UDACITY

CryptoV4ult Enterprise Security Review

Benjámin Rácskai

17.05.2024

Project Scenario

Section One: Integrating SDLC

Transitioning to Secure SDLC

Requirements Analysis
Conduct user interviews to gather functional requirements. Write a requirements document for task management features. Additional Task: Perform threat modeling to identify potential security risks and incorporate security requirements into the requirements document.
Design
Create a high-level architecture diagram for the application. Design the database schema for tasks. Additional Task: Conduct a security design review to ensure that the architecture and design decisions consider security principles and best practices.

Transitioning to Secure SDLC

Development
Implement CRUD operations for tasks. Implement interactive elements using JavaScript. Code the user interface using HTML and CSS. Set up a Flask application to handle API requests. Perform smoke testing on the deployed application. Additional Task: Implement input validation and output encoding to prevent common vulnerabilities such as injection attacks (e.g., SQL injection, cross-site scripting).
Testing
Write and execute functional test cases. Conduct browser compatibility testing. Additional Task: Perform security testing, including static code analysis, dynamic application security testing (DAST), and penetration testing, to identify and remediate security vulnerabilities.

Transitioning to Secure SDLC

Deployment
Deploy the application to Heroku. Additional Task: Configure secure deployment practices, such as using HTTPS, secure configuration management, and ensuring appropriate access controls are in place.
Maintenance
Monitor application logs and fix reported issues. Gather user feedback for future feature additions. Additional Task: Establish a process for timely security updates and patches, including vulnerability management and incident response procedures.

Advocating for Secure SDLC

1. **Continuous Risk Assessment and Mitigation**
With SDLC, security is integrated at every stage of development, allowing for continuous risk assessment and mitigation. This proactive approach ensures that potential vulnerabilities are identified early and addressed promptly, reducing the likelihood of security breaches that could compromise our cryptocurrency platform and erode user trust.
2. **Adaptability to Emerging Threats**
SDLC emphasizes adaptability and responsiveness to emerging cyber threats. By staying abreast of the latest security trends and incorporating security measures into the development process, CryptoV4ult can quickly adapt its defenses to counter new and evolving threats in the volatile cryptocurrency landscape.
3. **Faster Time to Market**
Unlike the rigid and sequential nature of Waterfall, SDLC promotes agility and iteration, enabling faster time to market for new features and updates. By incorporating security considerations from the outset and integrating security testing throughout the development lifecycle, CryptoV4ult can accelerate the release of secure, high-quality software without sacrificing speed or reliability.

4. Enhanced Regulatory Compliance

In the highly regulated cryptocurrency industry, adherence to security standards and regulatory requirements is paramount. SDLC provides a structured framework for incorporating security controls and compliance measures into the development process, helping CryptoV4ult demonstrate regulatory compliance and build trust with stakeholders, including users, investors, and regulatory authorities.

5. Improved Reputation and Customer Trust

Security breaches can have devastating consequences for CryptoV4ult's reputation and customer trust. By embracing SDLC and prioritizing security from the outset, CryptoV4ult can demonstrate its commitment to protecting user assets **and** sensitive information, thereby enhancing its reputation as a trustworthy and secure cryptocurrency platform. This increased trust can translate into greater user adoption, loyalty, and ultimately, long-term success in the competitive cryptocurrency market.

Section Two:

Vulnerabilities and Remediation

Vulnerabilities and remediation

1. Weak Password Policies
Description
Many login systems allow users to create weak passwords that are easy to guess or crack. Common issues include short passwords, lack of complexity, or not enforcing periodic password changes.
Risk
Weak passwords can be easily guessed or brute-forced by attackers, leading to unauthorized access to user accounts. This can result in data breaches, identity theft, and unauthorized transactions.
Remediation
Implement Strong Password Policies: Require passwords to be at least 12 characters long and include a mix of uppercase letters, lowercase letters, numbers, and special characters. *Enforce Password Expiration and History: Implement policies that require users to change their passwords regularly (e.g., every 90 days) and prevent them from reusing old passwords.* *Use Multi-Factor Authentication (MFA): Add an extra layer of security by requiring users to provide two or more verification factors, such as a password and a one-time code sent to their phone.*

