Vulnerability Assessment & Penetration Testing

Report prepared for :

Report issued: 31/10/2025 **Submitted by : Amarnadh S**

Sensitive: The information in this document is strictly confidential and is made by Amarnadh S

Confidentiality Notice

This report contains sensitive, privileged, and confidential information. Unauthorized access, disclosure, distribution, or reproduction of this document or its contents is strictly prohibited. The information is intended solely for authorized personnel. Proper precautions should be taken to safeguard the confidentiality of the data contained herein.

Disclaimer

This assessment was conducted as a “point-in-time” evaluation of the web application(s) in scope. While every effort has been made to identify vulnerabilities, this report may not disclose all existing security issues. Vulnerabilities identified in this report are based on the environment and data provided during the assessment period. Any changes to the systems or environment may affect the validity of the findings

Table of contents

1. Abstract

Web applications serve as critical gateways to modern services, but their widespread use also makes them attractive targets for cyber attacks. This project focuses on conducting a thorough security assessment of various web applications to identify, evaluate, and mitigate vulnerabilities that could threaten their confidentiality, integrity, and availability. The assessment aims to enhance the security posture of these applications by providing actionable recommendations to address discovered weaknesses effectively.

The project follows a systematic methodology, beginning with reconnaissance to gather information about the applications and identify potential entry points. This is followed by vulnerability discovery, using both manual testing and tools like SQLmap for SQL injection detection and Dirsearch for directory enumeration. These methods uncovered vulnerabilities such as SQL injection, exposed admin credentials, insecure cookie configurations, path traversal, and publicly accessible logs.

Critical vulnerabilities identified include SQL injection and exposed admin passwords, which pose significant risks of unauthorized access, data breaches, and system compromise. High-risk vulnerabilities, such as path traversal and accessible log files, could provide attackers with sensitive information for launching targeted attacks. Medium-risk issues, like missing secure flags on cookies, expose the applications to risks such as session hijacking.

To address these issues, the project provides a range of mitigation strategies. These include implementing prepared statements and parameterized queries to prevent SQL injection, enforcing strong password policies and multi-factor authentication for admin accounts, enabling secure flags for cookies, and restricting access to sensitive directories and files. Additional recommendations include encrypting logs, deploying web application firewalls, and performing regular security audits to ensure ongoing protection.

This project highlights the importance of proactive vulnerability assessment and management in securing web applications against evolving threats. By identifying and addressing weaknesses, organizations can safeguard their digital assets, protect user data, and ensure reliable service delivery. The findings and recommendations provide practical insights for developers, administrators, and security professionals, emphasizing best practices for maintaining secure and resilient web applications.

2. Acknowledgment

I would like to express my heartfelt gratitude to everyone who supported me throughout the course of this project. First and foremost, I extend my sincere thanks to my mentors and instructors, whose expertise and guidance were instrumental in shaping the direction and execution of this work. Your insights and encouragement have been invaluable. I am also deeply grateful to the developers and maintainers of the tools and frameworks utilized in this project, such as SQLmap and Dirsearch, whose contributions to the open-source community made this work possible. The comprehensive documentation and resources accompanying these tools greatly facilitated the assessment process. Special thanks go to the cyber security community for sharing best practices, guides, and tutorials, which provided essential context and direction for the project. I would also like to acknowledge my peers for their constructive feedback and collaborative spirit, which motivated me to refine and improve my work. Lastly, I am profoundly thankful for the unwavering support and encouragement of my family and friends. Their belief in my abilities provided me with the determination and confidence to overcome challenges and successfully complete this project. This project would not have been possible without the collective contributions and support of these individuals and communities, and I am deeply appreciative of their efforts.

