THE DILIGENT DEVELOPER CHRONICLES

SECURITY JOURNEY

THE DILIGENT DEVELOPER CHRONICLES

I n the realm of Secure Development, great emphasis was placed on constructing applications with a strong foundation of security. From the earliest stages of their training, every aspiring developer learned the importance of diligence and careful consideration in their coding practices.

They were instilled with an awareness of the prevalent vulnerabilities that could potentially impact their code, and as they honed their skills, they gained the ability to create elegant code by studying and having hands-on training defending against these vulnerabilities. This process enabled them to build applications efficiently and securely.
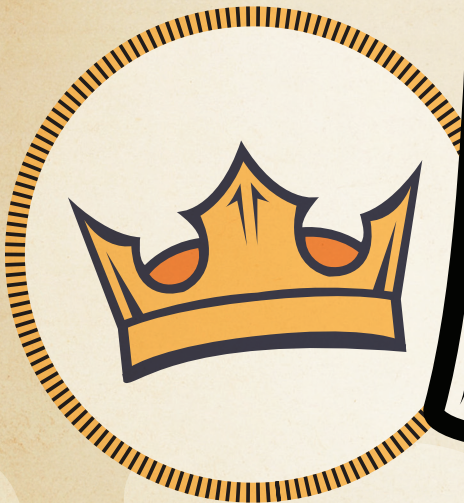
However, not all regions shared this enlightened approach. In many places, developers were taught to write code with minimal regard for the possibility of vulnerabilities, lacking any training in identifying or remedying these weaknesses. Consequently, a shadow fell upon these lands, threatening their security. Recognizing the impending danger, the wise king understood the need to intervene.

SECURITY JOURNEY

"Listen closely, for the kingdom is facing a grave peril. Our enemies lie beyond our gates, growing and evolving with each passing day. I call upon all Diligent Developers to embark on a noble quest to liberate our neighboring lands from the clutches of the most common vulnerabilities. It is our duty to impart knowledge and empower others, ensuring their continued freedom from such threats in the future. The task before us is daunting, but by fostering a resilient culture of security across our realm, we shall all reap the benefits."

- King Repoleved

www.SecurityJourney.com/DiligentDeveloper

SECURITY JOURNEY

V enturing beyond the borders of their kingdom, the Diligent Developers stumbled upon a dilapidated gate guarded by a group of determined security champions.

The champions recognized the dire consequences that could arise from such disrepair of broken access control, including unauthorized access to sensitive information, tampering with data, and even complete control over the application. They had wanted to make repairs, but alas had not been trained in how to do so.

Swiftly assessing the situation, the Diligent Developers set to work, applying the necessary security measures to fortify their neighbor's gate and helping the security champions ensure the overall safety of the application.

Security Controls to Repair the Gate of Broken Access Control ›

I. Role-Based Access Control:
Implementing a robust RBAC system ensures that users are granted access only to the resources and actions appropriate for their roles, restricting unauthorized access.

II. Principle of Least Privilege:
Grant users the minimum level of access necessary to perform their tasks, reducing the risk of unauthorized actions.

III. Proper Session Management:
Implement secure session management techniques, such as secure tokens, timeouts, and proper handling of session termination, to prevent unauthorized users from hijacking active sessions.

IV. Access Control Lists:
Moreover, they devised comprehensive Access Control Lists (ACLs) that explicitly defined permissions for both users and resources. By explicitly outlining what actions were permitted, they left no room for unauthorized actions to take place.

SECURITY JOURNEY

With the fire of their newfound knowledge, the security champions insisted on continuing the journey alongside the Diligent Developers. Moments later they encountered their second formidable adversary—the Wizard of Cryptographic Failures. Recognizing the gravity of this threat and the potential havoc it could wreak if left unvanquished, our heroes quickly gathered the team to make a gameplan for success.

Cryptographic Failures manifested when applications failed to adequately protect sensitive data through robust encryption or employed weak and outdated cryptographic algorithms. The repercussions of such failures were dire, encompassing unauthorized access to sensitive information, data breaches, and irreversible damage to reputation.

Undeterred by the challenge, the team confronted the Wizard of Cryptographic Failures in a tense battle of skill and strategy, applying the proper security controls, rendering the wizard's nefarious schemes powerless and neutralizing the looming threat.

