UDACITY

CryptoV4ult Enterprise Security Review

Simpson Innocent Nana
9th May, 2024

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 a threat modeling exercise to identify potential security risks associated with the task management features 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 evaluate the security implications of the chosen architecture and database schema, and document security controls and considerations in the design documentation.

Transitioning to Secure SDLC

Development

- Code the user interface using HTML and CSS.
- Implement interactive elements using JavaScript.
- Additional Task: Enforce secure coding practices, such as input validation, output encoding, and proper error handling, during the development phase to mitigate common security vulnerabilities like XSS and CSRF.

Testing

- Write and execute functional test cases.
- Conduct browser compatibility testing.
- Additional Task: Perform security-focused testing, such as penetration testing and vulnerability scanning, to identify and address security weaknesses in the application code and configurations.

Transitioning to Secure SDLC

Deployment

- Deploy the application to Heroku.
- Perform smoke testing on the deployed application.
- Additional Task: Implement continuous integration/continuous deployment (CI/CD) pipelines with automated security checks, such as static code analysis and dependency scanning, to ensure that only secure and tested code is deployed to production.

Maintenance

- Monitor application logs and fix reported issues.
- Gather user feedback for future feature additions.
- Additional Task: Establish a process for ongoing security monitoring and incident response, including regular security assessments, patch management, and proactive identification and mitigation of emerging threats.

Advocating for Secure SDLC

1. Earlier Vulnerability Identification
By embedding security into every stage of the software development process, Secure SDLC allows us to identify vulnerabilities much earlier in the development lifecycle. This proactive approach enables us to address security issues before they escalate into more significant problems, reducing the risk of potential security breaches and ensuring the integrity of our cryptocurrency platform.
2. Reduced costs
Transitioning to Secure SDLC can lead to significant cost savings for CryptoV4ult. By identifying and addressing security vulnerabilities early in the development process, we can avoid costly remediation efforts and potential fines associated with security incidents. Additionally, preventing security breaches through proactive security measures can help mitigate the financial impact of downtime, data breaches, and regulatory non-compliance.
3. Lower Business Risks
Secure SDLC helps mitigate business risks by minimizing the likelihood and impact of security breaches on our cryptocurrency platform. By prioritizing security throughout the development lifecycle, we can build trust with our users and stakeholders, protect sensitive data and assets, and safeguard the reputation and credibility of CryptoV4ult in the competitive cryptocurrency market.

Advocating for Secure SDLC

4. Enhanced Regulatory Compliance

Implementing Secure SDLC practices ensures that CryptoV4ult meets regulatory requirements and industry standards for data protection and security. By incorporating security into every stage of the development process, we can demonstrate due diligence and compliance with legal and regulatory mandates, reducing the risk of regulatory penalties and legal liabilities.

5. Improved Customer Confidence

Transitioning to Secure SDLC not only enhances the security of our cryptocurrency platform but also instills confidence and trust in our users. By prioritizing security from the outset, we demonstrate our commitment to protecting user data and assets, fostering long-term customer relationships, and maintaining CryptoV4ult's reputation as a reliable and trustworthy platform for cryptocurrency transactions.

Section Two:

Vulnerabilities and Remediation

Vulnerabilities and remediation

1. Weak Password Policies
Description
Weak password policies refer to lax requirements or inadequate enforcement of password complexity, length, and expiration. This vulnerability allows attackers to easily guess or brute-force user passwords, compromising account security.
Risk
Attackers can exploit weak password policies to gain unauthorized access to user accounts, potentially leading to account takeover, data theft, and unauthorized transactions. This vulnerability poses a high risk to CryptoV4ult's operational functionality, customer trust, and financial stability.
Remediation
Implement and enforce strong password policies that require users to create complex passwords with a minimum length, combination of uppercase and lowercase letters, numbers, and special characters. Additionally, enforce regular password changes and implement multi-factor authentication (MFA) to enhance account security.