3.Introduction

Web applications are an integral part of the modern digital ecosystem, enabling businesses, organizations, and individuals to interact with their users seamlessly. However, as their importance grows, so do the risks associated with their security. Cyberattacks targeting web applications are becoming increasingly sophisticated, with attackers exploiting vulnerabilities to gain unauthorized access, compromise sensitive data, or disrupt operations. The consequences of such attacks can be severe, ranging from financial loss and reputational damage to legal repercussions.This project aims to address these challenges by conducting a comprehensive security assessment of several web applications. The primary objective is to identify vulnerabilities that could compromise the confidentiality, integrity, and availability of these applications and their data. By uncovering weaknesses and proposing actionable remediation strategies, this project seeks to enhance the overall security posture of the targeted web applications. The assessment is conducted using a systematic approach that begins with reconnaissance to identify potential entry points and gather information about the target applications. This is followed by vulnerability testing using tools such as SQLmap for SQL injection detection and Dirsearch for directory enumeration. The testing is complemented by manual validation to ensure accuracy and relevance. Vulnerabilities discovered during the assessment are categorized based on their severity, using established frameworks like the Common Vulnerability Scoring System (CVSS), to prioritize remediation efforts effectively. The findings of this project highlight several critical and high-risk vulnerabilities, including SQL injection, exposed admin credentials, insecure cookie configurations, and publicly accessible logs. These vulnerabilities represent significant threats to the applications' security and functionality. By addressing these issues promptly, organizations can mitigate the risks of exploitation and strengthen their defenses against future attacks. This project highlights the need for proactive security assessments in today's threat landscape. By identifying and fixing vulnerabilities early, organizations can strengthen their web applications and protect users, data, and reputation.

4. Literature Overview

The foundation of this project is based on a strong body of literature and resources focused on web application security. A primary reference is OWASP (Open Web Application Security Project), which provides valuable guidance on identifying and mitigating common vulnerabilities such as SQL injection, cross- site scripting (XSS), and session management flaws. OWASP's vulnerability classification and mitigation strategies were key in shaping the testing approach used in this project, ensuring a systematic and effective process for addressing security risks. In addition to OWASP, research on practical tools like Burp Suite, SQLmap, and Dirsearch played a critical role in this project. SQLmap, with its ability to automate SQL injection testing, was particularly useful for identifying vulnerabilities quickly and accurately. Furthermore, articles and case studies on secure cookie handling and directory traversal attacks provided insights into best practices for configuring web applications securely and minimizing risks. To prioritize vulnerabilities, the Common Vulnerability Scoring System (CVSS) was used to assess the potential impact of each issue, helping to address the most critical problems first. The combination of these resources and methodologies enabled a structured, effective approach to web application security, ensuring that vulnerabilities were identified, assessed, and mitigated in a timely manner.

5. Technology Overview

Burp suite Professional

A comprehensive platform for web application security testing, used for intercepting, analyzing, and manipulating web traffic to identify vulnerabilities such as cross-site scripting and session management flaws.

Nuclei

Nuclei is an open-source scanning engine. Nuclie can be used for checking and detecting vulnerabilities , security misconfigurations , etc.

Dirsearch

A directory enumeration tool used to discover hidden directories and files on a web server. It is particularly useful for identifying access to sensitive areas such as admin panels and log directories.

CVSS (Common Vulnerability Scoring System)

A standardized framework for assessing the severity and impact of security vulnerabilities, ensuring consistent prioritization for remediation.

OWASP Resources

Guidelines, cheat sheets, and vulnerability classifications from OWASP were essential in understanding and mitigating web application security risks.

These technologies and resources collectively provide the tools and methodologies necessary for conducting a thorough and effective security assessment.

6. Executive Summery

The purpose of this assessment was to conduct a comprehensive security evaluation of the identified web applications to uncover potential vulnerabilities and assess their impact. The evaluation aimed to ensure the confidentiality, integrity, and availability of the applications and their associated data.

Key Findings :

The assessment identified 4 vulnerabilities , categorized by severity

Severity	No of Vulnerabilities
Critical	1
High	1
Medium	2
Low	0

The critical vulnerabilities pose significant risks, including unauthorized access to sensitive data, website defacement, or compromise of the application's infrastructure.

Recommendations:

It is imperative to address the identified vulnerabilities promptly. Implementing the recommended remedies will mitigate potential risks and enhance the security posture of the systems.

7. Scope

Security assessment includes testing for security loopholes in the scope defined below. Apart from the following, no other information was provided. Nothing was assumed at the start of the security assessment. The following was the scope covered under the security audit:

Web Application 1: [redacted]

Web Application 2: [redacted]

Web Application 3: [redacted]

Web Application 4: [redacted]

8. Testing Methodology

This assessment was carried out using a systematic and industry-standard approach to ensure comprehensive identification of security vulnerabilities. The methodology consisted of the following phases:

Phase 1: Reconnaissance

During the reconnaissance phase, information was collected about the web applications to identify potential entry points for attacks. Techniques included:

- DNS and WHOIS lookups.
- Subdomain enumeration.
- Crawling the web applications to map exposed endpoints.

