UDACITY

CryptoV4ult Enterprise Security Review

JUBRIL EDUN

[5-10-2024]

Project Scenario

Overview

As the lead security engineer for CryptoV4ult, a prominent international cryptocurrency platform, you're tasked with ensuring the security and integrity of our newly established infrastructure. With over 1 million users relying on our services, it's imperative that we maintain the highest standards of security to protect their digital assets.

Your role involves a comprehensive review of the security landscape for our new application technology stack, identifying potential vulnerabilities, and running scans to assess any existing threats. Your scope encompasses various entities within our architecture, including the application itself, containerized services, and the external-facing API.

Ultimately, your objective is to develop a robust remediation plan that not only addresses current vulnerabilities but also strengthens our overall security posture, safeguarding both user data and the platform's reputation. This critical mission presents an exciting opportunity to leverage your skills and expertise in cybersecurity to fortify our infrastructure and uphold our commitment to providing a secure and reliable platform for our users. Let's embark on this journey together to ensure CryptoV4ult remains a trusted leader in the cryptocurrency industry!

Section One: Integrating SDLC

Transitioning to Secure SDLC

As the lead security engineer at CryptoV4ult, you are tasked with ensuring the new infrastructure is developed securely. Your responsibility is to reorganize the existing development tasks to fit into a Secure Software Development Lifecycle (SDLC) framework, ensuring that each stage of the lifecycle incorporates necessary security tasks to protect user data and maintain the integrity of the cryptocurrency platform.

- ***Reorganize the Waterfall*** *task list from the next slide* ***into the Secure SDLC phases***
- ***Add at least one security related additional task to each phase***

Transitioning to Secure SDLC

Place every task into a Secure SDLC category in the next few slides. Add at least one additional task to each phase that helps enhance security.

1. Conduct user interviews to gather functional requirements.
2. Write a requirements document for task management features.
3. Create a high-level architecture diagram for the application.
4. Design the database schema for tasks.
5. Code the user interface using HTML and CSS.
6. Implement interactive elements using JavaScript.
7. Set up a Flask application to handle API requests.
8. Implement CRUD operations for tasks.
9. Write and execute functional test cases.
10. Conduct browser compatibility testing.
11. Deploy the application to Heroku.
12. Perform smoke testing on the deployed application.
13. Monitor application logs and fix reported issues.
14. Gather user feedback for future feature additions.

Transitioning to Secure SDLC

Requirements Analysis

Conduct user interviews to gather functional requirements.
Conduct feasibility studies to assess the project's viability.
Perform threat modeling to identify potential security risks and requirements.
Gather user requirements through interviews, surveys, and workshops.

Design

Write a requirements document for task management features.
Create a high-level architecture diagram for the application.
Design the database schema for tasks
Prepare detailed technical specifications..
Include security requirements and considerations in the requirements document, such as authentication, authorization, and data encryption

Transitioning to Secure SDLC

Development

Code the user interface using HTML and CSS.
Implement interactive elements using JavaScript.
Implement interactive elements using JavaScript.
Implement CRUD operations for tasks.
Implement secure coding practices to prevent common vulnerabilities
Create unit tests for individual components.
Implement proper authentication and authorization mechanisms for API endpoints.

Testing

Write and execute functional test cases

Conduct browser compatibility testing

Perform performance testing, security testing, and usability testing

Execute regression tests to ensure that new changes do not affect existing functionality.

Conduct security testing, including penetration testing and vulnerability scanning, to identify and address any security weaknesses

Transitioning to Secure SDLC

Deployment

Deploy the application to Heroku
Configure servers, databases, and other infrastructure components
Perform smoke testing on the deployed application
Enable HTTPS to encrypt data in transit and encryption to encrypt data at rest

Maintenance

Monitor application logs and fix reported issues
Gather user feedback for future feature additions
Apply patches and updates to fix bugs and security vulnerabilities
Plan for future upgrades and enhancements
Implement logging of security-related events and monitor for suspicious activities.

Advocating for Secure SDLC

As the lead security engineer at CryptoV4ult, you're spearheading the shift towards a more secure and agile development process. To get everyone on board, create a succinct list highlighting five essential advantages of transitioning to the Secure Software Development Lifecycle (SDLC) from our current Waterfall methodology. **For each advantage, include a brief explanation** that underscores its importance, particularly focusing on how it benefits the dynamic and security-centric nature of our cryptocurrency platform.

