Secure Coding and Not Leaving Security Until the End

Living in a time where cyber threats are constantly evolving, it is critical to approach all problems and projects with a security-focused approach no matter how big or small. Traditional security methods often approached security as an afterthought or a dedicated phase of the development process. This approach does not work in a world where security needs to be so heavily integrated with systems—leaving security to the end would result in complete overhauls of developed systems that will be expensive in terms of time and human resources. Instead, security should start from the very beginning during the initial planning of systems.

Secure coding is not only about writing programs that do not contain bugs or errors. Of course this is a part of it, but secure coding goes beyond that to take development to the next step and write code that actively supports a larger security infrastructure. This can be done in a standardized way through the implementation of a secure coding standard. Defining a secure coding standard may be a time-consuming commitment upfront, but it will be a worthwhile investment when systems are developed with consistent security standards regardless of the developers working on the system. This will also make secure coding easier for developers because they will not need to determine best practices themselves project to project as all rules for secure coding will be clearly defined for their reference.

Early integration of security practices through the implementation of a secure coding standard will help ensure that vulnerabilities are identified and fixed early in the development process. For example, code can be written securely to help prevent SQL injection attacks, buffer overflow events, memory allocation errors, mishandling of exceptions, and so on. These issues can be much more difficult to fix late in the development process than if best practices are written into the system from the beginning. Adoption of best coding standards like those outlined by OWASP or NIST can provide clear guidelines that reduce vulnerabilities, speed up development, simplify security audits, and create a security-minded culture.

Evaluation and Assessment of Risk and Cost Benefit of Mitigation

Effective security strategies should consider risks, costs, and benefits of mitigation. To do so, first risks should be assessed by identifying potential threats, their impact and their likelihood. Threats can be determined by identifying motives for attack, such as financial gain, personal vendettas, ideologies, revenge, and so on. They can also be identified by studying case studies for similar systems, and learning about emerging threats in recent and historic headlines. There are also common threats that are maybe not the most novel, but still need mitigation, such as injection attacks, denial of service attacks, and other similar attack types. Of course, in today’s world, it is close to impossible to predict every possible threat that could impact a system, but it is still good practice to consider likely threats.

Once threats are identified, the impact and probability should be determined to prioritize them appropriately. Risks that are associated with high loss or high probability (or both) should be prioritized more highly than low impact, low probability risks. Risks that could lead to severe financial loss, legal liabilities, reputational damage, ethical malpractice, or significant downtime could be considered high loss events. If any of these high-loss risks also have a high probability of occurring, they should be prioritized the highest for mitigation implementation. In some cases, the there can be solutions that can be implemented to resolve multiple issues. These types of solutions can also be prioritized more highly than single-issue solutions (depending on the nature of the associated risks). For example, implementation of multifactor authentication and role based access control can have a widespread impact that improves the security posture of the entire system, likely addressing multiple risk factors. Another consideration is the cost of implementing mitigative solutions (cost, meaning human resource requirements, financial requirements and time requirements for implementation). If two threats have similar risk profiles, it may be in the best interest of the system security to implement the solution that has a lower cost first. This will position the system to be safer during the implementation of the more expensive mitigation solution, which can take more time to develop.

While it may seem tempting to plan to implement every possible security solution on the market, it is important to take a more balanced approach for several reasons. First, security solutions can be very expensive in terms of upfront cost (for example initial server installations or licensing purchases) and ongoing costs (for example SaaS fees). As important as cybersecurity is, organizations typically have limited budget to dedicate to solutions. This will require careful selection and prioritization of security solutions to implement, as well as the ability to solidly justify the need for security solutions during budgeting cycles. Next, it will be important to ensure that there are enough human resources to support and maintain the security infrastructure put into place. Most systems are not just a “set it and forget it” type system, and they will not be beneficial to the organization if they are not properly maintained through updates, patches, and other maintenance protocol. For example, it was an out-of-date patch to Equifax’s Apache Struts Web Framework that enabled the 2017 cyber-attack leading to 143 million users’ data being stolen (Office of Public Affairs, 2020). If too many layers of protection are added beyond what is able to be maintained, it can actually be counterproductive to the organization’s security initiatives. Finally, throughout the implementation of more and more security solutions, there could be a point of diminished returns, even if there is more than enough budget and human resources to support them. This can lead to a system that is complex and confusing, and frankly the budget could probably be better spent elsewhere.