Vulnerabilities and remediation

2. Inadequate Protection Against Brute Force Attacks
Description
Brute force attacks involve an attacker trying many different combinations of usernames and passwords until they find a match. Login systems that do not limit the number of failed login attempts are vulnerable to such attacks.
Risk
If attackers are able to perform unlimited login attempts, they have a higher chance of successfully guessing user credentials. This can lead to unauthorized access and potential data breaches.
Remediation
Implement Account Lockout Mechanisms: Temporarily lock accounts after a certain number of failed login attempts (e.g., five attempts) and notify users of suspicious activity. *Use CAPTCHA Systems: Add CAPTCHAs to the login process after a few failed login attempts to ensure the login attempts are being made by a human.* *Monitor and Alert: Monitor login attempts and alert users and administrators of potential brute force attacks.*

Vulnerabilities and remediation

3. Session Management Vulnerabilities
Description
Poor session management practices, such as using predictable session tokens, not expiring sessions after a period of inactivity, or not securely storing session tokens, can be exploited by attackers.
Risk
If session tokens are compromised, attackers can hijack active sessions and gain unauthorized access to user accounts without needing to know the user's credentials.
Remediation
Use Secure Session Tokens: Generate session tokens using a cryptographically secure random number generator and ensure tokens are unique and unpredictable. *Implement Session Expiry: Automatically expire sessions after a certain period of inactivity (e.g., 15 minutes) and force users to re-authenticate.* *Secure Token Storage: Store session tokens securely using encryption and ensure they are transmitted over secure channels (e.g., HTTPS).* *Regenerate Tokens on Privilege Change: Regenerate session tokens when a user logs in or changes privilege levels (e.g., from user to admin).*

Threat Matrix

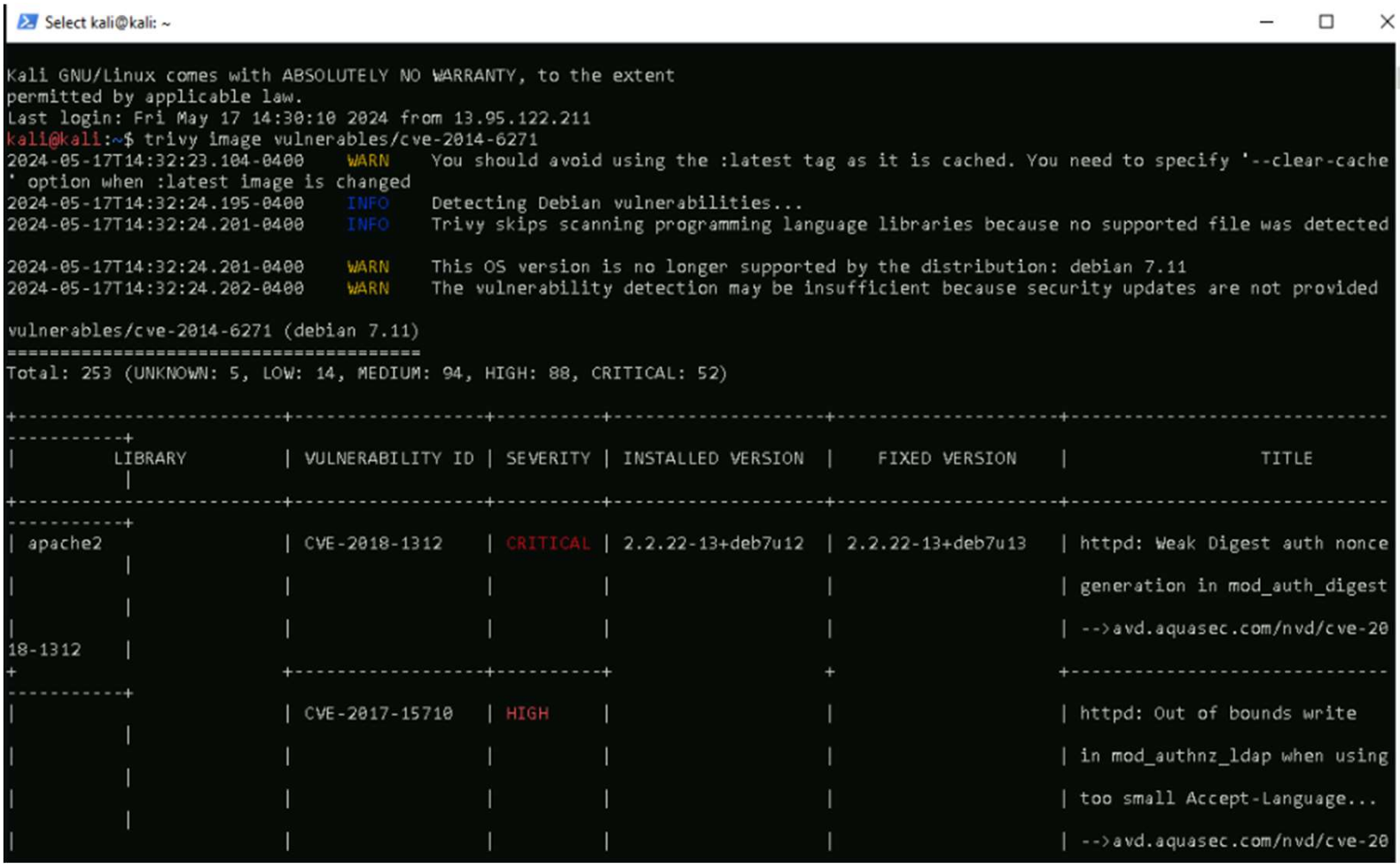
Pathway (Vulnerability)	Impact Level	Likelihood Level
Weak Password Policies	High	High
Brute Force Attack Vulnerability	Medium	High
Session Management Vulnerabilities	High	Medium