Vulnerabilities and remediation

2. Session Fixation
Description
Session fixation occurs when an attacker can manipulate or predict a user's session identifier, allowing them to hijack the user's session and gain unauthorized access to the application.
Risk
Attackers can exploit session fixation vulnerabilities to impersonate legitimate users, perform unauthorized actions, and access sensitive information within the application. This vulnerability poses a medium risk to CryptoV4ult's operational functionality and customer trust.
Remediation
Implement secure session management practices, such as generating unique session identifiers for each session, using secure cookies with the 'HttpOnly' and 'Secure' flags, and regenerating session identifiers after successful authentication or privilege changes.

Vulnerabilities and remediation

3. Lack of Account Lockout Mechanism
Description
Lack of an account lockout mechanism allows attackers to perform brute-force attacks by repeatedly attempting to guess user credentials without any restrictions or consequences.
Risk
Attackers can exploit this vulnerability to launch brute-force attacks against user accounts, leading to unauthorized access, account takeover, and potential data breaches. This vulnerability poses a high risk to CryptoV4ult's operational functionality, customer trust, and financial stability.
Remediation
Implement an account lockout mechanism that temporarily locks user accounts after a certain number of failed login attempts. Additionally, implement CAPTCHA challenges or progressive delays to thwart automated brute-force attacks and notify users of suspicious login attempts.

Threat Matrix

Pathway (Vulnerability)	Impact Level	Likelihood Level
Weak Password Policies	High	High
Session Fixation	Medium	Medium
Lack of Account Lockout	High	High

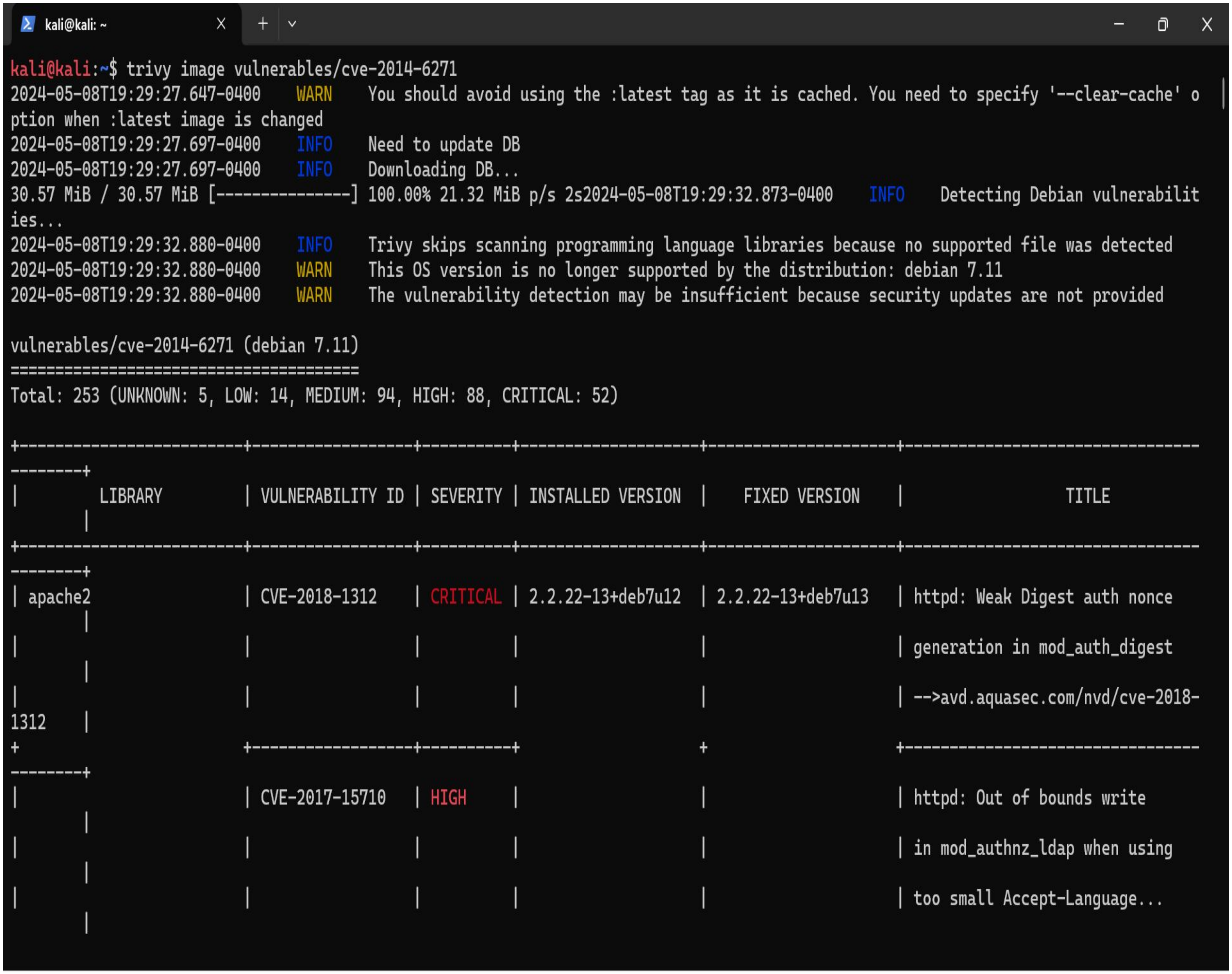
Fill out the matrix table. Impact levels are horizontal, and likelihood levels at the vertical axis.

