UDACITY

CryptoV4ult Enterprise Security Review

Uriyah Adriel Sam

20-05-2024

Digital Project Management

Project Scenario

Digital Project Management

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. **Identify security requirements and threats**.
Design
Create a high-level architecture diagram for the application. Design the database schema for tasks. **Perform threat modeling and design security controls**.

Transitioning to Secure SDLC

Development
Code the user interface using HTML and CSS. Implement interactive elements using JavaScript. Set up a Flask application to handle API requests. Implement CRUD operations for tasks. **Conduct secure coding training for developers.**
Testing
Write and execute functional test cases. Conduct browser compatibility testing. **Perform security testing (e.g., static code analysis, dynamic analysis).**

Transitioning to Secure SDLC

Deployment

Deploy the application to Heroku.
Perform smoke testing on the deployed application.
Conduct a security review and penetration testing before deployment.

Maintenance

Monitor application logs and fix reported issues.
Gather user feedback for future feature additions.
Regularly apply security patches and updates.

Advocating for Secure SDLC

1. Enhanced Security Throughout Development
Incorporating security tasks at each stage of development ensures that vulnerabilities are identified and mitigated early, reducing the risk of security breaches which is crucial for the integrity of a cryptocurrency platform.
2. Continuous Integration and Feedback
Secure SDLC promotes iterative development and continuous integration, allowing for regular feedback and timely updates. This agility helps quickly address and patch security vulnerabilities as they are discovered.
3. Improved Compliance and Risk Management
By embedding security practices into the SDLC, we can ensure compliance with regulatory standards and industry best practices. This proactive approach helps in managing and mitigating risks associated with handling sensitive financial data.
4. Higher Quality and More Reliable Code
Regular security testing and code reviews lead to higher quality and more reliable code. This reduces the likelihood of bugs and exploits, enhancing the overall robustness of our cryptocurrency platform.
5. [Benefit]
[1-2 sentence description]

Advocating for Secure SDLC

5.Cost Efficiency and Resource Optimization
Identifying and addressing security issues early in the development cycle is more cost-effective than fixing them post-deployment. This optimization saves resources and reduces the financial impact of potential security incidents.

Digital Project Management Section Two:

Vulnerabilities and Remediation

Vulnerabilities and remediation

1. Weak Password Policies
Description
Weak password policies often allow users to create passwords that are easily guessable, such as "123456" or "password." This vulnerability occurs when the system does not enforce strong password requirements, such as a minimum length, complexity, or regular password changes.
Risk
Attackers can exploit weak passwords using brute force or dictionary attacks, leading to unauthorized access. Once inside, they can steal sensitive information, manipulate transactions, or disrupt services, causing significant harm to users and the platform.
Remediation
Implement strong password policies that require a mix of uppercase and lowercase letters, numbers, and special characters. Enforce a minimum password length (e.g., at least 12 characters) and require periodic password changes. Additionally, consider using multi-factor authentication (MFA) to add an extra layer of security.

Vulnerabilities and remediation

2. SQL Injection
Description
SQL injection occurs when user input is not properly sanitized and is executed as part of an SQL query. This allows attackers to execute arbitrary SQL code, which can manipulate or access the database without authorization.
Risk
Attackers can use SQL injection to bypass authentication mechanisms, retrieve or alter sensitive data, delete records, and perform administrative operations on the database. This can lead to data breaches, loss of data integrity, and full system compromise.
Remediation
Use prepared statements and parameterized queries to ensure user inputs are treated as data rather than executable code. Sanitize and validate all user inputs, and employ ORM (Object-Relational Mapping) frameworks that abstract database interactions to reduce direct SQL query usage. Regularly review and test the code for injection vulnerabilities.

Vulnerabilities and remediation

3. Session Hijacking
Description
Session hijacking occurs when an attacker gains unauthorized access to a user's session token, allowing them to impersonate the user without needing their login credentials. This can happen through methods such as man-in-the-middle attacks, cross-site scripting (XSS), or session token leakage.
Risk
With access to a session token, attackers can perform actions on behalf of the user, access sensitive information, and potentially escalate their privileges within the system. This compromises the security and privacy of user accounts and the integrity of the platform.
Remediation
Use secure, randomly generated session tokens with sufficient entropy. Implement HTTPS to encrypt all data transmitted between the user and the server, protecting session tokens from being intercepted. Additionally, set appropriate session timeout periods and invalidate sessions after logout or after a period of inactivity. Use techniques such as HttpOnly and Secure flags for cookies to mitigate the risk of token theft via XSS.

