SECURITY ASSESSMENT

NILOTPALSARMA
CBS-0180

Project Information Slide

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

Project Information Slide

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 (20)

Project Information Slide

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***

Project Information Slide

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

Collect and document functional requirements such as blockchain transaction flows . Determine security and compliance requirements such as PCI-DSS, GDPR, and crypto-specific standards (e.g., FIPS 140-2 for cryptographic modules)Perform threat modeling with emphasis on blockchain threats (e.g., double-spending, 51% attack) and privacy of user data .Establish security acceptance criteria such as cryptography key management policy and wallet secure handling

Design

Design system architecture consisting of blockchain nodes, wallet services, and API gateways

Design safe key management and hardware security module (HSM) integration of private keys

Prescribe data flow diagrams with assurance of encryption at rest and during transit, off-chain storage as well as blockchain data

Conduct security design reviews with specialist blockchain security staff and develop smart contract vulnerability mitigation plans

Choose approved cryptographic libraries and warrant third-party packages adhere to rules-based standards

Project Information Slide

Transitioning to Secure SDLC

Development

Implement code in secure coding standards customized for blockchain as well as cryptography operations

Incorporate SAST tools set up to identify crypto-related coding defects and standard vulnerabilities

Enforce strict input validation and output encoding, particularly on smart contract interfaces and APIs

Perform peer code reviews by blockchain security experts with emphasis on transaction integrity and key management

Testing

Functional and integration testing with blockchain transaction validation

Run DAST and penetration testing with a focus on blockchain node security and wallet interfaces

Verify security controls like multi-factor authentication (MFA) for accessing wallets and signing transactions

Test for weakness like replay attacks, smart contract flaws, and incorrect access controls

Conduct security regression tests following patches or smart contract modifications

Transitioning to Secure SDLC

Deployment

Secure network configurations and hardened blockchain node settings with firewalls and DDoS mitigation
Conduct security configuration testing on node software and wallet service compliance automatically
Safely store secrets utilizing HSM or secure vault to store private keys and API tokens
Plan for incident responses for blockchain forking, compromise of key materials, or for unusual transaction anomalies

Maintenance

Ongoing monitoring of blockchain network health, transaction anomalies, and security incidents
Implementing timely security patches to blockchain clients, wallet software, and dependencies
Performing regular security audits such as smart contract code reviews and compliance scans
Ongoing security training on blockchain threat landscape and secure key management
Updating security policies to mitigate new blockchain threats and regulatory updates

Project Information Slide

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!*

Project Information Slide

Advocating for Secure SDLC

1. Early Vulnerability Detection

Having security integrated from the beginning enables us to detect and correct vulnerabilities early on, minimizing expensive late-cycle fixes and safeguarding valuable crypto assets

2. Faster Response to Threats

Rapid adaptation to new vulnerabilities through agile security practices is essential to sustaining trust in our high speed cryptocurrency ecosystem.

3. Continuous Security Integration

Secure SDLC enables constant security testing during development, complementary to our agile methodology and solidifying solid defenses against future threats

4. Improved Compliance and Audit Readiness

Incorporating security controls along the development path streamlines satisfying regulatory mandates and places us in a solid position for audits without eleventh-hour rushes

5. Improved Developer Security Awareness

Engaging developers in security procedures creates a security-first culture that minimizes human mistakes and improves the overall resilience of our platform

Section 2:

Vulnerabilities and Remediation (35)

Project Information Slide

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. **Credential Stuffing**
Description
Attackers try to gain access on CryptoV4ult by using large troves of stolen username/paswords from other breaches along with this vulnerability. Automated login attempts with reused credentials are not sufficiently guarded against by the login system.
Risk
When users use the same password across sites, attackers can sometimes gain unauthorized access to user accounts. Compromise of sensitive data, account takeover, and potential financial damage or fraud follow. it also damages user confidence and reputation of platform
Remediation
Introduce a second level of authentication by means of multifactor authentication (MFA). To stop automatic login attempts, use IP reputation checks and rate limiting. Use strong unique passwords yourself and employ credential stuffing identification tools.