Impact Likelihood	Low	Medium	High
High	Medium	Medium High	High (Lack of Account Lockout, Weak Password Policies)
Medium	Low Medium	Medium (Session Fixation)	Medium High
Low	Low	Low Medium	Medium

Section Three: Container Security

Trivy scan screenshot

Place a screenshot from the Trivy scan results on this slide.



```
kali@kali:~$ trivy image vulnerables/cve-2014-6271
2024-05-08T19:29:27.647-0400    WARN    You should avoid using the :latest tag as it is cached. You need to specify '--clear-cache' o
ption when :latest image is changed
2024-05-08T19:29:27.697-0400    INFO    Need to update DB
2024-05-08T19:29:27.697-0400    INFO    Downloading DB...
30.57 MiB / 30.57 MiB [---------------] 100.00% 21.32 MiB p/s 2s2024-05-08T19:29:32.873-0400    INFO    Detecting Debian vulnerabilit
ies...
2024-05-08T19:29:32.880-0400    INFO    Trivy skips scanning programming language libraries because no supported file was detected
2024-05-08T19:29:32.880-0400    WARN    This OS version is no longer supported by the distribution: debian 7.11
2024-05-08T19:29:32.880-0400    WARN    The vulnerability detection may be insufficient because security updates are not provided

vulnerables/cve-2014-6271 (debian 7.11)
=======================================
Total: 253 (UNKNOWN: 5, LOW: 14, MEDIUM: 94, HIGH: 88, CRITICAL: 52)

+-------------------------+------------------+----------+--------------------+---------------------+---------------------------------
--------+
|         LIBRARY         | VULNERABILITY ID | SEVERITY | INSTALLED VERSION  |    FIXED VERSION    |                  TITLE
        |
+-------------------------+------------------+----------+--------------------+---------------------+---------------------------------
--------+
| apache2                 | CVE-2018-1312    | CRITICAL | 2.2.22-13+deb7u12  | 2.2.22-13+deb7u13   | httpd: Weak Digest auth nonce
        |
|                         |                  |          |                    |                     | generation in mod_auth_digest
        |
|                         |                  |          |                    |                     | -->avd.aquasec.com/nvd/cve-2018-
1312    |
+                         +------------------+----------+                    +                     +---------------------------------
--------+
|                         | CVE-2017-15710   | HIGH     |                    |                     | httpd: Out of bounds write
        |
|                         |                  |          |                    |                     | in mod_authnz_ldap when using
        |
|                         |                  |          |                    |                     | too small Accept-Language...
        |
```

Report to Fix Container Issues

Fill out the report with at least 7 items.

