UDACITY

CryptoV4ult Enterprise Security Review

Hazrat Umer

16 Aug 2024

https://www.linkedin.com/in/hazrat-umer/

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
Tasks: Conduct user interviews to gather functional requirements. Write a requirements document for task management features. **Security related task:** Identify and document security related requirements of users for example data encryption, user authentication and access control.
Design
Tasks: Create a high-level architecture diagram for the application. Design the database schema for tasks. **Security-Related Task:** Perform threat modeling and risk assessment by Identifying threats and vulnerabilities in the system design, and document mitigation strategies.

Transitioning to Secure SDLC

Development
Tasks: 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. **Security-related task:** Integrate secure coding practices by ensuring that secure coding standards are followed, such as input validation, output encoding, and proper error handling.
Testing
Tasks: Write and execute functional test cases. Conduct browser compatibility testing. **Security-related task:** Perform security testing such as conducting a vulnerability scan, penetration testing, and code review to identify different security flaws.

Transitioning to Secure SDLC

Deployment
Tasks: Deploy the application to Heroku. Perform smoke testing on the deployed application. **Security-related task:** Implement logging and monitoring solution to detect and respond to security incidents after deployment.
Maintenance
Tasks: Monitor application logs and fix reported issues. Gather user feedback for future feature additions. **Security-related task:** Regularly apply security patches and updates to the systems and applications and ensure that all its dependencies are up-to-date with the latest security patches.

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. Early Detection of Vulnerabilities
Through threat modeling and security testing, developers and security teams can find potential threats and vulnerabilities early in the development process. This will help in reduction of production time and development cost.
2. *Agile and Adaptive Development.*
Compared to the Waterfall model, In Secure SDLC iterative development is used, helping us respond quickly to market changes and emerging threats.
3. *Improved Collaboration and Communication*
In Secure SDLC there is a mechanism of continuous feedback loops between different teams such as development, security, and operations, this reduces the communication gap between various teams and helps organizations in fulfilling their security needs.
4. *Cost Reduction:*
Early code reviews in Secure SDLC discover security-related vulnerabilities and misconfigurations early in the initial stages which helps in cost reduction, at initial stages there will be less cost to resolve the issues.
5. Reduced Risk and Compliance
By applying different security practices, Secure SDLC helps reduce the risk of data breaches and adherence to industry regulations, it increases customer trust, and builds reputation.

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. Brute Force Attacks
Description
When the attacker tries different combinations of usernames and passwords until the correct username and password are found. Attackers make use of automated tools through which they can try millions of combinations of usernames and passwords per second if the password is not strong and if there is no multi factor authentication or rate limiting feature available then the attacker can easily compromise the password and can get access to a user's data.
Risk
If an attacker succeeds in executing a brute force attack, then he/she can gain unauthorized access to the user's confidential and sensitive data. This can lead to data breaches, unauthorized financial transactions, or data exfiltration.
Remediation
There should be an account lockout mechanism implemented. After a certain number of failed attempts, the accounts should be temporarily locked. Enforce strong password policies, implement CAPTCHA challenges, and enforce users to enable multi-factor authentication.

Vulnerabilities and remediation

2. Session Hijacking
Description
In Session Hijacking vulnerability threat actors gain access to a session between a client and a web server. Here, an attacker first intercepts the active session between the user and a website and then takes over it. This vulnerability can be exploited by attackers either through stealing session cookies or by exploiting weaknesses in the session management system.
Risk
Through Session Hijacking an intruder (attacker) can gain unauthorized access to a user's sensitive data, and he/she can do unauthorized transactions and other malicious activities.
Remediation
Using secure cookies by marking session cookies with the "Secure" and "HttpOnly" flags to prevent them from accessing through client-side scripts. Only transmit them through HTTPs connection. There should be proper session timeout implemented, session IDs should be regenerated after login, and password changes.