Vulnerabilities and remediation

2. **Poor Password Policy**
Description
By means of the login system, people may establish simple or easily guessable passwords without the need of complexity, length, or blacklist checks. This makes guessing attempts or brute force simplification
Risk
Guess or brute force poor passwords could allow attackers to hijack accounts and therefore unauthorized access, data loss, and potential financial damage. Automated attacks have more chances of succeeding when weak passwords also help.
Remediation
Insist on strict password guidelines including minimum length, complexity (upper case, lower case, digit, symbol), and exclusion of often used or compromised passwords. Inform users on how to generate strong passwords and enforce password strength indicators.

Project Information Slide

Vulnerabilities and remediation

3. **Session Dating**
Description
When the login system does not invalidate or renew session IDs after successful login, allowing the hacker to pin a session ID and take over a user session, so iniquity is present here
Risk
Data theft, fraud, or account abuse can result from attackers taking over verified sessions, impersonating the users, and running unsanctioned operations. The integrity of the authentication process is here compromised
Remediation
Make absolutely sure the system wipes out expired sessions and generates a new session ID upon successful login. For session management, use secure, HttpOnly, and SameSite cookies. Offer logout capability in addition to session timeout feature.

Project Information Slide

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.*

Project Information Slide

Threat Matrix

Pathway (Vulnerability)	Impact Level	Likelihood Level
Credential Stuffing	High	High
Weak Password Policy	Medium	Medium
SessionDating	Low	Low

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

Impact	Low	Medium	High
Likelihood			
High			Credential Stuffing
Medium		Weak password Policy	
Low	SessionDating		

Section 3:

Container Security (20)

