Necessity of Secure Coding Practices: An Analysis

Paul Cruz, Madhuri Palanivelu, Alex Yang

1. ABSTRACT

Software is the heart of all technological applications. Protecting applications from cyber-attacks and security breaches is a definite necessity. Coding applications securely from the beginning can help prevent exploitation, security breaches and the overall compromise of the application. One such organization that faced such an issue is Huawei. Huawei is one of the world's leading providers of information and communications technology infrastructure and smart devices. Unfortunately, with their 5G initiative, many speculations arose regarding security threats and spying allegations. This paper will discuss several of their flaws in the security domain and the different kinds of secure coding practices that could be integrated into their industry. We will also focus on the current research initiatives in the security domain and their breakthroughs in secure coding practices.

2. INTRODUCTION

Software vulnerabilities are flaws or glitches that are ever present in software or in Operating Systems. These vulnerabilities can pose as a threat to the software if exploited and can get severe if left unattended. This is the reason why secure coding practices need to be followed to guard the software against them. Many vulnerabilities fall under certain categories that a single secure coding practice can solve all at once. So we will see how these practices can help reduce the overall technical debt in going back to address security issues and make our software less prone to security threats from the very beginning. Though Huawei has been advancing on the hardware, software and telecommunication fronts, it is still struggling partially because of its security issues. This proves how much of an important area of software engineering, secure coding practices are. This paper will aim to address several important solutions that have been and are currently being developed, to promote secure coding and keep technology secure.

3. ORIGINS

3.1 Origination from Information Security

The requirement for secure coding practice originates from threats involving information security. In terms of a technical standpoint, Information Security is to protect the availability, accuracy, authenticity, confidentiality, integrity, utility and possession of information (Whitman and Mattord, 2004). Information security developed into several branches including, but not limited to Cybercrime, Cyberwarfare, Network Security, Mobile Security, Automotive Security and Internet Security. Threats that can ensue as a result are viruses, malware, exploits, eavesdropping, vulnerabilities, spyware, trojans, etc. Though there are several methods of defense like encryption, multi-factor authorization and firewalls, when it comes to the application level of security, it is best to follow secure coding practices that will prevent such security threats to begin with.

3.2 Initial Information Security Research and Results

(Chen et al., 2007) discuss mitigation of information security risks by increasing user security awareness. They discovered that 67% of the organizations they spoke with, had inadequate or dangerously inadequate levels of information security awareness. So their study revolved around creating an Information Security Awareness System (ISAS) to help increase information security awareness in the employees of those organizations. The Systems Development Research Methodology was followed to create their system. For their system research, they used the following International Organization for Standardization (ISO) security guidelines (McAdams, 2004):

1. Information security policy

2. System access control
3. System development and maintenance
4. Personnel security
5. Physical and environmental security
6. Security organization
7. Asset classification and control
8. Communications and operations management
9. Business continuity management
10. Compliance

After introducing System Architecture functions like Incident Management, Evaluation Management and Awareness Activity Management and User functions like providing mini-courses, news and articles for the users, they achieved the resulting ISAS system. Based on interviewing several employees after using the system, they found several themes which may serve as guidelines for building future security awareness systems:

- ❏ Flexibility of system design,
- ❏ Authentication management,
- ❏ Webpage layout, etc.

3.3 Initial Secure Coding Research and Results

(Jones, Rastogi, 2006) discuss interweaving Security into the Software Development Lifecycle (SDLC) by including Risk Assessment and Mitigation, Threat Modeling and Security Reviews to the design phase of the life cycle. They then cover how the development phase requires developers to constantly follow secure coding guidelines in their work from scratch.
Here are several categories of secure coding guidelines suggested:

1. Perform Input Validation
2. Assume a Hostile Environment
3. Use Open Standards
4. Use Trusted Components
5. Protect Data in Process, Transit and Storage with Safeguards
6. Always Authenticate
7. Native Security Protections
8. Fail Securely
9. Log, Monitor and Audit
10. Accurate System Date and Time
11. Least Privilege
12. Exception Handling
13. Strong Cryptography
14. Random Numbers

As a result of using secure coding practices and integrating security into the development life cycle, they mention how this will result in great strides for information security in general and they show how this can be most effective as an iterative and evolutionary process. They promise better security in the SDLC by making sure of the following:

- ❏ Higher management of a company supports implementing better security measures,
- ❏ Introducing security into the SDLC in phases,
- ❏ Assessing the current level of secure coding practices in the company and building on it,
- ❏ Training the developers to follow the secure coding guidelines suitable to the company standards and
- ❏ Introduce metrics to ensure good performance of security evolution

4. STATE OF THE ART

Several recent breakthroughs in the area of developing secure coding practices and guidelines, include the following:

4.1 Secure Coding Practices in OOP languages like Java:

In the research discussed by (Na Meng et al., 2017), they studied numerous StackOverflow posts to understand developers' concerns on Java secure coding, their programming challenges, and common security vulnerabilities.