Phase 2: Target Assessment

In this phase, the applications were tested for vulnerabilities using both manual and automated approaches. Common vulnerabilities targeted included:

- Cross-Site Scripting (XSS)
- SQL Injection.
- Session Management Flaws.
- Rate Limiting Issues.

Phase 3: vulnerability Discovery

All findings were validated and prioritized based on their risk levels. A combination of automated tools and manual validation ensured accuracy.Controlled exploitation was performed to prove the existence of vulnerabilities without causing disruptions.

9. Risk Classification

The vulnerabilities identified during the assessment were prioritized based on their severity using the CVSS (Common Vulnerability Scoring System). This classification ensures consistency in evaluating the potential impact and urgency of the vulnerabilities.

Critical (CVSS Score: 9.0 – 10.0)

Critical vulnerabilities represent the highest level of risk, often leading t catastrophic consequences if exploited. Immediate action is required to mitigate these risks.

High (CVSS Score: 7.0 – 8.9)

High-risk vulnerabilities are exploitable with relative ease and can cause significant damage to the application or its data. They should be addressed Promptly.

Medium (CVSS Score: 4.0 – 6.9)

Medium-risk vulnerabilities require specific conditions for exploitation and pose a moderate threat. Resolving these issues should be part of a planned security strategy.

Low (CVSS Score: 0.1 – 3.9)

Low-risk vulnerabilities have limited impact but still pose potential security concerns. They can be addressed during routine maintenance.

This classification framework ensures that vulnerabilities are addressed based on their potential impact and likelihood of exploitation.

Range	Severity Category
0.0	None
0.1 – 3.9	Low
4.0 – 6.9	Medium
7.0 – 8.9	High
9.0 – 10.0	Critical

CVSS Scores

10. Assessment Findings

No.	Findings	CVSS	Severity
1	Session Hijacking	9.9	Critical
2	OTP Bypass	8.5	High
3	Reflected XSS	4.9	Medium
4	DNS Misconfiguration	6.5	Medium

Vulnerability #1: SESSION HIJACKING

Severity : Critical (9.9)

Vulnerable URL : [redacted]

Security Impact : Severe

Session hijacking is when an attacker takes over a valid user session (session cookie, token, or session identifier) to impersonate that user without needing their credentials.

Realistic example scenarios

- User session cookie stolen via XSS → attacker transfers money fro Account.
- Session ID sniffed on open Wi-Fi (no TLS) → attacker logs into webmail and exfiltrates sensitive attachments.
- Session fixation → attacker sets a session ID then convinces victim to log in; attacker then reuses that session.

Main Security Impacts :

1. Confidentiality

- Data exposure — attacker can read sensitive user data (personal info, financials, messages, files).
- Mass exposure risk — if a hijacked session belongs to an admin or service account, very large amounts of data can be accessed.

2. Integrity

- Unauthorized changes — attacker can modify data, change settings (passwords, email), or inject malicious content (e.g., alter invoices).
- Supply-chain/transaction tampering — attacker can alter orders, billing details, or deployment settings.

3. Availability

- Account lockout/denial — attacker can log users out, delete resources, or perform destructive actions that disrupt service.
- Operational disruption — cleanup and recovery consume resources and may require downtime.

4. Authorization & Privilege escalation

- Lateral movement — a hijacked low-privilege session can be used to get higher privileges or pivot to other systems.
- Impersonation of privileged users — immediate high-impact access if session belongs to admin/ops.

5. Financial & legal/regulatory impact

- Direct fraud or theft — money transfers, fraudulent purchases.
- Regulatory fines & compliance breaches — GDPR, PCI-DSS, HIPAA violations if personal or payment data were exposed.
- Remediation costs — investigation, forensic analysis, notifications, Litigation.

6. Reputation & customer trust

- User churn — customers may leave after a breach.
- Brand damage — public breaches harm trust and sales.

Mitigations :

- Secure cookie attributes: Secure, HttpOnly, SameSite.
- Short session lifetimes (and optional sliding expiration) for sessions.
- Rotate/renew tokens on authentication state changes (password change, logout).
- Require re-authentication for sensitive actions (payments, account changes).
- Enforce MFA for logins and for privilege-escalating operations.
- Avoid sending tokens in URLs or logging them.
- Protect against XSS (CSP, input validation/escaping, output encoding).
- Use unguessable token generation and consider server-side revocation for JWTs (keep a revocation list or use short JWT lifetimes plus refresh tokens).
- Device/IP binding or token binding heuristics where appropriate (with care for legitimate mobility).
- Least privilege for accounts and short-lived machine credentials.