Project Information Slide

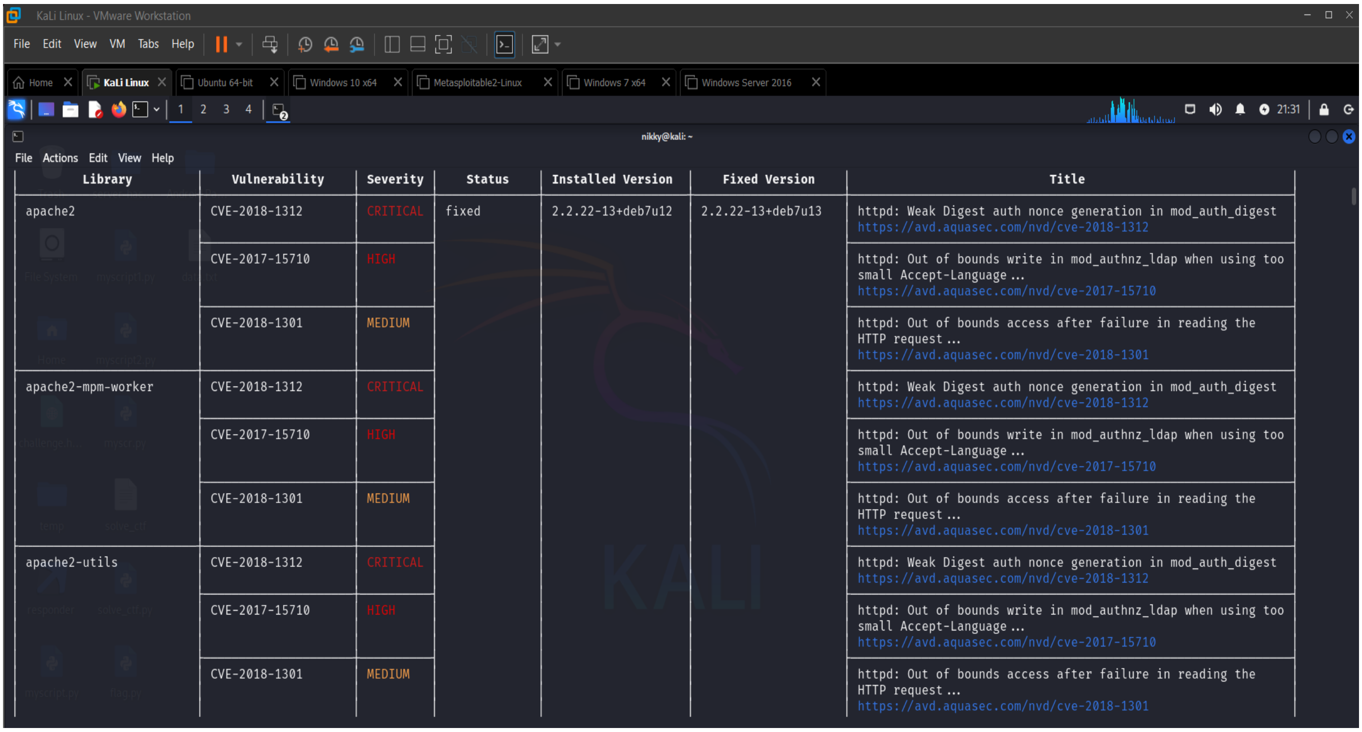
Container Security

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.

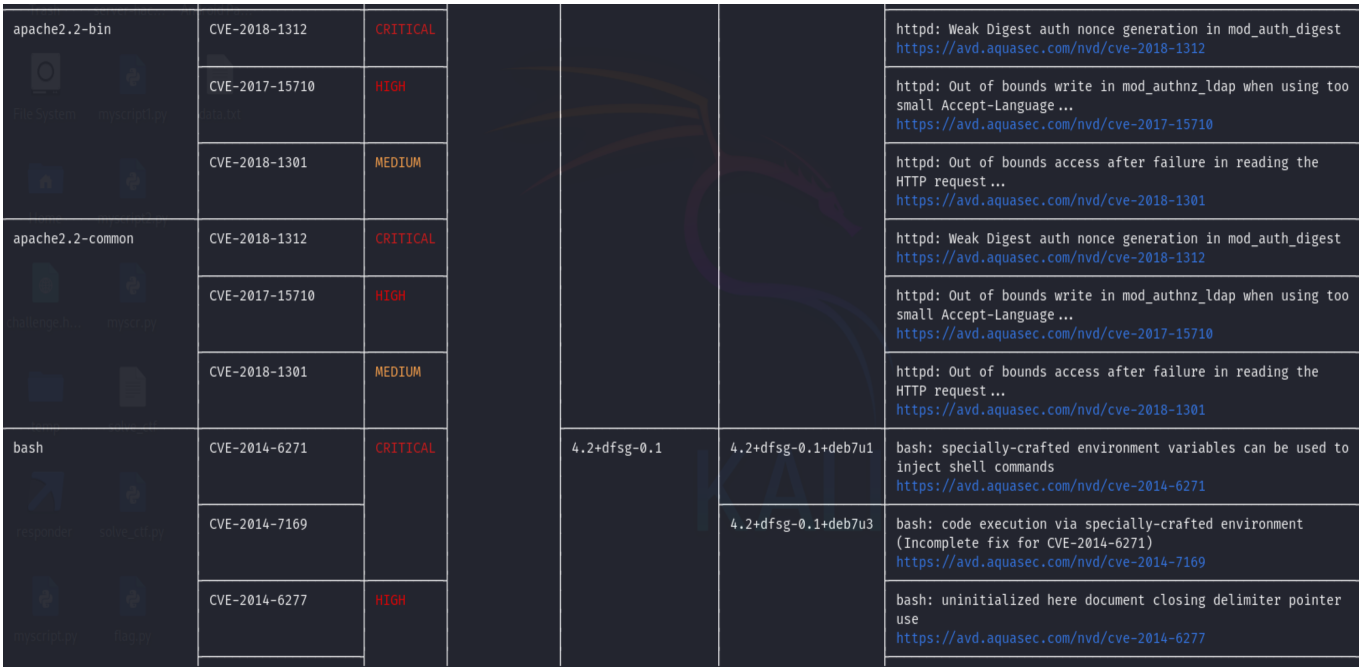
- *Using **Trivy**, run a **scan** against the container located at **vulnerables/cve-2014-6271**. You can run this scan from the Kali VM in the lab where Trivy is located or from your own computer*
- *Create a **screenshot** of the **Trivy scan results** (it does not have to show all the results) and 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