- *Write your answers on the next slide!*

Advocating for Secure SDLC

1. Improved Security Posture
SDLC emphasizes security at every stage of development, from design to deployment and maintenance. By integrating security practices throughout the development process, we can proactively identify and mitigate potential vulnerabilities, ensuring the integrity and resilience of our platform against evolving cyber threats.
2. Faster Response to Security Threats:
SDLC enables agile development cycles, allowing us to adapt quickly to emerging security threats and implement necessary countermeasures without disrupting ongoing operations. With continuous integration and automated testing, we can rapidly detect and remediate vulnerabilities, reducing the window of exposure to potential attacks and maintaining the trust of our users in our platform's security capabilities
3. Early Detection of Security Flaws
SDLC encourages thorough security assessments, including code reviews, static analysis, and vulnerability scanning, during the development process. By identifying and addressing security flaws early in the lifecycle, we can minimize the cost and effort required to remediate issues post-release, reducing the risk of security incidents and potential financial losses for CryptoV4ult and its users

Advocating for Secure SDLC

4. Enhanced Compliance and Regulatory Alignment
SDLC frameworks, such as those based on industry standards like OWASP and NIST, provide guidelines for achieving compliance with regulatory requirements and industry best practices. By adhering to these frameworks, we can demonstrate our commitment to security and compliance, instilling confidence in regulators, auditors, and stakeholders
5. Enhanced Trust and Customer Confidence
SDLC demonstrates a commitment to security and reliability, enhancing trust and confidence among CryptoV4ult's customers and stakeholders. By transparently communicating our security measures and adherence to industry best practices, we can differentiate ourselves in the competitive cryptocurrency market and attract and retain users who prioritize security and trustworthiness in their financial transactions.

Section Two:

Vulnerabilities and Remediation

Vulnerabilities and remediation

As CryptoV4ult enhances its infrastructure to support new features for its extensive user base, ensuring the security of user authentication mechanisms is paramount. The **login system** is critical to the platform's security, acting as the first line of defense against unauthorized access. Your task is to scrutinize a login system, **identify 3 potential vulnerabilities** they usually have, and propose effective remediation strategies.

- *Concentrate on **login systems in general***
- *The vulnerability can relate to any aspect of a login system, including user identification, authentication mechanisms, and session management*
- *Any common login system vulnerability is acceptable*
- ***For each identified potential vulnerability**, you need to:*
  - ***Describe the vulnerability***
  - ***Explain the risk***
  - ***Provide remediation strategy***

Vulnerabilities and remediation

1. Weak Password Policy
Description
Users are allowed to set weak passwords that are easily guessable or susceptible to brute-force attacks. Password complexities are not enforced leading to the login system being vulnerable to brute force, password spraying attacks etc.
Risk
Weak passwords increase the likelihood of unauthorized access to user accounts, leading to data breaches, identity theft, and financial losses.
Remediation
Implement and enforce a strong password policy that requires users to create complex passwords containing a mix of uppercase and lowercase letters, numbers, and special characters. Additionally, enforce password length and expiration requirements, and educate users about the importance of choosing strong passwords.

Vulnerabilities and remediation

2. Session Management Vulnerabilities
Description
The login system has weaknesses in session management, such as session fixation, session hijacking, or insufficient session expiration controls
Risk
Attackers can exploit session management vulnerabilities to gain unauthorized access to user sessions, manipulate user sessions, or impersonate legitimate users, compromising the confidentiality and integrity of user data.
Remediation
Implement secure session management practices, including using random and unique session identifiers, enforcing HTTPS for session communication, and implementing session expiration and invalidation mechanisms. Additionally, regularly rotate session tokens and monitor for suspicious activity to detect and mitigate session-related attacks.