They ultimately discovered that insecure coding amongst developers was prevalent at an alarming rate and they recommended the following:

- ❏ For Developers: Perform Security testing frequently and never disable security checks. Enforce caution while following reputed StackOverflow posts as they could be insecure or outdated.
- ❏ For Tool Builders: Develop automatic tools to diagnose security errors, locate buggy code, and suggest security patches or solutions. Build vulnerability prevention techniques, which compare similar applications that use the same set of APIs to warn potential misuses.
- ❏ For Library Designers: Deprecate the APIs whose security guarantees are broken. Design clean and helpful error reporting interfaces.

4.2 Secure Software Development Framework (SSDF) for mitigating software vulnerability risks:

(Donna Dodson et al., 2019) discuss their white paper where they introduce the SSDF as a collection of well-established secure coding guidelines and practice documents. The list of practices they curate was designed to be used by any kind of organization regardless of their cybersecurity sophistication and be easily integrable into their software development process.

This SSDF document containing secure coding guidelines was the result of their research and they categorized every practice with a brief explanation of the practice, the set as tasks required to accomplish it, an implementation example and finally some references for it. These are several of them:

- ❏ Define Security Requirements for Software Development,
- ❏ Define Criteria for Software Security Checks,
- ❏ Protect All Forms of Code from Unauthorized Access and Tampering,
- ❏ Provide a Mechanism for Verifying Software Release Integrity,
- ❏ Verify Third-Party Software Complies with Security Requirements,
- ❏ Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality,
- ❏ Create Source Code Adhering to Secure Coding Practices,
- ❏ Configure the Compilation and Build Processes to Improve Executable Security,
- ❏ Analyze Vulnerabilities to Identify Their Root Causes, and more.

5. ANALYSIS

Upon analyzing Huawei's 2019 Annual Report, we were able to concur that it's software and telecommunication initiatives were facing some backlash in regard to security. Many large scale and complex projects like those of Huawei's may require some of the efficient secure coding guidelines discussed. Here are some of the issues that were analyzed as part of the security flaws that Huawei might need to work on in the future:

5.1 Configuration Management:

Huawei's program implementations seem to contain different versions of the same components and libraries. They also mention having circular dependencies and outdated and soon-to-be unsupported components and libraries as well. For such complex projects, this is insecure as it can allow attackers to exploit errors that arise from these issues, and this will ultimately make their applications vulnerable to security threats. Some configuration management tools that can be used to help eliminate these issues are Ansible and Chef. By using tools like these, the code will automatically follow the secure coding guidelines pertaining to configuration management like updating code, keeping consistent documentation and maintaining integrity between software updates. (Johnson et al., 2011) discussed maintaining a Security-focussed Configuration Process (SecCM) that can be integrated into the SDLC project at hand. It will contain the following phases:

- ❏ Planning: involves developing policy and procedures to incorporate SecCM into the application lifecycle.
- ❏ Identifying and Implementing Configurations: a secure baseline configuration for the information system is developed, reviewed, approved, and implemented.
- ❏ Controlling Configuration Changes: emphasis is put on the management of change to maintain the secure, approved baseline of the information system.
- ❏ Monitoring: validating that the program is adhering to organizational policies, procedures, and the approved secure baseline configuration.