Security Controls to Defeat the Wizard of Cryptographic Failure ›

I. Use Strong Encryption Algorithms: Employ the latest, industry-accepted encryption algorithms, such as AES or RSA, to ensure strong protection of sensitive data.

II. Implement Proper Key Management: Securely generate, store, and rotate encryption keys to prevent unauthorized access to sensitive data.

III. Regularly Update Cryptographic Libraries: Keep cryptographic libraries up-to-date to benefit from security improvements and avoid using outdated, insecure algorithms.

IV. Securely Transmit Data: Use secure communication protocols, such as HTTPS and TLS, to protect data during transmission over networks.

www.SecurityJourney.com/DiligentDeveloper

Slaying the Multi-headed Dragon of Injection Attacks

T he Diligent Developers and the security champions marched forward; their unity palpable. A formidable beast awaited their conquest—the multi-headed dragon of injection attacks. Each head of this fearsome creature represented a distinct type of injection attack.

SQL Injection: Exploiting vulnerabilities in database queries, allowing attackers to access, modify, or delete sensitive data.

Command Injection: Manipulating vulnerable applications to execute unauthorized system commands.

Cross-Site Scripting (XSS): Injecting malicious scripts into web applications, compromising user data and security.

LDAP Injection: Exploiting weaknesses in LDAP queries to access or modify sensitive directory information.

Equipped with their knowledge and skill, the team faced the Multi-Headed Dragon of Injection Attacks. Engaged in a fierce and dramatic battle, they adeptly applied the appropriate security control to each head of the dragon, systematically neutralizing the threats and ultimately achieving victory over the menacing creature.

Security Controls for Slaying the Dragon of Injection Attacks

I. Input Validation: Validate and sanitize user inputs to prevent malicious data from being injected into the application.

II. Parameterized Queries: Use parameterized queries or prepared statements to separate user data from SQL queries, mitigating SQL injection risks.

III. Data Sanitization: Implement data sanitization to remove any potentially harmful characters or scripts from user input, defending against XSS attacks.

IV. Secure LDAP Queries: Utilize secure LDAP queries and input validation to protect against LDAP injection attacks.

T he Diligent Developers could see their band of champions were exhilarated but tired from their fight. Noticing a castle in the distance, they lead the weary travelers to its gates. As they suspected, the principles of secure design had not been used to build the castle, and the team would need to fortify it before they could rest.

They carefully inspected the castle, identifying its vulnerabilities and weaknesses. For each vulnerability discovered, they applied the principles of Secure Design, reinforcing the castle and ensuring its stability.

Their numbers were growing as word of their conquests moved across the land. They rested for the evening before heading back out on their path.

Security Controls for Fortifying the Castle from Insecure Design ›

I. Least Privilege:
Ensure that users and components of the application have the minimum necessary access to perform their tasks.

II. Fail Securely:
Design the application to handle errors and failures securely, minimizing the impact of any security incidents.
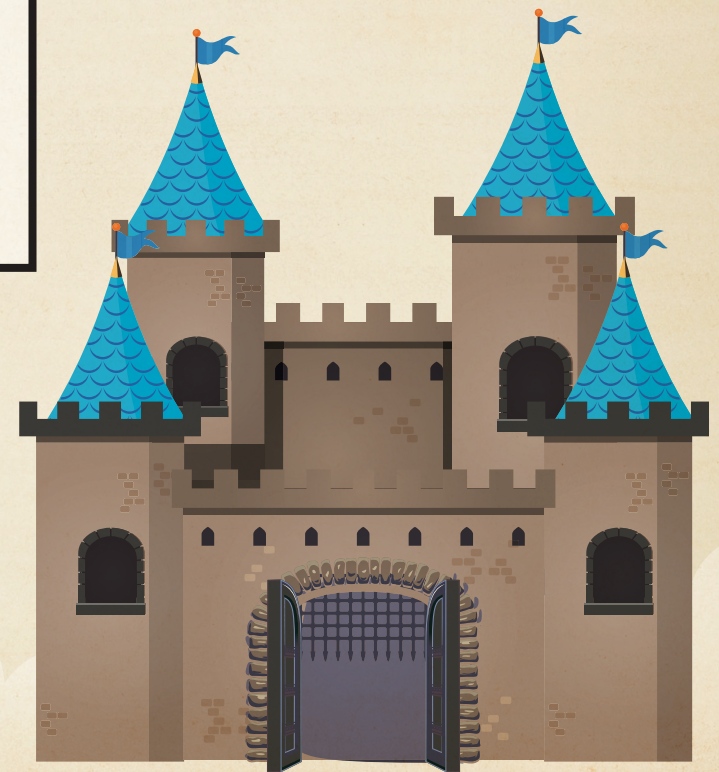
III. Separation of Concerns:
Divide the application into distinct components, each responsible for a specific function, to minimize complexity and improve maintainability.