Threat Matrix

Pathway (Vulnerability)	**Impact Level**	**Likelihood Level**
Weak Password	High	High
SQL Injection	High	Medium
Session Hijacking	High	Medium

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

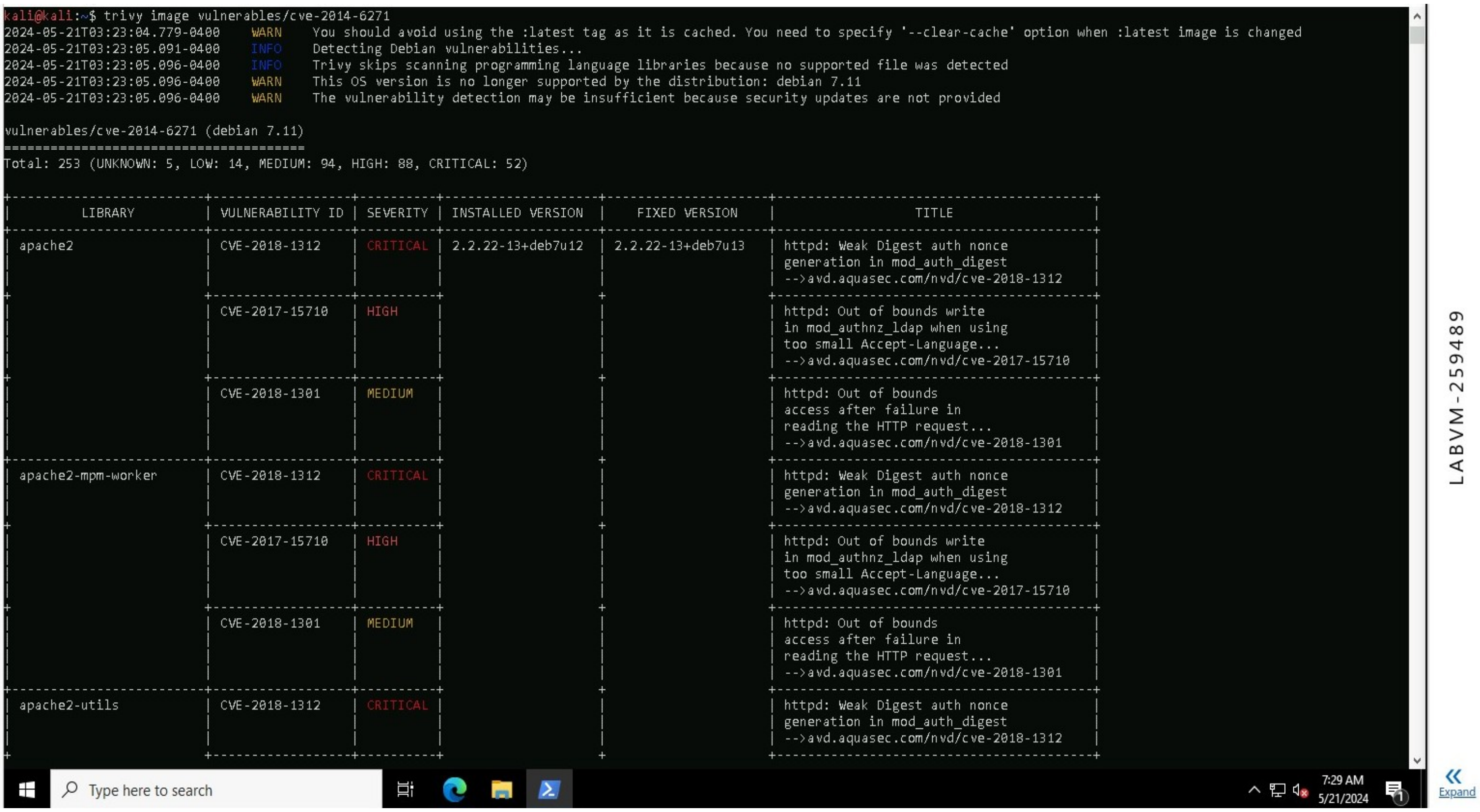
Impact Likelihood	Low	Medium	High
High			- Weak Password Policies, - SQL Injection
Medium			Session Hijacking
Low			