Vulnerabilities and remediation

3. Cross-Site Scripting (XSS) in Login Pages
Description
XSS stands for Cross Site Scripting is a web security vulnerability that allows attackers to inject malicious scripts into web pages that are viewed by other users. If an attacker succeeds in injecting malicious XXS script into the login page, then he/she can steal user credentials or session tokens when other users interact with the same compromised page.
Risk
Through Cross-Site Scripting (XSS), attackers can capture the login credentials of other users, session tokens, and other types of confidential or sensitive information. Through this vulnerability, an intruder can access users' accounts in an unauthorized way, which can lead to data theft and compromise the integrity of data.
Remediation
XSS can be remediated by sanitizing and validating the user's input. Ensuring that all the injection points such as login pages etc. are properly validated and sanitized so that it can prevent the injection of malicious scripts. Use a strong content security policy to prevent the execution of unauthorized scripts. Use appropriate response headers.

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
Brute Force Attacks	Medium	Medium
Session Hijacking	High	Medium
Cross-Site Scripting (XSS)	High	High

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

Impact / Likelihood	Low	Medium	High
High			Cross-Site Scripting (XSS)
Medium		Brute Force Attacks	Session Hijacking
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*

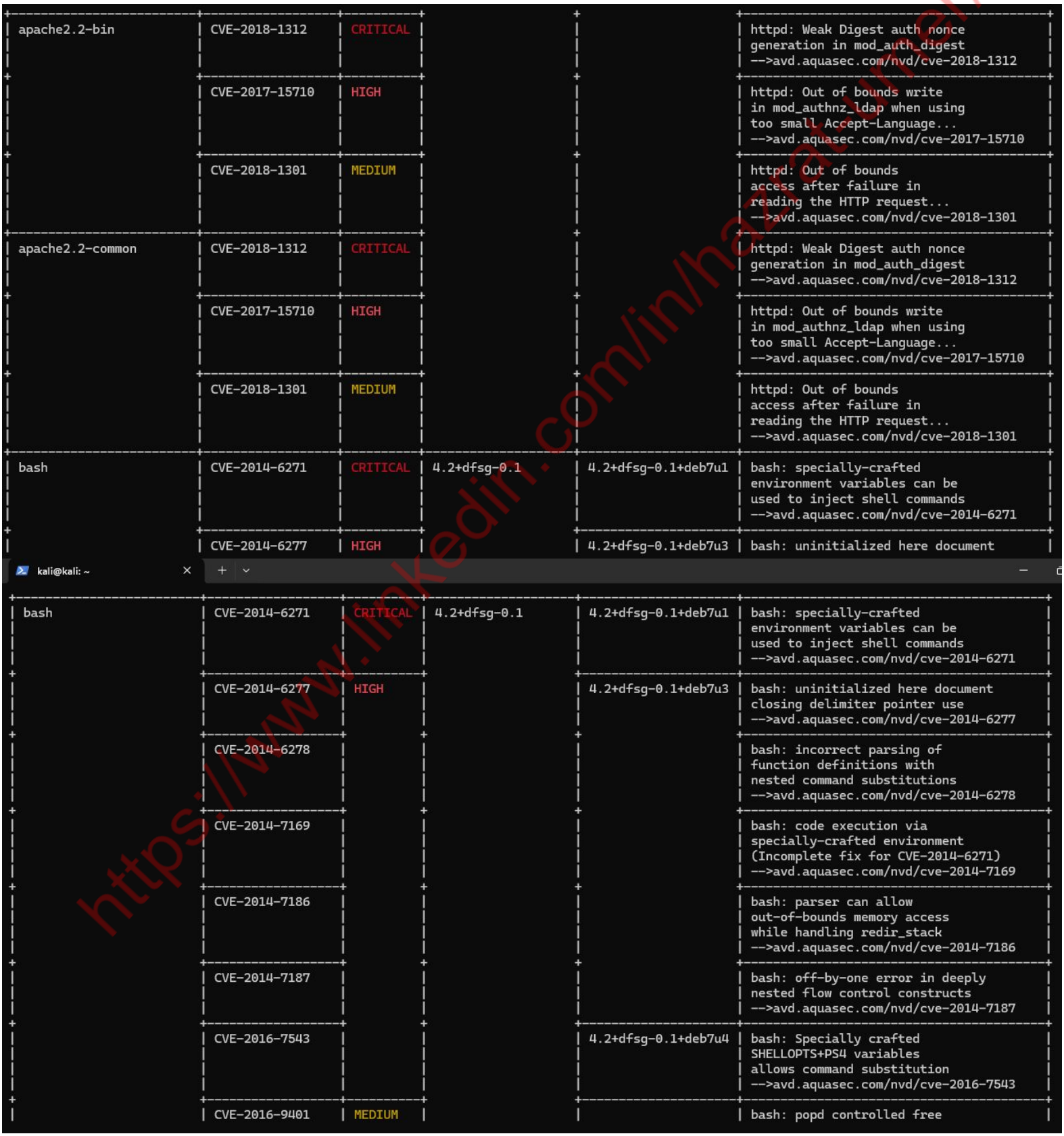
Trivy scan screenshot

```
kali@kali:~$ trivy image vulnerables/cve-2014-6271
2024-08-15T14:20:46.684-0400    WARN    You should avoid using the :latest tag as it is cached. You need to specify '--clear-cache' option when :lat
est image is changed
2024-08-15T14:20:49.271-0400    INFO    Detecting Debian vulnerabilities...
2024-08-15T14:20:49.276-0400    INFO    Trivy skips scanning programming language libraries because no supported file was detected
2024-08-15T14:20:49.276-0400    WARN    This OS version is no longer supported by the distribution: debian 7.11
2024-08-15T14:20:49.276-0400    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
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 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

Trivy scan screenshot



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

kali@kali: ~

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
	CVE-2014-7169				bash: code execution via specially-crafted environment (Incomplete fix for CVE-2014-6271) -->avd.aquasec.com/nvd/cve-2014-7169
	CVE-2014-7186				bash: parser can allow out-of-bounds memory access while handling redir_stack -->avd.aquasec.com/nvd/cve-2014-7186
	CVE-2014-7187				bash: off-by-one error in deeply nested flow control constructs -->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 -->avd.aquasec.com/nvd/cve-2016-7543