IV. Defense in Depth:
Implement multiple layers of security to create a robust defense against threats.

THE DILIGENT DEVELOPER CHRONICLES

A s the Diligent Developer ventured forth, they stumbled upon a mysterious and perplexing structure - the Labyrinth of Security Misconfiguration. The labyrinth was a complex network of pathways, each leading to different components of the web application, all interconnected in a tangled mess. The labyrinth's design was the result of numerous misconfigurations, leaving the application vulnerable to a multitude of threats.

The diligent developers knew they had to unravel this intricate web to ensure the web application's security. They split up, each taking a team of eager security champions and embarked on the daunting task of disentangling the labyrinth and addressing each security misconfiguration they encountered.

The team cleared a path through the once chaotic structure and carefully moved along the path towards the forest ahead...

Security Controls to Clear the Labyrinth of Security Misconfiguration ›

I. **Establish a Repeatable Process to Manage and Harden Configurations:** Create a robust and repeatable process for managing configurations across all environments, ensuring that the application was consistently secure and hardened against attacks.

II. **Remove Unnecessary Functionality and Components:** Remove any unnecessary components, features, and dependencies that could introduce potential security risks, simplifying the application and reducing its attack surface.

III. **Regularly Patch and Update Software:** Update all software components to the latest versions, patching known vulnerabilities and minimizing potential risks.

IV. **Implement Proper Access Controls:** Review user roles and permissions, implementing the principle of least privilege to limit access to only what was necessary.

V. **Ensure Secure Storage of Sensitive Information:** Implemented strong encryption and secure storage mechanisms to protect sensitive information within the application.

SECURITY JOURNEY

Ridding the Forest of Vulnerable and Outdated Components

A s they moved into the dense, dark forest, a security champion told the Diligent Developers tales they had heard about the Forest of Vulnerable and Outdated Components. The security champions knew it threatened the stability and security of the web application, but never had the skills to rid the forest of its mess.

The Diligent Developers explained that vulnerable and outdated components are software libraries, frameworks, and other dependencies that have known security vulnerabilities or ones that are no longer maintained or supported. These components pose a significant risk to the application as attackers can exploit the known vulnerabilities, potentially leading to unauthorized access, data breaches, and other security incidents.

They knew they had to replace these components with secure and up-to-date alternatives to protect the application from potential attacks. Armed with their knowledge and skills, the team moved on the arduous task of identifying and replacing the outdated and vulnerable components in the forest:

Security Control to Rid the Forest of Vulnerable and Outdated Components ›

I. Use (SCA) to Identify Vulnerable Components: Employ a Software Composition Analysis (SCA) tool to automatically detect and report any vulnerable components in the application to address the issues swiftly.

II. Replace Outdated and Unsupported Components: Identify and replace components that are no longer supported or have known security vulnerabilities.

III. Maintain an Up-to-date Inventory of all Components Using a (SBOM): Create a comprehensive inventory of all software components, including their versions and dependencies, using a Software Bill of Materials (SBOM) to gain a clear understanding of the application's structure and the components that need updating.

IV. Establish a Process for Updating and Patching Components: Implement a robust and repeatable process for updating and patching components.

V. Regularly Monitor for Updates and Security Patches: Check for updates and security patches for all components, applying them promptly to minimize potential risks.

SECURITY JOURNEY

www.SecurityJourney.com/DiligentDeveloper

Taming the Whispering Shadow of Security Logging and Monitoring Failures

As they moved into the fog-shrouded Valley of Silence, the Diligent Developer caught glimpses of the Whispering Shadow, a spectral figure with the power to conceal malicious activities and evade detection. By exploiting Security Logging and Monitoring Failures, this elusive enemy could wreak havoc on the kingdom's web application, leaving no trace of its sinister deeds.