Digital Project Management

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
2024-05-21T03:23:04.779-0400    WARN    You should avoid using the :latest tag as it is cached. You need to specify '--clear-cache' option when :latest image is changed
2024-05-21T03:23:05.091-0400    INFO    Detecting Debian vulnerabilities...
2024-05-21T03:23:05.096-0400    INFO    Trivy skips scanning programming language libraries because no supported file was detected
2024-05-21T03:23:05.096-0400    WARN    This OS version is no longer supported by the distribution: debian 7.11
2024-05-21T03:23:05.096-0400    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...            |
|                          |                  |          |                   |                     | -->avd.aquasec.com/nvd/cve-2017-15710   |
+                          +------------------+----------+                   +                     +-----------------------------------------+
|                          | CVE-2018-1301    | MEDIUM   |                   |                     | httpd: Out of bounds                    |
|                          |                  |          |                   |                     | access after failure in                 |
|                          |                  |          |                   |                     | reading the HTTP request...             |
|                          |                  |          |                   |                     | -->avd.aquasec.com/nvd/cve-2018-1301    |
+--------------------------+------------------+----------+                   +                     +-----------------------------------------+
| apache2-mpm-worker       | CVE-2018-1312    | CRITICAL |                   |                     | 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...            |
|                          |                  |          |                   |                     | -->avd.aquasec.com/nvd/cve-2017-15710   |
+                          +------------------+----------+                   +                     +-----------------------------------------+
|                          | CVE-2018-1301    | MEDIUM   |                   |                     | httpd: Out of bounds                    |
|                          |                  |          |                   |                     | access after failure in                 |
|                          |                  |          |                   |                     | reading the HTTP request...             |
|                          |                  |          |                   |                     | -->avd.aquasec.com/nvd/cve-2018-1301    |
+--------------------------+------------------+----------+                   +                     +-----------------------------------------+
| apache2-utils            | CVE-2018-1312    | CRITICAL |                   |                     | httpd: Weak Digest auth nonce           |
|                          |                  |          |                   |                     | generation in mod_auth_digest           |
|                          |                  |          |                   |                     | -->avd.aquasec.com/nvd/cve-2018-1312    |
+                          +------------------+----------+                   +                     +-----------------------------------------+
```

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
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
Libprocps0: CVE-2018-1126	1:3.3.3-3	1:3.3.3-3+deb7u1
Libssl1.0.0: CVE-2017-3735	1.0.1t-1+deb7u2	1.0.1t-1+deb7u3
Openssl: CVE-2017-3735	1.0.1t-1+deb7u2	1.0.1t-1+deb7u3
Procps: CVE-2018-1126	1:3.3.3-3	1:3.3.3-3+deb7u1

Digital Project Management

Section Four: API Security

API Vulnerabilities and remediation

1. Broken Object Level Authorization (BOLA)
Description
Broken Object Level Authorization occurs when an API endpoint does not properly enforce access controls, allowing users to access objects they should not be able to. This vulnerability often arises when unique identifiers (e.g., user IDs) are used in API requests without sufficient validation of user permissions.
Risk
Attackers can exploit BOLA to access or manipulate data belonging to other users, leading to unauthorized data exposure, data modification, and potential data breaches. This can severely compromise user privacy and platform integrity, especially when sensitive information is involved.
Remediation
Implement strict access control checks at the API level for every endpoint. Ensure that the server verifies the requester’s permissions for the requested object before performing any operations. Use role-based access control (RBAC) and attribute-based access control (ABAC) mechanisms to enforce fine-grained permissions.

API Vulnerabilities and remediation

2. Lack of Rate Limiting
Description
Lack of rate limiting occurs when an API does not restrict the number of requests a user or client can make in a given time period. This can lead to abuse, such as brute force attacks, denial of service attacks, and excessive data extraction.
Risk
Attackers can exploit this vulnerability to overwhelm the system, leading to degraded performance or service outages. Additionally, it can facilitate brute force attacks on authentication endpoints or mass extraction of user data, resulting in security breaches and data loss.
Remediation
Implement rate limiting for all API endpoints to control the number of requests that can be made from a single IP address or user account within a specific time frame. Use API gateways or middleware to enforce rate limiting policies. Additionally, monitor and log API usage to detect and respond to suspicious activity.

API Vulnerabilities and remediation

3. Insufficient Data Validation and Sanitization
Description
Insufficient data validation and sanitization occurs when the API fails to properly validate and sanitize user inputs, allowing malicious data to be processed by the system. This can lead to various types of injection attacks, such as SQL injection, command injection, and cross-site scripting (XSS).
Risk
Attackers can inject malicious code or commands through unsanitized inputs, leading to unauthorized data access, data corruption, or full system compromise. This poses significant risks to user data confidentiality and platform security.
Remediation
Implement comprehensive input validation and sanitization on all API endpoints. Ensure that all inputs are checked for type, length, format, and allowed characters. Use libraries and frameworks that provide built-in protection against injection attacks. Additionally, apply output encoding where appropriate to prevent XSS.