Steps to Reproduce :

Go to the target URL [redacted]

Create an account

Inspect the page

Go to storage > Cookies > then copy the session id

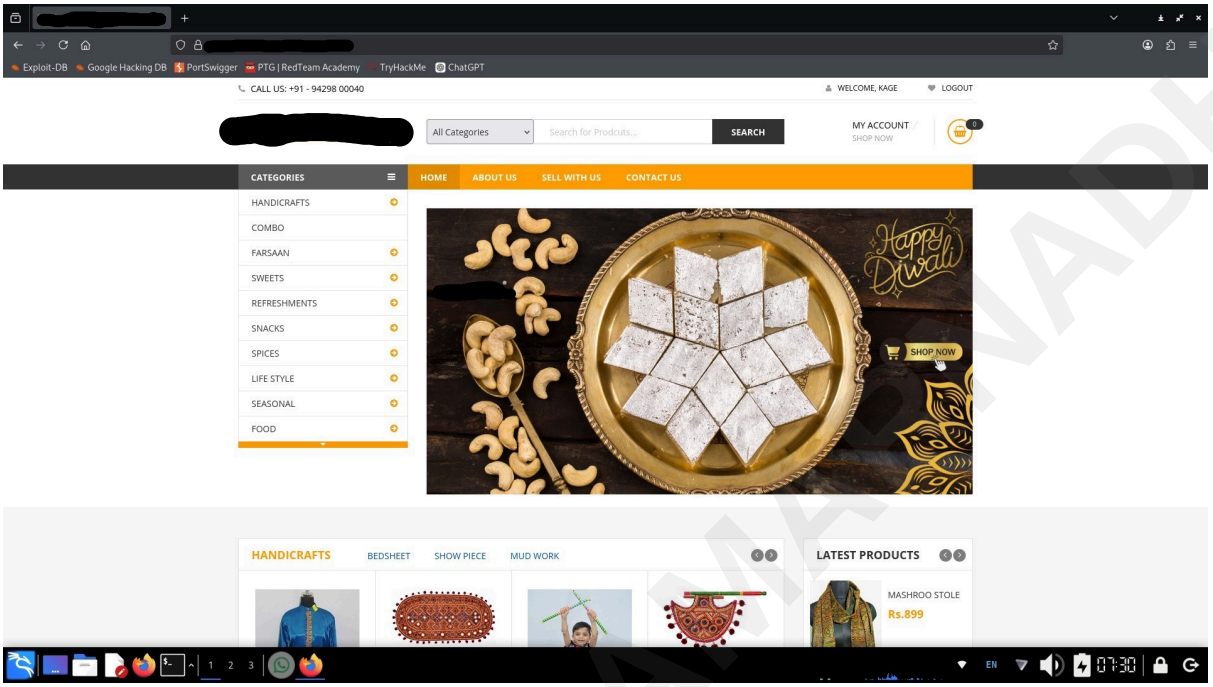