The Diligent Developers looked on proudly as their band of recruits were undeterred by the Whispering Shadow's elusive nature. They stepped in to battle and as they applied each security control, the Whispering Shadow's power to conceal its activities waned. Their efforts forced the shadow to retreat and relinquish its control over the Valley of Silence.

Security Controls to Tame to Whispering Shadow of Security Logging and Monitoring Failures >

I. Comprehensive Logging:
Set up comprehensive logging of security events, covering various aspects of the application, such as user authentication, access control, and data manipulation, to track and detect malicious activities.

II. Regular Log Review and Analysis:
Establish a process for regularly reviewing and analyzing logs, using automated tools and manual inspection to identify security incidents and potential threats.

III. Real-time monitoring and alerting:
Implement real-time monitoring and alerting systems to quickly detect and respond to security events, minimizing the impact of attacks and reducing the likelihood of successful breaches.

IV. Incident Response Planning:
Devise an incident response plan that outlines roles, responsibilities, and procedures for handling security events detected through logging and monitoring, ensuring a swift and effective response to potential threats.

V. Continuous Improvement:
Continuously refine and enhance logging and monitoring practices, learning from past incidents and adjusting strategies to address evolving threats.

www.SecurityJourney.com/DiligentDeveloper

Conquering the Twins of Identification and Authentication Failures

As the team moved out of the forest, they stumbled upon a castle shrouded in darkness, the lair of the Evil Twins, Identification and Authentication Failures.

These wicked twins had the power to disrupt the web application's security by exploiting weaknesses in the identification and authentication processes, allowing unauthorized access and potentially leading to data breaches, account takeovers, and other security incidents.

Emboldened by their journey so far, the Diligent Developers quickly used their expertise to protect the web application and conquer the Evil Twins and their plans.

Security Controls to Conquer the Twins of Identification and Authentication Failures ›
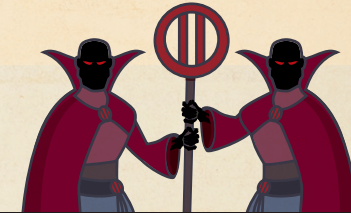
I. Implement Multi-factor Authentication (MFA):
Introducing MFA, requiring users to provide at least two forms of evidence to verify their identity, making it more difficult for attackers to impersonate legitimate users.

II. Use Strong, Unique Passwords:
Encourage users to create strong, unique passwords and implement password policies that require a minimum length, complexity, and frequent rotation, reducing the chances of passwords being cracked or guessed.

III. Employ Secure Password Storage:
Ensure that passwords are securely stored using proper hashing algorithms and salted hashes, making it more challenging for attackers to obtain and use stolen password data.

IV. Limit Login Attempts and Use Throttling:
To thwart brute force attacks, impose limits on the number of failed login attempts and implement throttling, progressively increasing the delay between login attempts after each unsuccessful try, slowing down attackers and reducing their chances of success.

V. Leverage Secure Session Management:
Implement secure session management practices, including secure session cookies, proper session timeouts, and secure handling of session identifiers, preventing unauthorized access to authenticated user sessions.

VI. Monitor and Log Authentication Events:
Set up monitoring and logging of authentication events, allowing the detection and investigation of suspicious activities, helping to identify potential breaches and other security incidents.

Vanquishing the Shape-Shifting Serpent of Software and Integrity Failures

The team found themselves emerging from the Valley of Silence to a place where the very fabric of reality seemed to shift and change. Amidst the chaos, they encountered a formidable foe: the Shapeshifting Serpent, the manifestation of Software and Data Integrity Failures. This treacherous beast had the power to corrupt software and alter data, jeopardizing the web application's reliability, performance, and security.

Realizing the dire threat posed by the Shapeshifting Serpent, the Diligent Developers confident in the team's abilities, stepped forward, leading the team to use their knowledge and skill to help maintain software and data integrity and vanquish the serpent, bringing clarity to the group that they would need as they made their way to their final test.