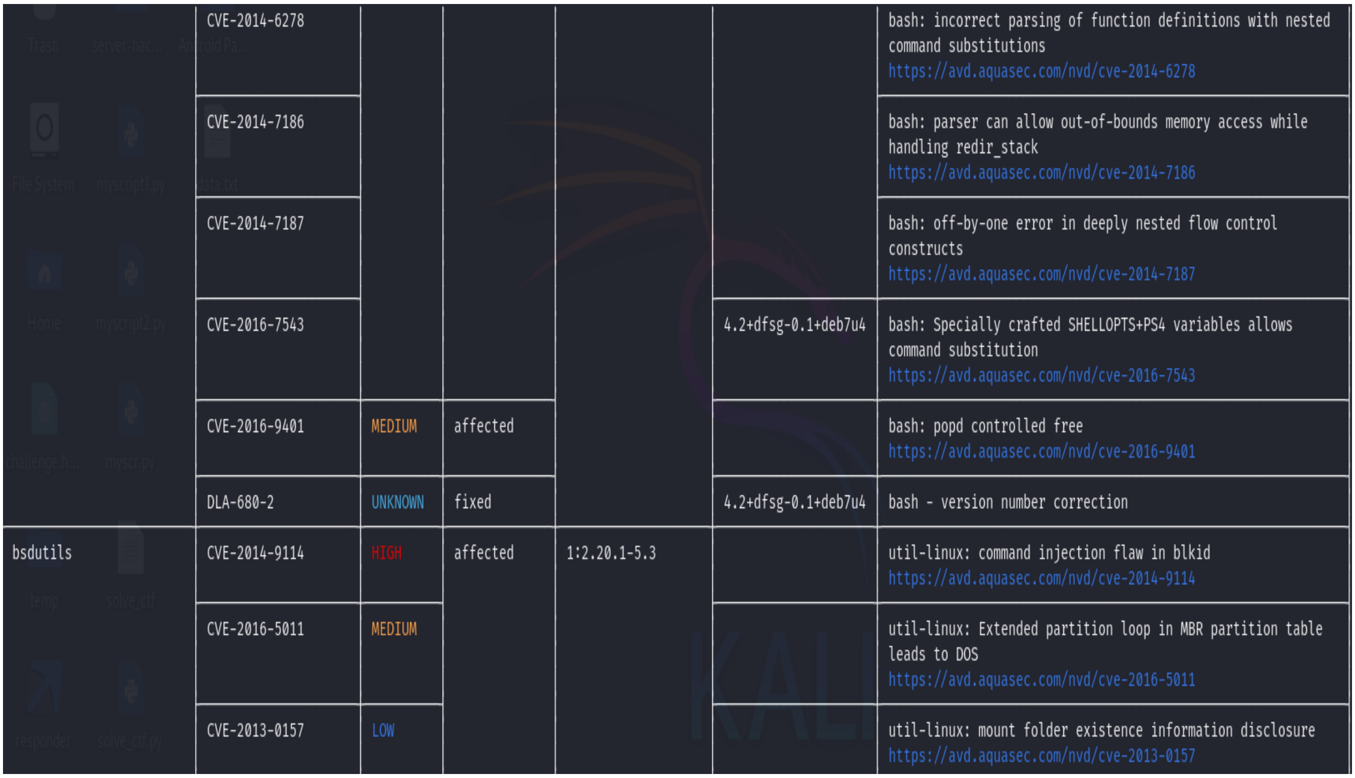
```
│     Library      │  Vulnerability  │ Severity │ Status │ Installed Version │ Fixed Version    │                          Title
│ apache2          │ CVE-2018-1312   │ CRITICAL │ fixed  │ 2.2.22-13+deb7u12 │ 2.2.22-13+deb7u13 │ httpd: Weak Digest auth nonce generation in mod_auth_digest
│                  │                 │          │        │                   │                   │ https://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 ...
│                  │                 │          │        │                   │                   │ https://avd.aquasec.com/nvd/cve-2017-15710
│                  │ CVE-2018-1301   │ MEDIUM   │        │                   │                   │ httpd: Out of bounds access after failure in reading the
│                  │                 │          │        │                   │                   │ HTTP request ...
│                  │                 │          │        │                   │                   │ https://avd.aquasec.com/nvd/cve-2018-1301
│ apache2-mpm-worker │ CVE-2018-1312 │ CRITICAL │        │                   │                   │ httpd: Weak Digest auth nonce generation in mod_auth_digest
│                  │                 │          │        │                   │                   │ https://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 ...
│                  │                 │          │        │                   │                   │ https://avd.aquasec.com/nvd/cve-2017-15710
│                  │ CVE-2018-1301   │ MEDIUM   │        │                   │                   │ httpd: Out of bounds access after failure in reading the
│                  │                 │          │        │                   │                   │ HTTP request ...
│                  │                 │          │        │                   │                   │ https://avd.aquasec.com/nvd/cve-2018-1301
│ apache2-utils    │ CVE-2018-1312   │ CRITICAL │        │                   │                   │ httpd: Weak Digest auth nonce generation in mod_auth_digest
│                  │                 │          │        │                   │                   │ https://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 ...
│                  │                 │          │        │                   │                   │ https://avd.aquasec.com/nvd/cve-2017-15710
│                  │ CVE-2018-1301   │ MEDIUM   │        │                   │                   │ httpd: Out of bounds access after failure in reading the
│                  │                 │          │        │                   │                   │ HTTP request ...
│                  │                 │          │        │                   │                   │ https://avd.aquasec.com/nvd/cve-2018-1301
```