now open the URL in another browser

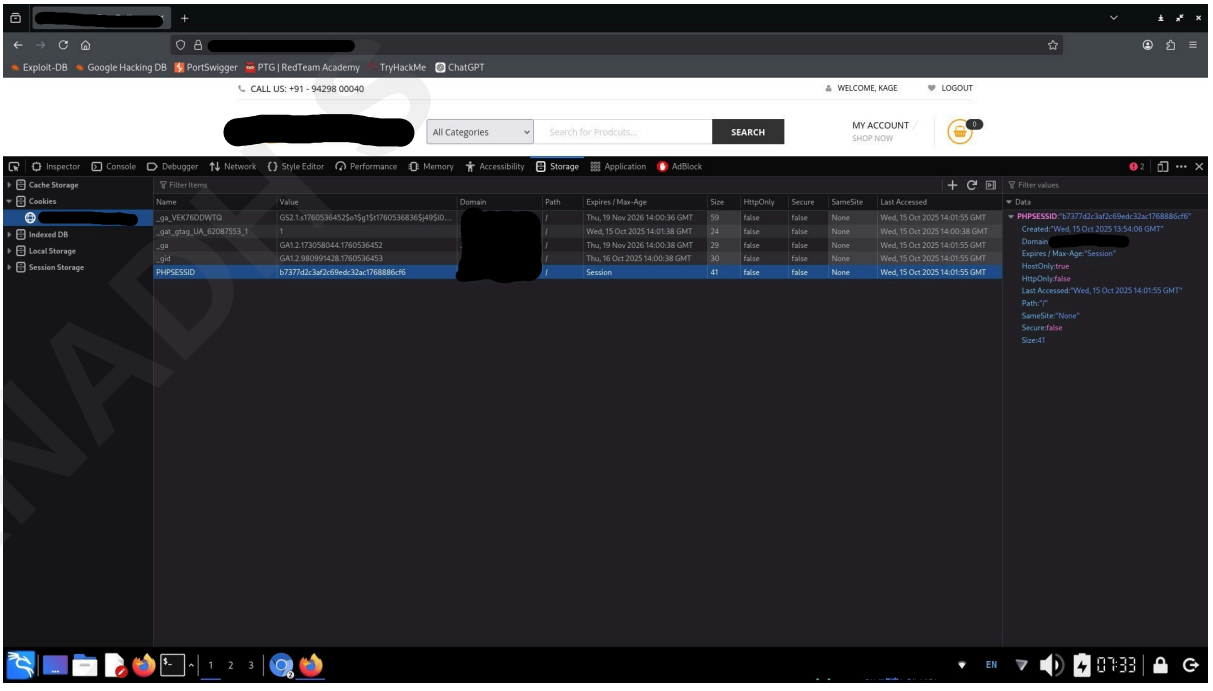
Inspect the page

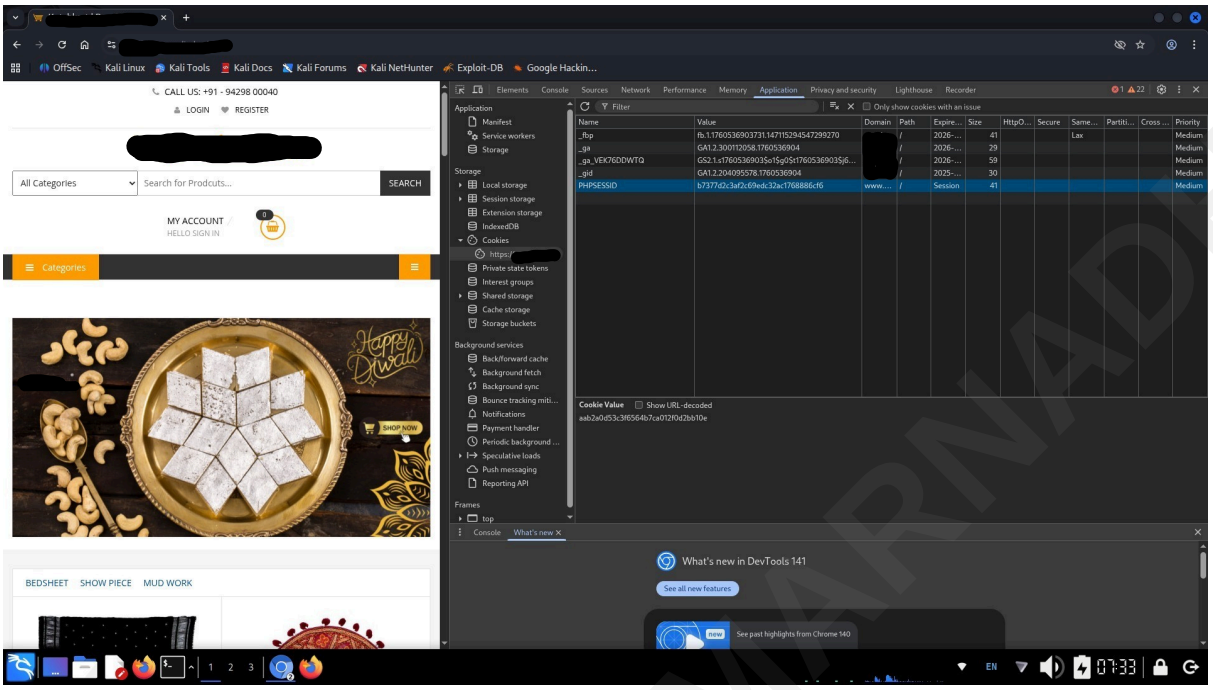
Go to Application > paste the session id in the PHPSESSID

