. Meetingcoach: An intelligent dashboard for supporting effective & inclusive meetings. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.

[6] Naama Tepper, Anat Hashavit, Maya Barnea, Inbal Ronen, and Lior Leiba. 2018. Collabot: Personalized group chat summarization. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 771–774.

A BASIC INTERVIEW QUESTIONS

- What are your responsibilities in the application development process?
- Did you previously know about the awake lock mechanism?
  - Was the awake lock mechanism implemented in any of the applications you developed before?
- [Demonstrating the awake lock absence vulnerability] Were you previously aware of the awake lock absence vulnerability illustrated in the demonstration?
- Do you believe that this vulnerability poses a substantial security risk?
  - If so, what specific security risks could arise from it?
  - If not, in the context of phone sharing scenarios [Explaining phone sharing scenarios], does this vulnerability present a significant security risk?
- In your opinion, what are the underlying causes that lead to the introduction of the vulnerabilities during the application development process?
- From the perspective of application testing and analysis, is the vulnerability challenging to detect?
- Which method is suitable for discovering this vulnerability?
- In your perspective, what factors contributed to the oversight of this vulnerability during the testing and analysis of applications, leading to its emergence in the final product release?
- What are some potential mitigation strategies that can be implemented to address this vulnerability?