Jeremy Baldner

CS-405-R3362 Secure Coding 24EW3

Professor Sarah North

March 3, 2024

Journal

**Adoption of a secure coding standard, and not leaving security to the end**

One of the best practices for secure coding is, “Don’t leave security to the end.” Starting with security is a proactive approach that has been proven to better protect applications and their data. Implementing security fundamentals in the initial design can reduce technical debt, prevent breaches, and better control data (Federal Trade Commission, 2015).

To take a proactive approach, a developer needs to pay attention to data collection, access control, and implementation of a security policy. The data that is collected should be pertinent to the application. Do not collect unnecessary data and delete data that is no longer relevant. This reduces the data available to a hacker and prevents collateral damage from how they decide to use it (Federal Trade Commission, 2015).

Access control should utilize the Principle of Least Privilege. Users should only be given the minimum amount of access that they need to complete a task. This will minimize the areas for attack and reduce the impact of error. Secure passwords should also be enforced to prevent vulnerability from password-guessing tools (Federal Trade Commission, 2015).

Creating documentation for coding standards is important to keep the development staff up to date on the best practices. Enforcing coding standards can guide the developers in decisions throughout development. During the maintenance phase developers can also reference the security policy to help handle new vulnerabilities or a breach (Federal Trade Commission, 2015).

Security testing can be used to verify that issues are discovered and not left as vulnerabilities to be exploited. Testing techniques include static analysis, unit testing, and penetration testing. Static analysis can catch vulnerabilities in the syntax and dependencies used in an application. Unit testing can verify that classes and functions behave appropriately, especially after refactoring. Penetration testing can be completed by a white-hat hacker periodically to try and break the application. This will expose vulnerabilities to the latest techniques being used in the hacker community.

**Evaluation and assessment of risk and cost benefit of mitigation**

There are many different reasons why individuals or groups would want to hack into a system. The top reasons are for recognition, money, control, spying, and disruption (McAfee, 2011). Not all these motives may be applicable to your current system, but they should be evaluated. Reviewing the size of the system, data it holds, and population it targets, the specific motives may become clearer. I would apply this assessment to my own practice to better evaluate the security policies necessary to help maintain a secure system.

Since there are so many motives for attacking a system, a zero-trust concept should be considered. Errors can happen accidentally or through malicious attempts. Providing verification though every step can prevent successful attacks on a system. Preventing these issues can save a company millions of dollars in revenue and protect their reputation with their client base.

**Zero trust**

Zero trust is not trusting anyone even those already inside the network perimeter (Cloud Flare, 2024). Zero trust involves continuous monitoring, validation, least privilege, segmentation, etc. This concept builds upon the Defense in Depth security policy we created this term by providing more barriers to data.

This is useful to prevent data breaches, but you must be careful with how many barriers are in place for zero trust. A customer who must continuously validate their credentials during a single session could become frustrated.

To persuade developers to adopt zero trust, there must be a valid reason. Zero trust can require a lot of resources to implement and therefore is very costly. Terranova describes the implementation process as, “To set up Zero Trust architecture, organizations need to have a complete picture of their data and workflows. Every single resource and endpoint needs identifying, access control, and monitoring.” Looking at reasons such as data usage, business model, and future plans can promote developer adoption of zero trust and help achieve long term business goals.

**Implementation and recommendations of security policies**

Implementing a security policy is important to keep all staff up to date on the company’s standard. Enforcing security policies can help guide the developers in decisions throughout development. During the maintenance phase, developers can also reference the security policies to help handle new vulnerabilities or a breach (Federal Trade Commission, 2015).

Documentation should include the core principles, coding standards, development lifecycle, risk assessments, and other important security measures such as encryption or triple A. It is recommended that the policy stays up to date with the latest business practices to reflect the current business goals.

# References

Cloud Flare. (2024). *What is Zero Trust security?* Retrieved from Cloud Flare: https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/#:~:text=Zero%20Trust%20is%20a%20security,already%20inside%20the%20network%20perimeter.

Federal Trade Commission. (2015). *Start with Security: A Guide for Business*. Retrieved from Federal Trade Commission: https://www.ftc.gov/business-guidance/resources/start-security-guide-business

McAfee. (2011). *7 Types of Hacker Motivations*. Retrieved from McAfee: https://www.mcafee.com/blogs/consumer/family-safety/7-types-of-hacker-motivations/

Terra Nova. (2024). *The Limitations of Zero Trust Architecture and How to Overcome Them*. Retrieved from Terranova Security: https://terranovasecurity.com/blog/limitations-of-zero-trust-architecture/