```
│ apache2.2-bin    │ CVE-2018-1312   │ CRITICAL │        │                   │                      │ httpd: Weak Digest auth nonce generation in mod_auth_digest
│                  │                 │          │        │                   │                      │ https://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 ...
│                  │                 │          │        │                   │                      │ https://avd.aquasec.com/nvd/cve-2017-15710
│                  │ CVE-2018-1301   │ MEDIUM   │        │                   │                      │ httpd: Out of bounds access after failure in reading the
│                  │                 │          │        │                   │                      │ HTTP request ...
│                  │                 │          │        │                   │                      │ https://avd.aquasec.com/nvd/cve-2018-1301
│ apache2.2-common │ CVE-2018-1312   │ CRITICAL │        │                   │                      │ httpd: Weak Digest auth nonce generation in mod_auth_digest
│                  │                 │          │        │                   │                      │ https://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 ...
│                  │                 │          │        │                   │                      │ https://avd.aquasec.com/nvd/cve-2017-15710
│                  │ CVE-2018-1301   │ MEDIUM   │        │                   │                      │ httpd: Out of bounds access after failure in reading the
│                  │                 │          │        │                   │                      │ HTTP request ...
│                  │                 │          │        │                   │                      │ https://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
│                  │                 │          │        │                   │                      │ https://avd.aquasec.com/nvd/cve-2014-6271
│                  │ CVE-2014-7169   │          │        │                   │ 4.2+dfsg-0.1+deb7u3 │ bash: code execution via specially-crafted environment
│                  │                 │          │        │                   │                      │ (Incomplete fix for CVE-2014-6271)
│                  │                 │          │        │                   │                      │ https://avd.aquasec.com/nvd/cve-2014-7169
│                  │ CVE-2014-6277   │ HIGH     │        │                   │                      │ bash: uninitialized here document closing delimiter pointer
│                  │                 │          │        │                   │                      │ use
│                  │                 │          │        │                   │                      │ https://avd.aquasec.com/nvd/cve-2014-6277
```