	CVE-2016-9401	MEDIUM			bash: popd controlled free

Trivy scan screenshot

bsdutils	CVE-2014-9114	HIGH	2.20.1-5.3		util-linux: command injection flaw in blkid -->avd.aquasec.com/nvd/cve-2014-9114
	CVE-2016-5011	MEDIUM			util-linux: Extended partition loop in MBR partition table leads to DOS -->avd.aquasec.com/nvd/cve-2016-5011
	CVE-2013-0157	LOW			util-linux: mount folder existence information disclosure -->avd.aquasec.com/nvd/cve-2013-0157
coreutils	CVE-2014-9471	HIGH	8.13-3.5		coreutils: memory corruption flaw in parse_datetime() -->avd.aquasec.com/nvd/cve-2014-9471
	CVE-2016-2781	MEDIUM			coreutils: Non-privileged session can escape to the parent session in chroot -->avd.aquasec.com/nvd/cve-2016-2781
gcc-4.7-base	CVE-2014-5044	CRITICAL	4.7.2-5		gcc: integer overflow flaws in libgfortran -->avd.aquasec.com/nvd/cve-2014-5044
	CVE-2002-2439	HIGH			gcc: Integer overflow can occur during the computation of the memory region... -->avd.aquasec.com/nvd/cve-2002-2439
	CVE-2017-11671	MEDIUM			gcc: GCC generates incorrect code for RDRAND/RDSEED intrinsics
gnupg	CVE-2015-1607		1.4.12-7+deb7u9		gnupg2: memcpy with overlapping ranges (keybox_search.c) -->avd.aquasec.com/nvd/cve-2015-1607
gpgv					
inetutils-ping	CVE-2014-3634	HIGH	2:1.9-2		rsyslog: remote syslog PRI vulnerability -->avd.aquasec.com/nvd/cve-2014-3634
libapr1	CVE-2017-12613		1.4.6-3+deb7u1	1.4.6-3+deb7u2	apr: Out-of-bounds array deref in apr_time_exp*() functions -->avd.aquasec.com/nvd/cve-2017-12613
libaprutil1	CVE-2017-12618	MEDIUM	1.4.1-3	1.4.1-3+deb7u1	apr-util: Out-of-bounds access in corrupted SDBM database -->avd.aquasec.com/nvd/cve-2017-12618
libaprutil1-dbd-sqlite3					

Trivy scan screenshot

libaprutil1-ldap					
libblkid1	CVE-2014-9114	HIGH	2.20.1-5.3		util-linux: command injection flaw in blkid -->avd.aquasec.com/nvd/cve-2014-9114
	CVE-2016-5011	MEDIUM			util-linux: Extended partition loop in MBR partition table leads to DOS -->avd.aquasec.com/nvd/cve-2016-5011
	CVE-2013-0157	LOW			util-linux: mount folder existence information disclosure -->avd.aquasec.com/nvd/cve-2013-0157
libbz2-1.0	CVE-2016-3189	MEDIUM	1.0.6-4		bzip2: heap use after free in bzip2recover -->avd.aquasec.com/nvd/cve-2016-3189
libc-bin	CVE-2014-9761	CRITICAL	2.13-38+deb7u12		glibc: Unbounded stack allocation in nan* functions -->avd.aquasec.com/nvd/cve-2014-9761
	CVE-2017-15670				glibc: Buffer overflow in glob with GLOB_TILDE -->avd.aquasec.com/nvd/cve-2017-15670

libc-bin	CVE-2014-9761	CRITICAL	2.13-38+deb7u12		glibc: Unbounded stack allocation in nan* functions -->avd.aquasec.com/nvd/cve-2014-9761
	CVE-2017-15670				glibc: Buffer overflow in glob with GLOB_TILDE -->avd.aquasec.com/nvd/cve-2017-15670
	CVE-2017-15804				glibc: Buffer overflow during unescaping of user names with the ~ operator... -->avd.aquasec.com/nvd/cve-2017-15804