Issues	Unpatched Software Version	Patched Software Version
apache2: CVE-2018-1312	2.2.22-13+deb7u12	2.2.22-13+deb7u13
libssl1.0.0: CVE-2017-3735	1.0.1t-1+deb7u2	1.0.1t-1+deb7u3
bash: CVE-2014-6271	4.2+dfsg-0.1	4.2+dfsg-0.1+deb7u1
bash: CVE-2014-6277	4.2+dfsg-0.1	4.2+dfsg-0.1+deb7u3
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
libprocps0: CVE-2018-1126	1:3.3.3-3	1:3.3.3-3+deb7u1

Section Four: API Security

API Vulnerabilities and remediation

1. Broken User Authentication
Description
Broken user authentication refers to vulnerabilities in the authentication mechanisms of the API, such as weak password policies, insufficient credential protection, or improper session management. Attackers could exploit these weaknesses to gain unauthorized access to sensitive user data or perform unauthorized actions on behalf of legitimate users.
Risk
Attackers could compromise user accounts, extract sensitive information, or manipulate user data, leading to identity theft, fraud, or unauthorized access to confidential data. This vulnerability poses a high risk to the confidentiality, integrity, and availability of user data, potentially damaging customer trust and tarnishing the reputation of both CryptoV4ult and the external sales vendor.
Remediation
Implement strong authentication mechanisms, such as multi-factor authentication (MFA) or biometric authentication, to verify the identity of users securely. Encrypt sensitive data during transmission and storage, use secure session management techniques, such as session tokens with short lifetimes, and regularly audit and monitor authentication logs for suspicious activity to detect and mitigate unauthorized access attempts promptly.

API Vulnerabilities and remediation

2. Excessive Data Exposure
Description
Excessive data exposure occurs when APIs inadvertently expose more data than necessary, such as personally identifiable information (PII), financial records, or other sensitive data, without proper access controls or data masking techniques. This vulnerability could result from improper data handling, insufficient access controls, or inadequate data anonymization practices.
Risk
Attackers could exploit excessive data exposure to access and exfiltrate sensitive user data, such as usernames, passwords, email addresses, or financial information, leading to identity theft, fraud, or unauthorized access to confidential information. This vulnerability poses a high risk to user privacy, regulatory compliance, and the reputation of CryptoV4ult and the external sales vendor.
Remediation
Implement strict access controls and data minimization practices to limit access to sensitive data based on user roles and permissions. Use encryption, tokenization, or data masking techniques to protect sensitive data both in transit and at rest. Conduct regular data privacy impact assessments and vulnerability scans to identify and mitigate potential data exposure risks proactively.

API Vulnerabilities and remediation

3. Lack of Rate Limiting
Description
Lack of rate limiting refers to the absence of restrictions on the number of requests or transactions that a user or IP address can make within a given time period. Without rate limiting controls, attackers could overwhelm the API with a high volume of requests, leading to service degradation or denial of service (DoS) attacks.
Risk
Attackers could launch brute-force attacks, API abuse, or automated bots to flood the API with a large number of requests, causing service disruptions, performance degradation, or complete downtime. This vulnerability poses a high risk to the availability and reliability of CryptoV4ult's API services, impacting user experience and potentially leading to financial losses or reputational damage.
Remediation
Implement rate limiting controls, such as request throttling, API usage quotas, or IP address-based rate limiting, to restrict the number of requests or transactions that a user or IP address can make within a specified time frame. Monitor API usage metrics and set appropriate thresholds for rate limiting rules based on expected usage patterns and traffic volumes. Implement anomaly detection mechanisms to identify and mitigate unusual or suspicious API activity indicative of potential DoS attacks.