SECURITY ASSESSMENT

ARSHAD TK
CBS-0004

26-01-2025

HOW TO USE THIS TEMPLATE

- We have provided these slides as a guide to ensure you submit all the required components to complete your project successfully.
- When presenting your project, remember that these slides are merely a guide. We strongly encourage you to embrace your creative freedom and make changes that reflect your unique vision as long as the required information is present.
- You can add slides to the template when your answers or screenshots do not fit on the previously provided pages.
- **Remember to add your name and the date to the cover page.**
- **Submit this file in a PDF**
- **Make sure the file is named in this format {FirstName}_{RollNO}.pdf**

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 1:
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. • Write a requirements document for task management features. • Identify and document security requirements(authentication, encryption, logging etc.).
Design
• Create a high-level architecture diagram for the application. • Design the database schema for tasks. • Conduct a threat modeling exercise to identify potential vulnerabilities in the architecture and database schema.

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. • Integrate security libraries for input validation, encryption and follow secure coding practices.
Testing
• Write and execute functional test cases. • Conduct browser compatibility testing. • Perform security testing, including penetration testing.

Transitioning to Secure SDLC

Deployment
• Deploy the application to Heroku. • Perform smoke testing on the deployed application. • Conduct a vulnerability assessment of the deployed environment, ensuring all external-facing endpoints are secure.
Maintenance
• Monitor application logs and fix reported issues. • Gather user feedback for future feature additions. • Implement continuous security monitoring and automated alerting 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.

Advocating for Secure SDLC

1. Enhanced Security Integration
By embedding security tasks throughout the SDLC phases, potential vulnerabilities are identified and mitigated early, reducing the risk of costly breaches. Also this ensures robust protection for user data and digital assets which is important for our organization.
2. Faster Response to Changes
The iterative nature of Secure SDLC allows for quick changes to handle new security threats or business needs. This flexibility is essential in our industry, where things change fast, and threats are always evolving.
3. Cost-Effective Risk Mitigation
Addressing security issues during early stages (e.g., requirements or design) is significantly less expensive than fixing vulnerabilities post-deployment. This proactive approach saves resources and reputations of our industry.
4. Improved Collaboration and Awareness
Secure SDLC fosters cross-functional collaboration between developers, security teams, and stakeholders. This improves awareness of security best practices and ensures security is a shared responsibility across the organization.
5.Enhanced Compliance and Trust
Following Secure SDLC frameworks helps us to stay compliant with regulations and industry standards. It shows the platform's dedication to security, builds trust with users, partners, and strengthens our reputation.

Section 2:
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. SQL Injection
Description
SQL injection occurs when attackers manipulate input fields (e.g., username or password fields) to inject malicious SQL commands, potentially granting unauthorized. This happens when user inputs aren't properly validated or sanitized.
Risk
If an SQL injection attack is successful, it can cause severe damage. Attackers could access sensitive data like usernames, passwords, credit card details, or personal information. In worse cases, they might gain control of the entire database, alter or delete records, or access other parts of the system. This could lead to data breaches, financial losses, reputational harm, and legal issues due to violations of data protection laws.
Remediation
To prevent SQL injection, use parameterized queries to separate queries from user input. Validate and sanitize inputs to allow only safe characters. Regular security testing with tools like SQLMap helps find vulnerabilities early. Apply the least privilege principle, and use a Web Application Firewall to block SQL injection attempts.

Vulnerabilities and remediation

2. Weak Password Policy
Description
A weak password policy occurs when a system allows simple, easily guessable passwords. Users often avoid complex passwords due to inconvenience or poor password management. This vulnerability opens the door for attacks like brute force or credential stuffing, where attackers use large combinations or leaked credentials to gain unauthorized access.
Risk
Attackers can use brute force to guess passwords, especially if they're simple or common. Credential stuffing allows attackers to exploit leaked credentials across multiple platforms. If successful, attackers can access sensitive data, initiate fraud, or cause other harm. This compromises individual accounts and can lead to large-scale data breaches, financial losses, and reputational damage for organizations.
Remediation
To mitigate weak password risks, enforce strong policies with complex requirements and regular password changes. Implement account lockouts after failed login attempts. Encourage password managers and educate users on the importance of unique, non-reused passwords.