	CVE-2018-6485				glibc: Integer overflow in posix_memalign in memalign functions -->avd.aquasec.com/nvd/cve-2018-6485
	CVE-2015-5180	HIGH			glibc: DNS resolver NULL pointer dereference with crafted record type -->avd.aquasec.com/nvd/cve-2015-5180
	CVE-2016-2856				pt_chown in the glibc package before 2.19-18+deb8u4 on Debian jessie; the elibc... -->avd.aquasec.com/nvd/cve-2016-2856
	CVE-2017-1000408				glibc: Memory leak reachable via LD_HWCAP_MASK -->avd.aquasec.com/nvd/cve-2017-1000408
	CVE-2017-1000409				glibc: Buffer overflow triggerable via LD_LIBRARY_PATH -->avd.aquasec.com/nvd/cve-2017-1000409
	CVE-2017-16997				glibc: Incorrect handling

Trivy scan screenshot

	CVE-2017-16997			glibc: Incorrect handling of RPATH in elf/dl-load.c can be used to execute... -->avd.aquasec.com/nvd/cve-2017-16997
	CVE-2018-1000001			glibc: realpath() buffer underflow when getcwd() returns relative path allows privilege escalation... -->avd.aquasec.com/nvd/cve-2018-1000001
	CVE-2016-10228	MEDIUM		glibc: iconv program can hang when invoked with the -c option -->avd.aquasec.com/nvd/cve-2016-10228
	CVE-2016-4429			glibc: libtirpc: stack (frame) overflow in Sun RPC clntudp_call() -->avd.aquasec.com/nvd/cve-2016-4429
	CVE-2017-12132			glibc: Fragmentation attacks possible when EDNS0 is enabled -->avd.aquasec.com/nvd/cve-2017-12132
	CVE-2017-12133			glibc: Use-after-free read access in clntudp_call in sunrpc -->avd.aquasec.com/nvd/cve-2017-12133
	CVE-2017-15671			glibc: Memory leak in glob with GLOB_TILDE -->avd.aquasec.com/nvd/cve-2017-15671
	CVE-2013-2207	LOW		glibc (pt_chown): Improper pseudotty ownership and permissions changes when granting access to... -->avd.aquasec.com/nvd/cve-2013-2207
libc6	CVE-2014-9761	CRITICAL		glibc: Unbounded stack allocation in nan* functions -->avd.aquasec.com/nvd/cve-2014-9761
	CVE-2017-15670			glibc: Buffer overflow in glob with GLOB_TILDE -->avd.aquasec.com/nvd/cve-2017-15670
	CVE-2017-15804			glibc: Buffer overflow during unescaping of user names with the ~ operator... -->avd.aquasec.com/nvd/cve-2017-15804
	CVE-2018-6485			glibc: Integer overflow in posix_memalign in memalign functions -->avd.aquasec.com/nvd/cve-2018-6485
	CVE-2015-5180	HIGH		glibc: DNS resolver NULL pointer dereference with crafted record type -->avd.aquasec.com/nvd/cve-2015-5180
	CVE-2016-2856			pt_chown in the glibc package before 2.19-18+deb8u4 on Debian jessie; the elibc... -->avd.aquasec.com/nvd/cve-2016-2856
	CVE-2017-1000408			glibc: Memory leak reachable via LD_HWCAP_MASK -->avd.aquasec.com/nvd/cve-2017-1000408
	CVE-2017-1000409			glibc: Buffer overflow triggerable via LD_LIBRARY_PATH -->avd.aquasec.com/nvd/cve-2017-1000409
	CVE-2017-16997			glibc: Incorrect handling

Trivy scan screenshot

libssl1.0.0	CVE-2017-3735	MEDIUM	1.0.1t-1+deb7u2	1.0.1t-1+deb7u3	openssl: Malformed X.509 IPAdressFamily could cause OOB read -->avd.aquasec.com/nvd/cve-2017-3735