Vulnerabilities and remediation

3. SQL Injection (SQLi)
Description
The login system is vulnerable to SQL injection (SQLi) attacks, allowing attackers to manipulate SQL queries by inserting malicious SQL code into input fields.
Risk
Attackers can exploit SQL injection vulnerabilities to bypass authentication mechanisms, extract sensitive information from the database, or execute arbitrary SQL commands, leading to data leakage, data manipulation, and unauthorized access
Remediation
Implement input validation and output encoding to sanitize user input and reject malicious SQL payloads effectively. Additionally, restrict database privileges to mitigate the impact of successful SQL injection attacks.

Create a threat Matrix

Dissect and categorize the 3 vulnerabilities that you have identified for the login system. Understanding these vulnerabilities from a strategic viewpoint will enable the company to allocate resources efficiently, prioritize remediation efforts, and maintain CryptoV4ult's reputation as a secure and reliable platform.

- *For each identified vulnerability, critically assess its potential to disrupt CryptoV4ult's operational functionality, erode customer trust, and impact financial stability.* ***Assign an impact level of 'Low', 'Medium', or 'High'*** *based on the evaluated potential consequences.*
- *Analyze the complexity and feasibility of exploiting each identified vulnerability. Consider the sophistication required for exploitation and the accessibility of the vulnerability to potential attackers.* ***Rate the likelihood of exploitation as 'Low', 'Medium', or 'High'****.*
- *Utilize the provided risk matrix framework to* ***map out the vulnerabilities*** *according to your assessments of their impact and exploit likelihood.*

Threat Matrix

Pathway (Vulnerability)	Impact Level	Likelihood Level
Weak Password Policy	High	Medium
Session Management Vulnerabilities	Medium	Medium
SQL Injection (SQLi)	High	HIgh

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

Impact / Likelihood	Low	Medium	High
High			**SQL Injection (SQLi)**
Medium		Weak Password Policy	Session Management Vulnerabilities
Low			