Fill out the matrix table. Impact levels are horizontal, and likelihood levels at the vertical axis.

Impact / Likelihood	Low	Medium	High
High		Brute Force Attack Vulnerability	Weak Password Policies
Medium			Session Management Vulnerabilities
Low			

Section Three: Container Security

Trivy scan screenshot



Report to Fix Container Issues

Issues	Unpatched Software Version	Patched Software Version
apache2: CVE-2018-1312	2.2.22-13+deb7u12	2.2.22-13+deb7u13
bash: CVE-2014-6271	4.2+dfsg-0.1	4.2+dfsg-0.1+deb7u1
libapr1: CVE-2017-12613	1.4.6-3+deb7u1	1.4.6-3+deb7u2
libaprutil1: CVE-2017-12618	1.4.1-3	1.4.1-3+deb7u1
tzdata: DLA-1155-1	2017b-0+deb7u1	2017c-0+deb7u1
procps: CVE-2018-1126	1:3.3.3-3	1:3.3.3-3+deb7u1
sensible-utils: CVE-2017-17512	0.0.7	0.0.7+deb7u1

Section Four: API Security

API Vulnerabilities and remediation

1. Insufficient Authentication and Authorization
Description
The API lacks proper authentication and authorization mechanisms, allowing unauthorized users or entities to access sensitive user data.
Risk
Unauthorized access to user data can lead to privacy breaches, data theft, identity theft, and regulatory compliance issues (e.g., GDPR, CCPA).
Remediation
Implement strong authentication mechanisms such as OAuth 2.0 with JWT (JSON Web Tokens) for user authentication. *Enforce role-based access control (RBAC) to restrict access to sensitive data based on user roles and permissions.* *Utilize HTTPS with TLS/SSL to encrypt data transmission between clients and the API server.*

API Vulnerabilities and remediation

2. Injection Attacks
Description
The API does not sanitize or validate input properly, allowing attackers to inject malicious code (e.g., SQL queries, NoSQL commands) into API requests.
Risk
Injection attacks can lead to unauthorized access to the database, data manipulation, data leakage, and in severe cases, complete compromise of the system.
Remediation
Use parameterized queries or prepared statements to prevent SQL injection vulnerabilities. *Employ ORM (Object-Relational Mapping) libraries to interact with the database, which automatically sanitizes inputs.* *Validate and sanitize all user inputs, including query parameters, request headers, and payloads, before processing them.*

API Vulnerabilities and remediation

3. Inadequate Data Protection
Description
The API does not adequately protect sensitive user data, such as storing it in plaintext or transmitting it over insecure channels.
Risk
Exposure of sensitive data can result in data breaches, identity theft, financial loss, reputational damage, and legal repercussions.
Remediation
Encrypt sensitive data at rest using strong cryptographic algorithms (e.g., AES) and industry-standard encryption protocols. *Use secure key management practices to protect encryption keys and ensure they are rotated regularly.* *Implement data masking techniques to anonymize or pseudonymize sensitive data whenever possible, reducing the impact of a breach.*