	CVE-2018-0737				openssl: RSA key generation cache timing vulnerability in crypto/rsa/rsa_gen.c allows attackers to... -->avd.aquasec.com/nvd/cve-2018-0737
	CVE-2018-0739			1.0.1t-1+deb7u4	openssl: Handling of crafted recursive ASN.1 structures can cause a stack overflow... -->avd.aquasec.com/nvd/cve-2018-0739
	CVE-2014-3566	LOW			SSL/TLS: Padding Oracle On Downgraded Legacy Encryption attack -->avd.aquasec.com/nvd/cve-2014-3566

libssl1.0.0	CVE-2017-3735	MEDIUM	1.0.1t-1+deb7u2	1.0.1t-1+deb7u3	openssl: Malformed X.509 IPAdressFamily could cause OOB read -->avd.aquasec.com/nvd/cve-2017-3735
	CVE-2018-0737				openssl: RSA key generation cache timing vulnerability in crypto/rsa/rsa_gen.c allows attackers to... -->avd.aquasec.com/nvd/cve-2018-0737
	CVE-2018-0739			1.0.1t-1+deb7u4	openssl: Handling of crafted recursive ASN.1 structures can cause a stack overflow... -->avd.aquasec.com/nvd/cve-2018-0739
	CVE-2014-3566	LOW			SSL/TLS: Padding Oracle On Downgraded Legacy Encryption attack -->avd.aquasec.com/nvd/cve-2014-3566

libstdc++6	CVE-2014-5044	CRITICAL	4.7.2-5		gcc: integer overflow flaws in libgfortran -->avd.aquasec.com/nvd/cve-2014-5044
	CVE-2002-2439	HIGH			gcc: Integer overflow can occur during the computation of the memory region... -->avd.aquasec.com/nvd/cve-2002-2439
	CVE-2017-11671	MEDIUM			gcc: GCC generates incorrect code for RDRAND/RDSEED intrinsics -->avd.aquasec.com/nvd/cve-2017-11671

passwd	CVE-2017-12424	CRITICAL	1:4.1.5.1-1+deb7u1		shadow-utils: Buffer overflow via newusers tool -->avd.aquasec.com/nvd/cve-2017-12424
	CVE-2018-7169	MEDIUM			shadow-utils: newgidmap allows unprivileged user to drop supplementary groups potentially allowing privilege... -->avd.aquasec.com/nvd/cve-2018-7169
perl	CVE-2018-6797	CRITICAL	5.14.2-21+deb7u5		perl: heap write overflow in regcomp.c -->avd.aquasec.com/nvd/cve-2018-6797
	CVE-2018-6913			5.14.2-21+deb7u6	perl: heap buffer overflow in pp_pack.c -->avd.aquasec.com/nvd/cve-2018-6913

Report to Fix Container Issues

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

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
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
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

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 Object Level Authorization
Description
Broken Object Level Authorization happens when an Application Programming Interface (API) does not properly enforce access controls for individual objects. In this vulnerability, the threat actor could access other users' data by manipulating (changing) user IDs or order IDs in API requests.
Risk
If the Broken Object Level Authorization vulnerability is not addressed properly this could allow unauthorized access to sensitive information of other users. This vulnerability could lead to severe privacy breaches.
Remediation