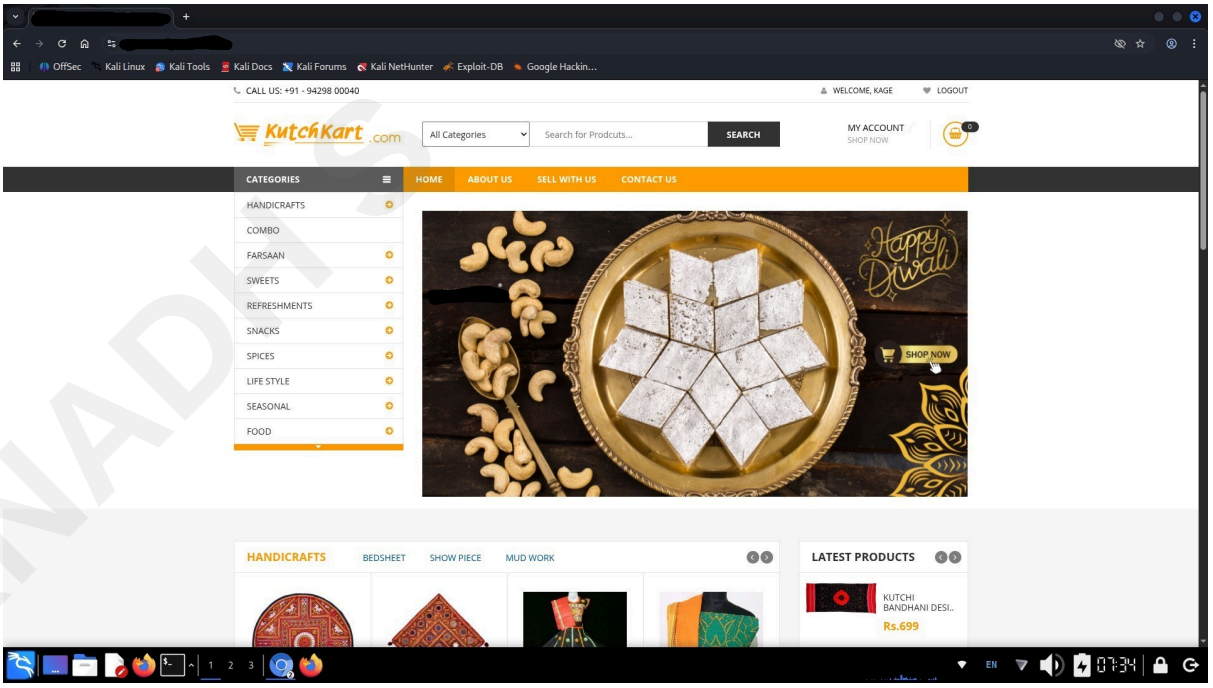
Now refresh the page you will be logged into the targets account

Screenshots






CALL US: +91 - 94298 00040
LOGIN
REGISTER
All Categories
Search for Prodcuts...
SEARCH
MY ACCOUNT
HELLO SIGN IN
Categories
Happy Diwali
SHOP NOW
BEDSHEET
SHOW PIECE
MUD WORK
What's new in DevTools 141
See all new features


CALL US: +91 - 94298 00040
WELCOME, KAGE
LOGOUT
KutchKart .com
All Categories
Search for Prodcuts...
SEARCH
MY ACCOUNT
SHOP NOW
CATEGORIES
HOME
ABOUT US
SELL WITH US
CONTACT US
HANDICRAFTS
COMBO
FARSAAN
SWEETS
REFRESHMENTS
SNACKS
SPICES
LIFE STYLE
SEASONAL
FOOD
Happy Diwali
SHOP NOW
HANDICRAFTS
BEDSHEET
SHOW PIECE
MUD WORK
LATEST PRODUCTS
KUTCHI BANDHANI DESI..
Rs.699

Recommendations :

1. Transport Layer Security (TLS)

- Enforce HTTPS across all endpoints.
- Use HSTS (HTTP Strict Transport Security) to prevent protocol downgrades.
- Redirect all HTTP requests to HTTPS automatically.
- Renew and monitor TLS certificates regularly.
  Without TLS, cookies can be intercepted on public networks.

2. Session Management Best Practices

- Regenerate session IDs after login, privilege changes, or MFAvalidation.
- Expire sessions after inactivity or fixed lifetime (e.g., 15–30 mins idle, 8–12 hrs max life).
- Invalidate sessions on logout and password reset.
- Use short-lived, opaque session IDs, not predictable or base64-encoded values.
- Store sensitive state server-side — never inside the cookie itself (JWTs should be short-lived if used).

