Secure Code Reviews

What is a Secure Code Review?

We use static and dynamic analysis tools to analyze our source code and running applications respectively. Even though these tools are very efficient in identifying a wide range of bugs and vulnerabilities they are not without their limitations. They can be somewhat limited in finding vulnerabilities in our business logic. Since no tooling or strategy is 100% in identifying vulnerabilities in software, this is where a secure code review can help, or fill in the blanks. Just like a normal code review where source code is manually analyzed to ensure that required features are correct and in place, a secure code review ensures that there is no vulnerabilities in our business logic that can be potentially exploited. The goal of this type of review is to manually go through the code and identify any existing security flaws or vulnerabilities that may have crept it's way into the code and inadvertently slipped through the tooling.

Why do we need Secure Code Reviews?

A secure code review will supplement what a tool or set of tools may not have identified. No automated tool can possibly identify everything and many results of automated scans can end up as false positives. A secure code review will, in a way, fill in the loose ends and identify potential vulnerabilities in business logic that tooling may have a difficult time identifying. It will also confirm or verify reported false positives or false negatives. A secure code review can reveal flaws or potential vulnerabilities in software that may not otherwise be apparent such as implementation issues, design flaws, data validation errors, hard-coded credentials, configuration issues and many other problems.

What Problems Does a Secure Code Review Solve?

Secure code reviews help to identify weak security controls that might take a lot of testing time to find with a dynamic testing tool. For example, if an application is using a blacklist to prevent a SQL Injection, this can be identified rather quickly in a code review. On the other hand this same issue may take a longer time to identify via dynamic testing, depending on how good the blacklist is. Other examples may include:

- Input validation which can prevent a wide range of attacks such as injection, cross-site scripting and path manipulation.

- Parameterized SQL which can prevent injection attacks when input validation cannot be done.

- OS command injection can help prevent remote code execution..

- Data encryption in transit and at rest can help prevents data breaches.

- Neutralizing output prevents cross-site scripting flaws.

- Indirect object references prevent dangerous flaws associated with file management such as path traversal.

To Wrap Things Up

Secure code reviews are an important part of a secure software development lifecycle. When doing a code review we must focus on areas that are better suited for manually reviewing the code and having a human in the loop who can provide the context that tooling cannot. Code reviews can readily identify issues such as broken authentication, broken access control, database communication security, data encryption, input validation, and output sanitization.

Specifically, a secure code review should at the very least check that:

- ✓ User input is validated correctly
- ✓ Web applications use allowlist validation and not denylist validation
- ✓ User-supplied data is encoded before displaying it to the user
- ✓ Authentication and authorization mechanisms are in place where required
- ✓ There are no hardcoded credentials
- ✓ The application doesn’t use broken cryptographic algorithms for cryptography purposes
- ✓ The application uses prepared SQL statements to communicate with databases