Zero Trust

The idea behind zero trust is that all activity should be treated as an active threat, never trusted until full authentication and authorization requirements have been fulfilled. This is a very different approach to more traditional perimeter-only security practices because it integrates security into every part of the system to ensure that even if an attacker makes it through the perimeter, other lines of defense will still be able to mitigate the attack. Zero trust requires continuous authentication, real-time monitoring, and struct access controls that use the principle of least privilege to protect the system.

Implementation of zero trust policies is not a simple installation of a single software or purchasing of a service. It requires a comprehensive understanding of the IT and OT networks for the entire organization. In implementing zero trust, it will be important to find a balance between strict protocols and functionality. For example, if a user is required to enter their credentials every 10 minutes, the security protocols may be in excess to the point where their job functions are being hindered, especially if the data and systems they are working in are not highly sensitive protected data. In some cases, it may be best practice to require entering of credentials this often or more often, it will simply depend on the context and security needs based on regulatory requirements and company security policies. This is simply an example of one of the many considerations that should be made when defining a zero-trust policy for an organization.

Implementation and Recommendations of Security Policies

When it comes to implementation of security policies, one of the first things to tackle is company culture. It will be important to develop a security-first minded culture to ensure that all employees have at least a foundational knowledge of security best practices. A Stanford University study found that 88% of breaches are caused by human error (Sjouwerman, n.d.). Addressing of culture is a difficult task, but it can involve regular security training activities for all employees. The content of these trainings can be tailored to their job function (developers may require much more rigorous technical training than what is required for less technical roles). Activities like simulated phishing attempts can help employees better identify common attack patterns. Clear, easily accessible policies for how to handle security best practices can also help employees understand how to respond to incidents quickly. These policies should be written in accessible language free from technical jargon that is difficult to understand. Regular audits of security practices can help incentivize participation in security initiatives as well.

Concurrent to addressing a cultural shift, integration of security into the development process should be prioritized as a transition from DevOps to DevSecOps processes. Each phase of development should have strengthened security practices through the implementation of secure coding standards, Triple A policies, encryption, and security automation. This can be done pretty seamlessly by integrating automation of security practices into each step of development, from including security analysis automation into the planning phase of a project through the automation of monitoring and response protocol post-deployment. Examples of automation for each phase of DevSecOps is shown in the following table from my Green Pace Security Policy Presentation (Eeg, 2025).

|  |  |  |
| --- | --- | --- |
| **Phase** | **Automation of:** | **Tools** |
| Assess and Plan | Evaluation of threat landscape and regulatory compliance requirements | Vanta, Drata, ThreatModeler |
| Design | Development of security roadmap | OWASP Software Assurance Maturity Model |
| Build | Checks for dependencies, secure build, and other misconfigured security settings/code | Snyk, Cppcheck |
| Verify and Test | Testing for vulnerabilities, such as fuzz testing or runtime application self-protection (RASP) | beSTORM, ThreatModeler |
| Transition and Health Check | Provisioning of security infrastructure | Terraform |
| Monitor and Detect | Logging and analytics in runtime, such as through security information and event management (SIEM) tools | Splunk by Cisco |
| Respond | Security Orchestration, Automation, and Response operations | Splunk Phantom, IBM Resilient |
| Maintain and Stabilize | System stabilization and returning to baseline security after events | Splunk by Cisco |

The final recommendation is the implementation of a zero trust policy. This will be supported by the prior recommendations (security automation, Triple A policy, and so on), but a dedicated definition and implementation of a zero trust framework will support a highly resilient system that can withstand existing and future novel threats.