Access control checks should be implemented e.g. Every request should be validated against the user's permissions. *Use complex object IDs and make it difficult to guess. e.g. Instead of using sequential IDs use UUIDs or use complex identifiers that are difficult to predict by attackers or for brute force attacks* *Regularly review access control policies. .*

API Vulnerabilities and remediation

2. Excessive Data Exposure
Description
When API sends more data than the required data in its response. For Example: Instead of sending the required data such as username, if an API endpoint sends a full user profile containing sensitive information then this is called Excessive Data Exposure vulnerability.
Risk
Excessive Data Exposure vulnerability increases the attack surface for hackers and the risk of data breaches. The attacker can easily sniff the traffic and can see the sensitive information. Through which sensitive information could be leaked. If sensitive or private data is exposed to unauthorized individuals then it could cause the violation of different privacy regulations such as the General Data Protection Regulation (GDPR)and the California Consumer Privacy Act (CCPA).
Remediation
Sensitive data should be filtered on the server side. Sanitize all the API response data. *API responses should be carefully reviewed to make sure they contain only legitimate data.* *Avoid using generic methods such as to_json() and to_string().*

API Vulnerabilities and remediation

3. Lack of Resources & Rate Limiting
Description
An API on which rate limiting and throttling controls are not implemented is vulnerable to Denial of Service (Dos) attacks and brute force attacks. *API requests consume different resources such as CPU, Network, and storage. Multiple API client requests compete for resources. API is vulnerable if limits such as execution timeouts, max allocable memory, number of processes, request payload size, number of requests per client/resource, and number of records per page to return in a single response are missing or set inappropriately.*
Risk
Due to a lack of rate limiting, the API can be overwhelmed and will cause to disrupt service availability which will result in data loss.
Remediation
Implement Rate and throttling and monitor API usage *Add proper server-side validation for query string* *There should be a limit that how often a client can call an API within a specified amount of time.*

References

https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization/

https://owasp.org/API-Security/editions/2019/en/0xa3-excessive-data-exposure/

https://owasp.org/API-Security/editions/2019/en/0xa4-lack-of-resources-and-rate-limiting/

https://vulcan.io/blog/secure-sdlc-best-practices/

https://portswigger.net/web-security/cross-site-scripting#:~:text=Cross%2Dsite%20scripting%20(also%20known%20as%20XSS)%20is,to%20segregate%20different%20websites%20from%20each%20other.