Vulnerabilities and remediation

3. Lack of Multifactor Authentication (MFA)
Description
The lack of Multifactor Authentication (MFA) creates a major security gap. MFA adds an extra layer of protection by requiring multiple verification forms. Without MFA, passwords alone are vulnerable to attacks like phishing or brute force, making unauthorized access easier for attackers.
Risk
Without MFA, attackers who obtain credentials through phishing or other means can easily access systems, increasing the risk of data breaches, fraud, or manipulation. MFA reduces this risk by requiring additional layers of authentication, making it harder for attackers to exploit stolen credentials. Its absence leaves systems and accounts vulnerable, especially when passwords are compromised.
Remediation
To reduce MFA risks, organizations should enforce MFA for all accounts with access to sensitive data. Options include OTPs, hardware tokens, or biometrics. Risk-based authentication can prompt MFA for unusual activity. Regularly update MFA methods and offer flexible options, like authentication apps or SMS codes, to maintain security.

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

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

Impact / Likelihood	Low	Medium	High
High			• Weak Password Policy • Lack of Multifactor Authentication (MFA)
Medium			• SQL Injection
Low			

Section 3:
Container Security

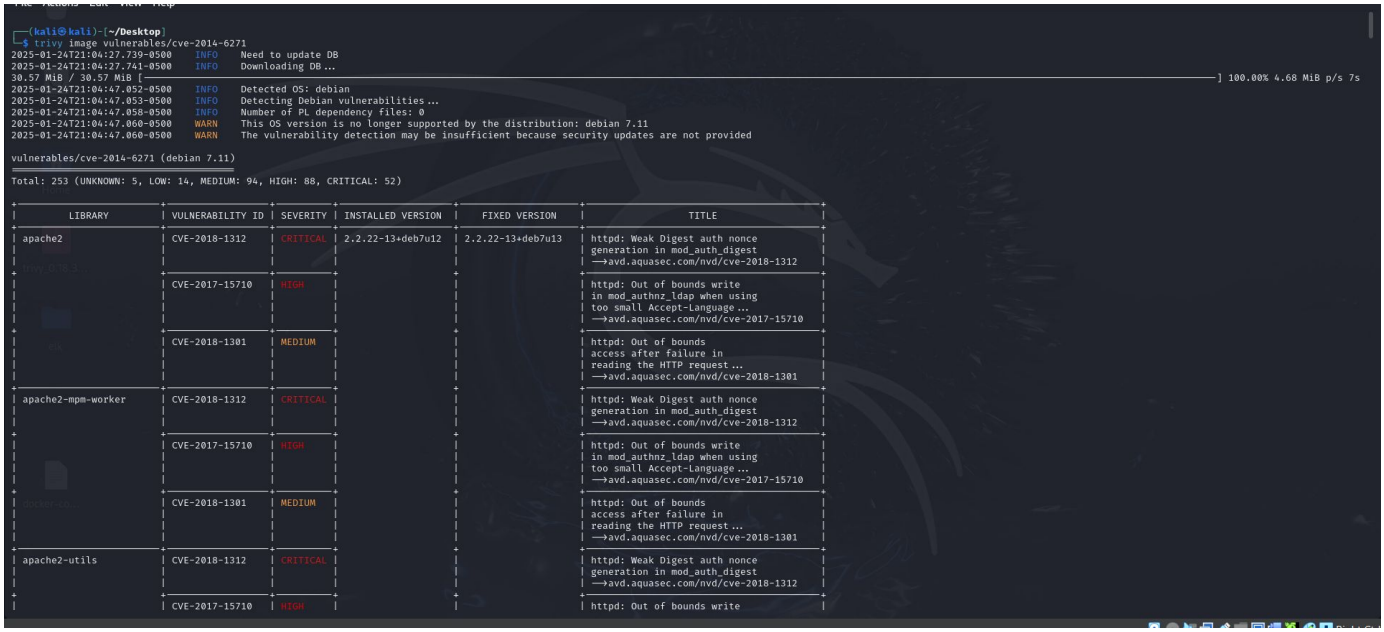
Container Security

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*

Project Information Slide

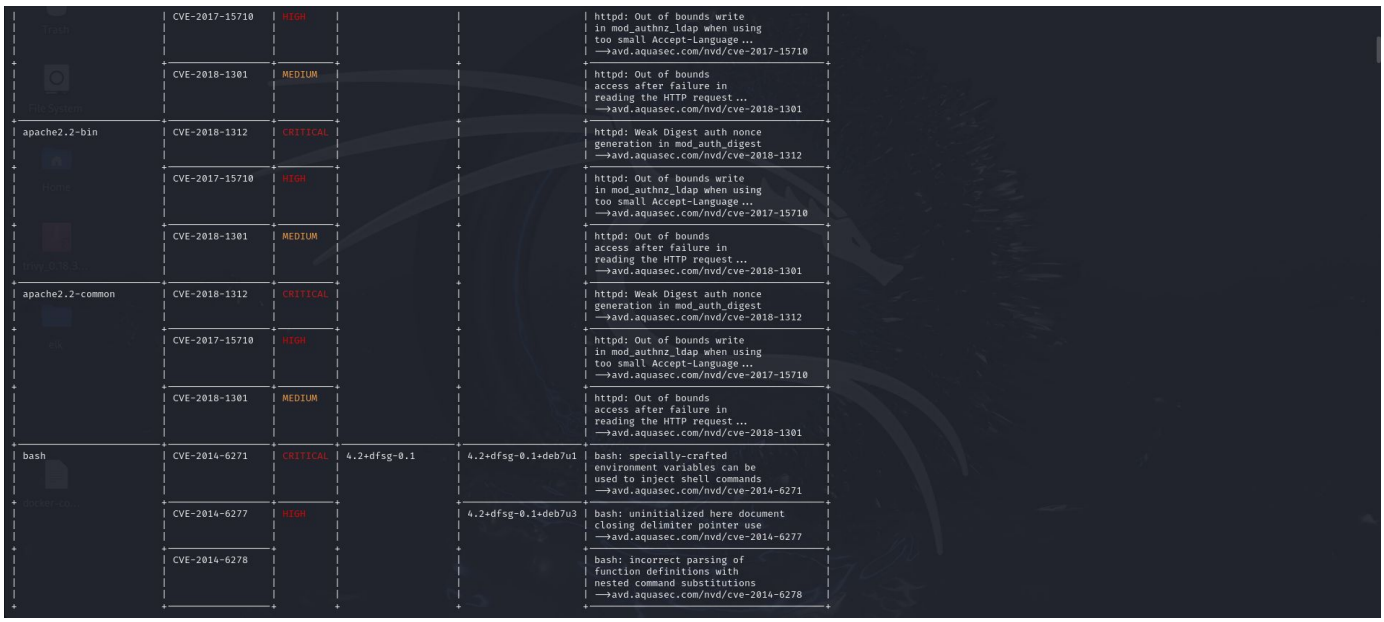
Trivy scan screenshot



```
┌──(kali㉿kali)-[~/Desktop]
└─$ trivy image vulnerables/cve-2014-6271
2025-01-24T21:04:27.739-0500    INFO    Need to update DB
2025-01-24T21:04:27.741-0500    INFO    Downloading DB...
30.57 MiB / 30.57 MiB [-------------------------------------------------------------------------------------------------------------------------------------------------------------] 100.00% 4.68 MiB p/s 7s
2025-01-24T21:04:47.052-0500    INFO    Detected OS: debian
2025-01-24T21:04:47.053-0500    INFO    Detecting Debian vulnerabilities...
2025-01-24T21:04:47.058-0500    INFO    Number of PL dependency files: 0
2025-01-24T21:04:47.060-0500    WARN    This OS version is no longer supported by the distribution: debian 7.11
2025-01-24T21:04:47.060-0500    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  |
|                       +------------------+----------+                   |               +---------------------------------------+