Section Three: Container Security

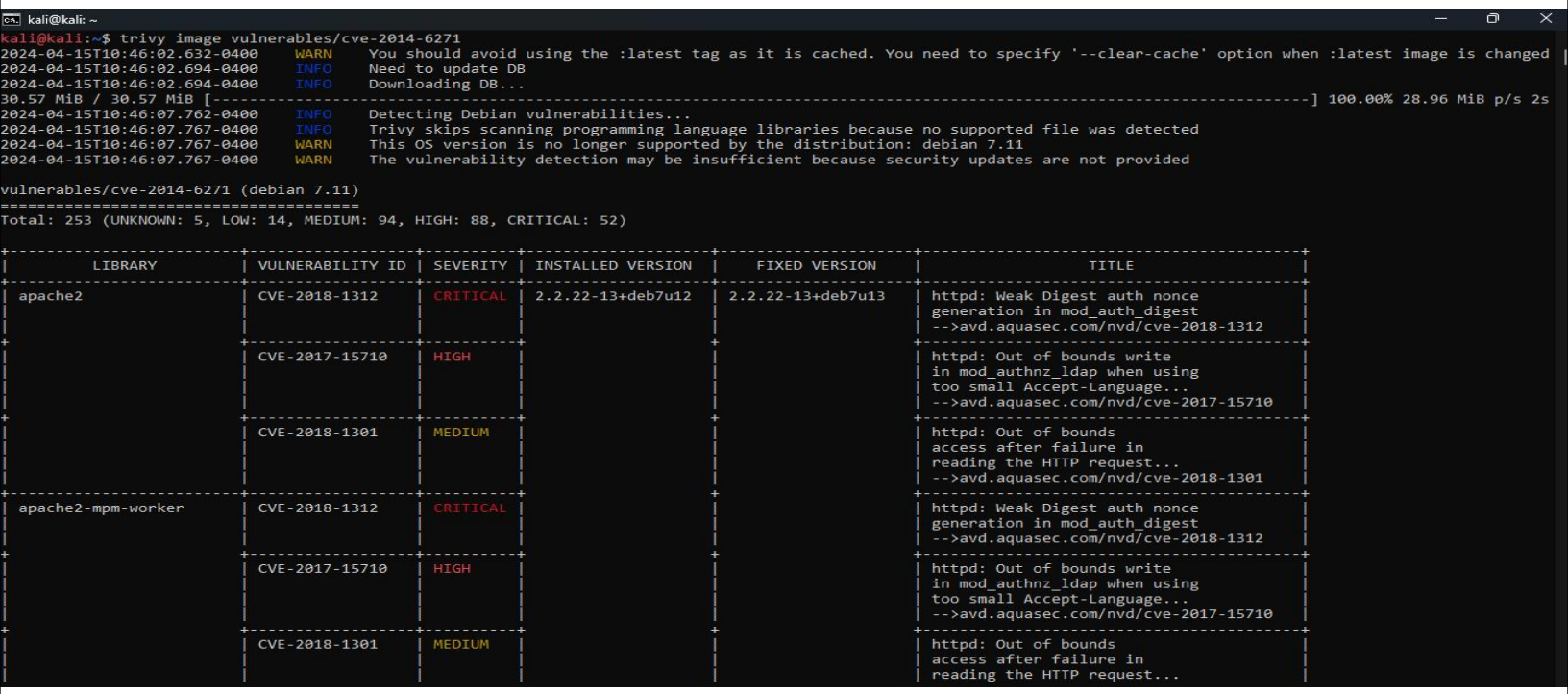
Container Security

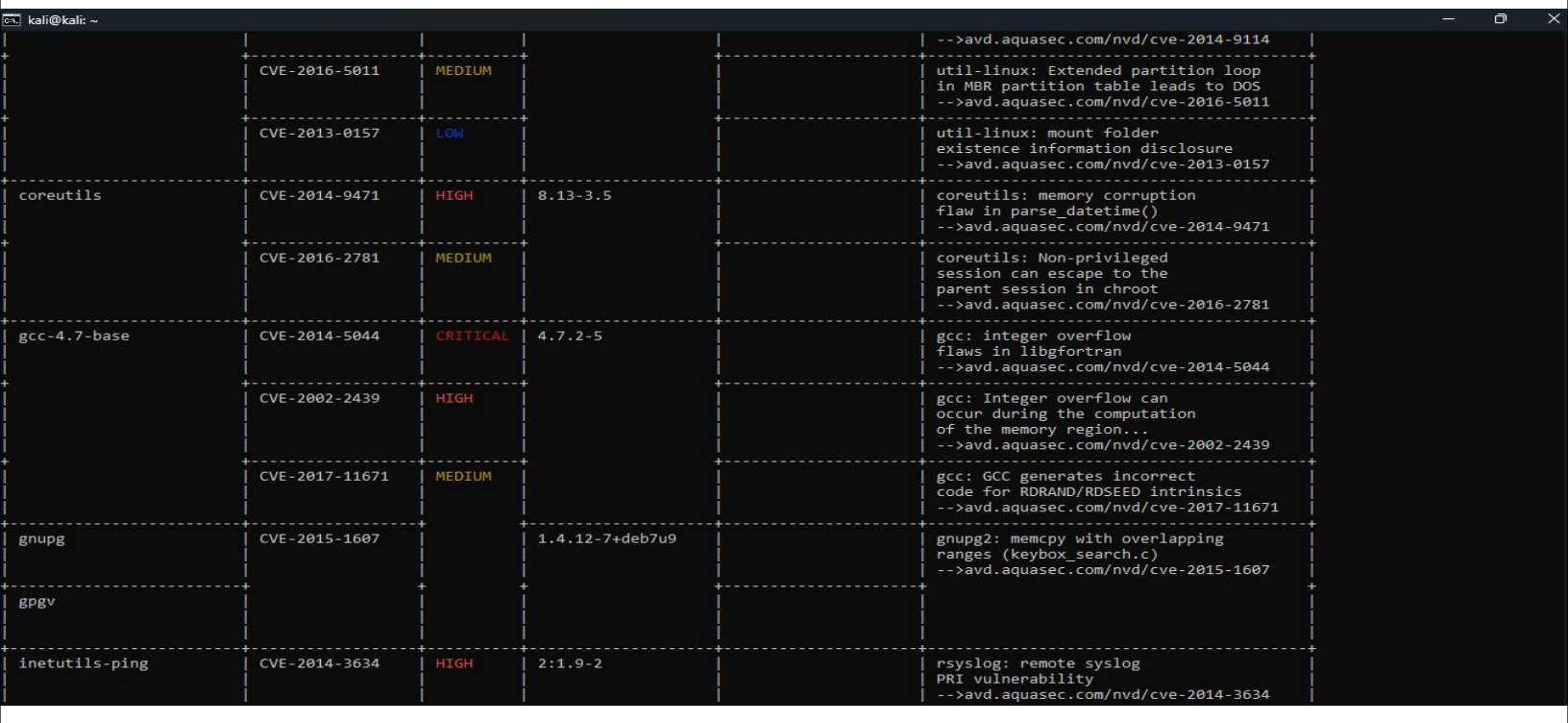
It is time to delve into the container services underpinning CryptoV4ult's application infrastructure by scanning for potential vulnerabilities. Scan one of the container services running in the application (located at vulnerables/cve-2014-6271) and identify potential vulnerabilities. Then, you will build a remediation plan to resolve some of the container vulnerabilities.