5.2 Third-Party Component Support Issue:

Using unsupported third-party components (TPC) within a software can be dangerous as the providers may decide to no longer provide security patches. Huawei might need to follow secure lifecycle management practices to monitor the end of life for the components they use. The whitepaper by (Bisht et al., 2017) mentions how users of third-party components should establish internal guidance for how to determine whether a TPC has reached the end of its life (EOL) and what to do when a TPC reaches it. They also provide a set of recommended software which can be used to get detailed reports on the TPCs used in the application like BinDiff, Nexus, etc. Some techniques to mitigate vulnerabilities in case they exist in an active TPC are:

- ❏ Patch/Update the Version
- ❏ Replace with an Equivalent
- ❏ Branch Code Internally
- ❏ Contribute to Community/Vendor
- ❏ Mitigate Through Code
- ❏ Accept Risk

5.3 Wider Component and Lifecycle Management Issue:

This issue leads back to Huawei's configuration management issue where they found that certain common components, libraries and their own operating system needs to be updated, in order to prevent vulnerabilities. This proves an overall issue with their engineering and lifecycle management process. With their advancements towards building an ecosystem involving chip device cloud capabilities in their devices, a Data Lifecycle Management (DLM), as suggested by the research conducted by (Li et al., 2015), could be beneficial. It is an approach that can be policy based and refers to the managing of an information systems data throughout its life cycle; from creation of the initial storage to the time when it becomes obsolete and is therefore deleted. One such DLM tool that can be used is Hierarchical Storage Management (HSM) which automates the management of the software.

5.4 Further Breakthroughs:

Based on the research work discussed, there are several futuristic results that can be implemented or are already in the works:

1. While incorporating secure coding practices into our code from scratch is required, being able to save manpower and avoiding technical debt in case of errors found later in the applications would always be welcome. Automated or semi-automated security bug detection tools are in development in companies like Veracode. Yet it would be great if a system could detect a larger multitude of security errors in the source code. A way to enforce that is by using Artificial Intelligence to detect all kinds of errors that have

occurred till date. Data is abundant and ever growing so AI systems would be able to distinguish threatening errors from benign ones and find more threats that couldn't have been detected with the naked eye.

2. Systems like ISAS (Chen et al., 2007), which are systems that are trying to provide awareness about user security in organizations, can be further implemented to reach people who do not have a technical background, so that they do not unknowingly become causes or victims of security breaches.

6. CONCLUSION

The conclusions drawn from our analysis show that though many efforts are being made on finding better secure coding guidelines, a lot of ongoing research and products are inclined towards creating secure analysis systems and software and not much research is being done to develop more guidelines. Nonetheless, in this age of technology, this is a vital and necessary step and it helps integrate the guidelines a lot better into ongoing IT projects and applications. Here is a summary of the kind of breakthroughs we analyzed, that are being made in this area of research:

- ❏ User interfaces for spreading security awareness are being developed,
- ❏ Few detailed secure coding guideline documents are being made for developers of most organizations,
- ❏ Security based measures are being included into the Software Development Lifecycle,
- ❏ Automated security bug detection tools are being developed.

We also performed an analysis on the information and telecommunications technology leader, Huawei, to try to understand a couple of the security based issues they recently faced and discuss what kind of secure coding guidelines or secure coding systems would be beneficial for complex and widespread applications like theirs. Based on the level of negative impact that arose from their security problems, we were able to deduce how important secure coding guidelines are and how necessary it is to work with secure analysis software to further reduce the difficulties.

7. REFERENCES

Whitman, Michael E., and Herbert J. Mattord. "Making Users Mindful of IT Security." Security Management 48.11 (2004): 32-35.

Chen, Charlie C. et al. "Mitigating Information Security Risks by Increasing User Security Awareness : A Case Study of an Information Security Awareness System." (2007).

McAdams, A. C. Security and risk Asynchronous Learning Networks, 5(2), 111-127. management: A fundamental business issue. Information Management Journal, 38(4), (2004): 36-44.

Jones, Russell L. and Abhinav Rastogi. "Secure Coding: Building Security into the Software Development Life Cycle." *Information Systems Security* 13 (2004): 29 - 39.

Na Meng et al. "Secure Coding Practices in Java: Challenges and Vulnerabilities." (2017).

Donna Dodson et al. "Mitigating the Risk of Software Vulnerabilities by Adopting a Secure Software Development Framework (SSDF)." (2019).

Arnold Johnson et al. "Guide for Security-Focused Configuration Management of Information Systems" (2011): 800-128

Prithvi Bisht et al. "Managing Security Risks Inherent in the Use of Third-party Components." (2017)

Li, Jingran, et al. "Big Data in Product Lifecycle Management." *The International Journal of Advanced Manufacturing Technology*, vol. 81, no. 1-4, (2015), pp. 667–684.