|                       | CVE-2017-15710   | HIGH     |                   |               | httpd: Out of bounds write            |
```



```
|                       | 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  |
+-----------------------+------------------+----------+                   |               +---------------------------------------+
| apache2.2-bin         | 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  |
+-----------------------+------------------+----------+                   |               +---------------------------------------+
| apache2.2-common      | 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  |
+-----------------------+------------------+----------+-------------------+---------------+---------------------------------------+
| bash                  | CVE-2014-6271    | CRITICAL | 4.2+dfsg-0.1      | 4.2+dfsg-0.1+deb7u1 | bash: specially-crafted           |
|                       |                  |          |                   |               | environment variables can be          |
|                       |                  |          |                   |               | used to inject shell commands         |
|                       |                  |          |                   |               | -->avd.aquasec.com/nvd/cve-2014-6271  |
|                       +------------------+----------+                   +---------------+---------------------------------------+
|                       | CVE-2014-6277    | HIGH     |                   | 4.2+dfsg-0.1+deb7u3 | bash: uninitialized here document |
|                       |                  |          |                   |               | closing delimiter pointer use         |
|                       |                  |          |                   |               | -->avd.aquasec.com/nvd/cve-2014-6277  |
|                       +------------------+          +                   +               +---------------------------------------+
|                       | CVE-2014-6278    |          |                   |               | bash: incorrect parsing of            |
|                       |                  |          |                   |               | function definitions with             |
|                       |                  |          |                   |               | nested command substitutions          |
|                       |                  |          |                   |               | -->avd.aquasec.com/nvd/cve-2014-6278  |
+-----------------------+------------------+----------+-------------------+---------------+---------------------------------------+
```