- *Using **Trivy**, run a **scan** against the container located at **vulnerables/cve-2014-6271**. You can run this scan from the Kali VM in the lab where Trivy is located or from your own computer*
- *Create a **screenshot** of the **Trivy scan results** (it does not have to show all the results) and place it on the next slide*
- ***Fill out the Report** to Fix Container Issues with at least 7 items*

Trivy scan screenshot

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





Report to Fix Container Issues

Fill out the report with at least 7 items.

Issues	Unpatched Software Version	Patched Software Version
apache2	2.2.22-13+deb7u12	2.2.22-13+deb7u13
bash	4.2+dfsg-0.1	4.2+dfsg-0.1+deb7u1
libapr1	1.4.6-3+deb7u1	1.4.6-3+deb7u2
libaprutil1	1.4.1-3	1.4.1-3+deb7u1
perl	5.14.2-21+deb7u5	5.14.2-21+deb7u6
libprocps0	1:3.3.3-3	1:3.3.3-3+deb7u1
libssl1.0.0	1.0.1t-1+deb7u2	1.0.1t-1+deb7u3

Section Four: API Security

API Security

Management has partnered with an external sales vendor and asked for a generic API to be developed that tracks user's data. Based on the data ingested they will create targeted sales advertisements to the customer base, this means a lot of confidential info about the users will be shared to 3rd party vendors.

You need to **identify 3 common API vulnerabilities** and propose effective remediation strategies. Keep in mind this code does not exist; this is the initial stages of development, and you are providing guidance to the engineering team. Feel free to make any assumptions about API features, implementations, and what private data might be shared.

- ***For each identified common API vulnerability:***
  - ***Describe the vulnerability***
  - ***Explain the risk***
  - ***Provide remediation strategy***

API Vulnerabilities and remediation

1.Broken Authentication
Description
Broken authentication occurs when the authentication mechanisms in the API are implemented incorrectly or are insufficiently secure, allowing attackers to compromise user accounts or gain unauthorized access.
Risk
Attackers can exploit broken authentication to hijack user sessions, access sensitive data, or perform unauthorized actions on behalf of authenticated users
Remediation
Implement secure authentication mechanisms such as multi-factor authentication (MFA), strong password policies, session management controls (e.g., session expiration, CSRF protection), and rate limiting to prevent brute force attacks

API Vulnerabilities and remediation

2. Security Misconfiguration
Description
Security misconfigurations occur when the API is deployed or configured with insecure settings, leaving it vulnerable to exploitation.
Risk
Attackers can exploit security misconfigurations to gain unauthorized access, escalate privileges, or perform other malicious actions.
Remediation
Regularly review and audit the API configuration to identify and remediate security misconfigurations. Follow security best practices and guidelines provided by the platform/framework used for API development. Automate configuration management processes to ensure consistency and adherence to security standards

API Vulnerabilities and remediation

3.Insufficient Logging and Monitoring
Description
Insufficient logging and monitoring occur when the API fails to adequately log and monitor security-related events, making it difficult to detect and respond to security incidents.
Risk
Attackers can perform malicious activities without being detected, leading to data breaches, unauthorized access, or prolonged compromises
Remediation
Implement comprehensive logging of security-relevant events (e.g., authentication attempts, access control failures) and monitor logs for suspicious activities. Use intrusion detection systems (IDS), security information and event management (SIEM) systems, and automated alerts to quickly identify and respond to security incidents.