3. Authentication & Access Controls

- Enable Multi-Factor Authentication (MFA) — even if cookies are stolen, MFA helps prevent full takeover.
- Re-authenticate users before sensitive actions (payments, password change, account deletion).
- Implement role-based access control (RBAC) and least privilege for accounts.
- Limit concurrent sessions where feasible (optional, but useful for high- risk systems).

4. Web Application Defenses

- Prevent XSS attacks, which are the #1 cause of cookie theft:
- Use frameworks with automatic output encoding.
- Apply Content Security Policy (CSP) headers.
- Validate and sanitize all user inputs.

- Avoid inline JavaScript when possible.
- Prevent CSRF attacks:
- Use CSRF tokens or SameSite cookies.
- Validate origin and referer headers for sensitive requests.

5. Monitoring & Detection

- Log all session creation, usage, and termination events.
- Alert on:
- Same session ID used from different IPs or countries.
- Abnormal login patterns (rapid logins/logouts).
- High-risk operations from unfamiliar devices or IPs.
- Use device fingerprinting or geo-based anomaly detection for added Defense.

6. Infrastructure & Code Hygiene

- Never expose session IDs in:
- URLs
- Logs
- Error messages
- Referrer headers
- Limit cookie visibility to the minimum domain/path.
- Avoid storing tokens in localStorage or sessionStorage
- Consider token binding or device-based checks if supported.

7. Incident Response (if a hijack occurs)

1. Force logout for affected users (invalidate sessions).
2. Rotate session keys and revoke tokens.
3. Require password resets and MFA re-validation.
4. Review logs to identify attack vectors (e.g., XSS, misconfigurations).
5. Patch vulnerabilities and verify new sessions are secure.
6. Notify users and regulators if personal data exposure occurred.

8. Organizational Control

- Conduct regular security audits and penetration testing.
- Educate developers on secure cookie practices.
- Use automated scanners (e.g., OWASP ZAP, Burp Suite) to detect insecure cookie settings.
- Keep frameworks, libraries, and dependencies up to date

Reference :

https://owasp.org/www-community/attacks/Session_hijacking_attack

https://en.wikipedia.org/wiki/Session_hijacking

https://www.barracuda.com/support/glossary/session-hijacking

https://www.imperva.com/learn/application-security/session-hijacking/

https://www.geeksforgeeks.org/ethical-hacking/session-hijacking/

Vulnerability #2 : OTP BYPASS

Severity : High (8.5)

Vulnerable URL [redacted]

Security Impact : High

OTP Bypass is a type of cyberattack where an unauthorized individual or malicious entity defeats or circumvents the requirement to submit a valid One-Time Password (OTP) in order to gain access to a user's account or perform a sensitive transaction.

Impact :

Financial Fraud and Loss: Attackers can gain access to banking, e-commerce, or payment accounts, leading to unauthorized transactions, fund transfers, and financial loss for the victim.

Data Breach and Identity Theft: Compromised accounts—especially those containing Personally Identifiable Information (PII)—can result in the theft of sensitive data, such as personal details, payment information, and medical records, which facilitates identity theft and other malicious activities.

Reputational Damage and Loss of Trust: For businesses, successful bypass attacks erode customer trust in their security measures, leading to reputational harm, loss of business, and potentially regulatory fines.

Full System Access (in organizational contexts): If an employee's account is compromised, the attacker may gain a foothold within an organization's network, enabling them to escalate privileges, move laterally, or pivot to other applications and infrastructure.

Account Lockout: Attackers may lock the legitimate user out of their own account after a takeover

Mitigations :

1. Robust Server-Side Validation

The server must be the sole authority for validating the OTP and managing the user's session.

- Validate on Server Only: Ensure that OTP verification is never performed on the client side (e.g., in JavaScript) and that the server-side logic is robust.

- Prevent Response Manipulation: After an OTP validation, the server must independently check the entire session state and not rely on a simple client-controlled parameter (like success: true) to grant access.

- Single-Use and Short Expiry:

- An OTP must be invalidated and deleted from the system immediately after its first successful use.

- Set a very short expiry time (e.g., 2–5 minutes) to minimize the window for interception and use.

2. Rate Limiting and Anti-Brute Force

This prevents attackers from guessing the OTP through automated, high-volume attempts.

- Limit Attempts: Enforce strict limits on the number of failed OTP submission attempts for a user/account (e.g., 3–5 attempts). After the limit is reached, the account should be locked out or require a separate, high- assurance recovery process.