Report to Fix Container Issues

*Fill out the report with 7 items. Make sure to write the **Issues in the correct form of (Application Name: CVE number).***

Vulnerability Name	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
perl (CVE-2018-6913)	5.14.2-21+deb7u5	5.14.2-21+deb7u6
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

Section 4:
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***

Vulnerabilities and remediation

1. Broken Object-Level Authorization (BOLA)
Description
The API fails to enforce proper access controls on objects (e.g., user accounts, files). Attackers can manipulate parameters to access unauthorized data.
Risk
This vulnerability can lead to unauthorized access to sensitive user data, such as financial information or personal identifiers. It also increases the risk of data breaches, which can result in violations of regulations like GDPR.
Remediation
To mitigate BOLA, implement role-based access control (RBAC) or attribute-based access control (ABAC) to ensure users can only access resources they are authorized to use. Use secure resource identifiers, to avoid exposing sensitive IDs directly in API requests. Additionally, monitor and log access attempts to detect and respond to suspicious activity in real time.

Vulnerabilities and remediation

2. Lack of Rate Limiting and Abuse Protection
Description
APIs without rate limiting are vulnerable to abuse, such as brute-force attacks, data scraping, or denial of service (DoS) attacks. Attackers can overwhelm the API with excessive requests, leading to service disruptions or unauthorized access to sensitive data.
Risk
The absence of rate limiting can result in compromised accounts through brute-force password guessing or data leakage through scraping. It can also lead to increased infrastructure costs due to excessive traffic, as the system struggles to handle the load.
Remediation
Implement rate limiting to restrict the number of requests a user or IP address can make within a specific timeframe.Integrate Web Application Firewalls (WAFs) and API gateways to manage and throttle traffic efficiently, ensuring the system remains available and secure.

Vulnerabilities and remediation

3. Insufficient Data Protection
Description
APIs that transmit sensitive data without proper encryption or validation are vulnerable to interception, tampering, or injection attacks. This includes personal identifiable information (PII), financial data, or other confidential information shared with third-party vendors.
Risk
Insufficient data protection can lead to data interception during transmission, such as through man-in-the-middle attacks. It also increases the risk of injection attacks.Further damaging the platform's reputation.
Remediation
Enforce HTTPS with strong TLS configurations to encrypt all API communications and ensure data is encrypted at rest using robust algorithms like AES-256. Validate and sanitize all incoming API requests to reject unexpected or malicious payloads. Conduct regular security audits.