Project Information Slide

Trivy scan screenshot



	CVE-2014-6278					bash: incorrect parsing of function definitions with nested command substitutions https://avd.aquasec.com/nvd/cve-2014-6278
	CVE-2014-7186					bash: parser can allow out-of-bounds memory access while handling redir_stack https://avd.aquasec.com/nvd/cve-2014-7186
	CVE-2014-7187					bash: off-by-one error in deeply nested flow control constructs https://avd.aquasec.com/nvd/cve-2014-7187
	CVE-2016-7543				4.2+dfsg-0.1+deb7u4	bash: Specially crafted SHELLOPTS+PS4 variables allows command substitution https://avd.aquasec.com/nvd/cve-2016-7543
	CVE-2016-9401	MEDIUM	affected			bash: popd controlled free https://avd.aquasec.com/nvd/cve-2016-9401
	DLA-680-2	UNKNOWN	fixed		4.2+dfsg-0.1+deb7u4	bash - version number correction
bsdutils	CVE-2014-9114	HIGH	affected	1:2.20.1-5.3		util-linux: command injection flaw in blkid https://avd.aquasec.com/nvd/cve-2014-9114
	CVE-2016-5011	MEDIUM				util-linux: Extended partition loop in MBR partition table leads to DOS https://avd.aquasec.com/nvd/cve-2016-5011
	CVE-2013-0157	LOW				util-linux: mount folder existence information disclosure https://avd.aquasec.com/nvd/cve-2013-0157

Project Information Slide

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
apache2: CVE-2017-15710	2.2.22-13+deb7u12	Not specified
bash: CVE-2014-6271	4.2+dfsg-0.1	4.2+dfsg-0.1+deb7u1
bash: CVE-2016-9401	Not specified	Not specified
bsdutils: CVE-2014-9114	1:2.20.1-5.3	Not specified
apache2-utils: CVE-2018-1301	Not specified	Not specified
bash: CVE-2016-7543	Not specified	4.2+dfsg-0.1+deb7u4

Section 4:

API Security (15)

Project Information Slide

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**
Description
This vulnerability manifests if the API fails to properly verify that the authenticated user is authorized to access or modify a specific resource or data object. If, for example, the API lets a third party or user access some other user's data by modifying object IDs or parameters without proper permission checks
Risk
50055.77level lower. ***Unauthorised third parties or hackers may gain access sensitive other users' personal information, thus infringing privacy, data breaches, and regulatory noncompliance (e.g., GDPR). Particularly noteworthy is disclosure of sensitive user information externally***
Remediation
All API endpoints making use of user data should contain proper authorization verifications. Ensure the API verifies the identity of the user and confirms access permission to view or modify the needed material. Impose least privilege concepts and use userspecific tokens.

Vulnerabilities and remediation

2. **Excessive Data Exposure**
Description
This situation occurs when the API within its responses returns excess data than needed, such as sensitive, confidential data not needed by the third party or client. Returning full user profiles with PII (Personally Identifiable Information) when only few are needed for focused advertising
Risk
Exposure of excessive data increases the attack surface and the risk of leakage of sensitive information. Breaking into any middleman or third-party provider might enable attackers to gain excessive sensitive information, thus increasing privacy risks and likelihood of misuse
Remediation
Design the API to deliver only the bare minimum required information fields for the proposed usage scenario. Implement data filtering, field-level access controls, and data masking techniques. Perform regular data reviews to ensure accidental disclosure of sensitive data is not present

Vulnerabilities and remediation

3. **Lack of good rate limiting and throttling**
Description
Without rate restrictions, APIs allow hackers or malicious users to send any number of requests, potentially leading to misuse such as data scraping, denial of service, or brute force attacks
Risk
Attackers would be able to automate the requests to obtain large quantities of sensitive user data or flood the API and disrupt service. This would lead to data breaches or decreased availability of service to legitimate customers and contacts.
Remediation
Over a time frame, implement robust rate limiting and throttling mechanisms to restrict API requests that each user or client can make. To establish these limits and monitor for aberrant usage patterns, utilize API gateways or management tools.