Security Controls to Vanquish the Shape-Shifting Serpent of Software and Integrity Failures ›

I. Establish a Strong Integrity Protection Mechanism: Implement a robust mechanism to protect the integrity of software and data, such as digital signatures, checksums, and cryptographic hashes.

II. Securely Manage and Protect Cryptographic Keys: Ensure that cryptographic keys are securely managed, stored, and protected from unauthorized access, preventing attackers from tampering with software and data.

III. Verify the Integrity of External Components: Check the integrity of third-party components and dependencies before incorporating them into the application, ensuring that they are free from corruption or tampering.

IV. Implement Secure Software Update Processes: Establish a secure process for updating the software and its dependencies, which includes validating the integrity of updates before installation.

V. Employ Secure Coding Practices: Use secure coding practices to minimize vulnerabilities that could lead to software and data integrity failures, such as input validation and sanitization, and secure data storage and transmission.

SECURITY JOURNEY

Securing the Wall of Server-Side Request Forgery

A s they made their way to the edge of the realm, a large wall on the horizon posed one the biggest problems of their journey - server-side request forgery. Evidence of malicious requests, servers being tricked into executing unauthorized actions abound and the Diligent Developers quickly got to work on assessing whether any bad actors had gained access to sensitive data or internal systems.

Relieved to find nothing further, working beside hundreds of security champions they began to secure the wall of server-side request forgery. Satisfied that they had fortified their last lines well, they gathered the team for one final lesson.

Security Controls to Tame to Secure the Wall of SSRF ›

I. **Limit Outbound Traffic:** Restrict outbound traffic from the application server, minimizing the potential impact of SSRF attacks and reducing the risk of unauthorized access to internal systems.

II. **Use a Strong Allowlist:** Employ a strong allowlist of trusted domains and IP addresses, preventing the server from executing requests to untrusted or malicious destinations.

III. **Sanitize User Input:** Sanitized user input, ensuring that unsafe or untrusted data could not be used to manipulate server requests.

IV. **Monitor and Log Network Activity:** Establish robust monitoring and logging of network activity, detecting and responding to any suspicious or malicious behavior indicative of SSRF attacks.

SECURITY JOURNEY

www.SecurityJourney.com/DiligentDeveloper

As the story reaches its conclusion, the Diligent Developers knew that the ever-evolving nature of application security meant that new threats would inevitably emerge. Vigilance and continuous efforts would be crucial to ensure the ongoing security of all of the realm's web applications.

However, they recognized the strength and knowledge of the team of security champions they had gathered along the way. The Diligent Developers proudly passed along their shields for the champions to continue their journey to impart their knowledge and build teams of Diligent Developers across the realm.

Their tale would be told for generations, reminding all of the importance of diligence, education, and the unwavering pursuit of application security.

The Diligent Developer Chronicles are part of a larger Diligent Developer security awareness and education program. We believe that developers need to be more than just aware of vulnerabilities, they need to have hands-on training with the opportunity to actually break and fix code to build the skills needed to build software securely. This program is designed for organizations to roll out broadly, but the content is open to any individuals wanting to further build their skills.

The Diligent Developer Security Awareness & Education Program Contains:

- The Diligent Developer Chronicles: OWASP Top 10 - A resource to get conversations going and raise awareness of the OWASP Top 10 across your organization and for teams to keep as a reference.

- Access to (3) OWASP Top 10 Video Lessons - podcast-style video lessons to engage both developers and non-developers to build a foundational understanding of the top application vulnerabilities today. Each lesson is less than 15 minutes long.
  - OWASP Top 10: Part 1
  - OWASP Top 10: Part 2
  - OWASP Top 10: Part 3

- Access to (3) Break/Fix Lessons - hands-on training on OWASP Top 10 vulnerabilities to build secure coding skills in your development team. Each lesson should take less than 15 minutes to complete.
  - Server-Side Request Forgery
  - Security Logging and Monitoring
  - SQL Injection

- Diligent Developer Assets - If you'd like to theme your program around the Diligent Developer, we have provided Zoom Backgrounds, Sticker Designs, Digital Badges, Emojis and artwork to support your efforts.

Because the content is based on the OWASP Top 10 published in late 2021, this content will only be available until December 31, 2023.

Learn more at www.SecurityJourney.com/DiligentDeveloper

SECURITY JOURNEY