- Limit Generation: Restrict how often a user can request a new OTP (resend limit) to prevent abuse and denial-of-service (DoS) via SMS pumping. Invalidate the previous OTP immediately upon generating a new one.

- Use CAPTCHA/Throttling: Implement a CAPTCHA or a time-delay (exponential backoff) after a few failed attempts.

3. Secure OTP Generation and Transmission

- Strong Algorithm: Use a cryptographically secure pseudo-random number generator (CSPRNG) to create unpredictable OTPs. Avoid sequential or easily guessable patterns.

- Use Secure Channels: For high-risk transactions, prioritize more secure methods than SMS, such as Time-based One-Time Passwords (TOTP) via dedicated authenticator apps (e.g., Google or Microsoft Authenticator) or hardware tokens.

- Use HTTPS/TLS: Ensure all network traffic is encrypted using HTTPS/TLS to prevent eavesdropping during transmission.

4. User Education and Transparency

Human factors are often the weakest link in the security chain.

- User Training: Educate users to never share their OTPs with anyone, even a person claiming to be from customer support.

- Informative Messaging: Include a clear warning in the OTP message/email, for example: "Your code is 123456. DO NOT SHARE THIS CODE with anyone, even us."

- Monitor for Social Engineering: Implement additional steps or warnings if an OTP request is coming from an unusual location or device to help mitigate social engineering attacks like vishing or SIM swapping.

Steps to Reproduce :

Go to target URL : [redacted]

Give some fake credentials and click GET OTP

Open Burpsuite and go to proxy and on intercept

Now give a random OTP number and click verify

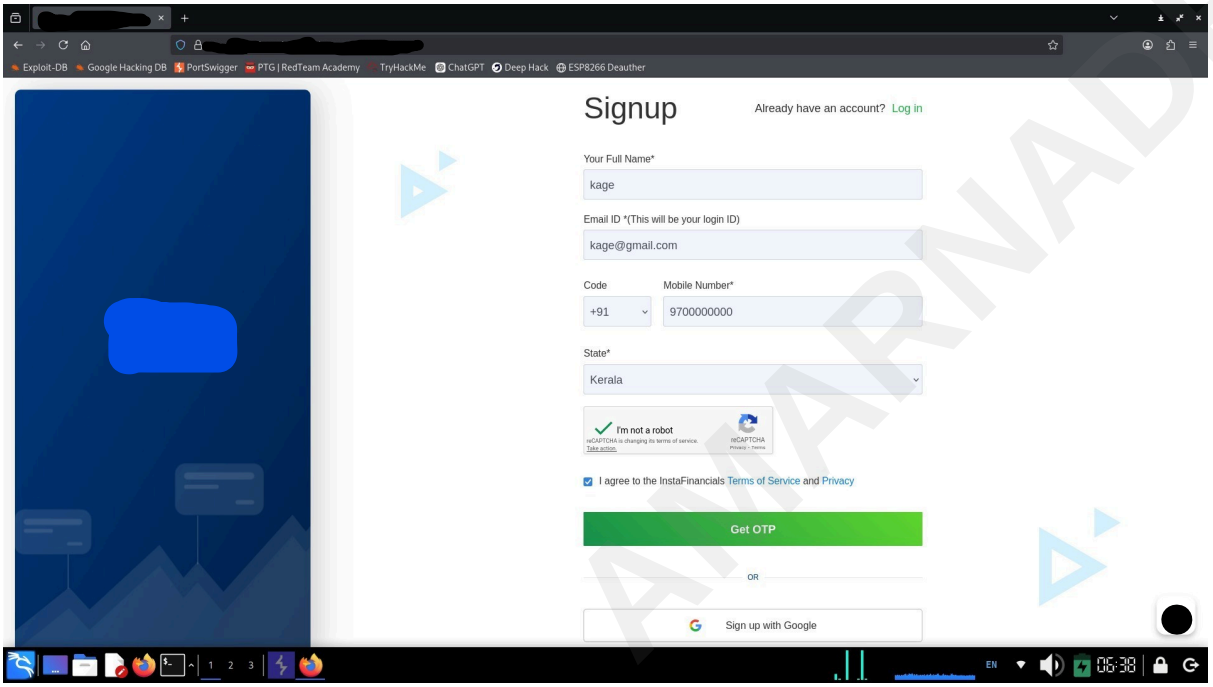
Now intercept the packet in burpsuite and alter the response

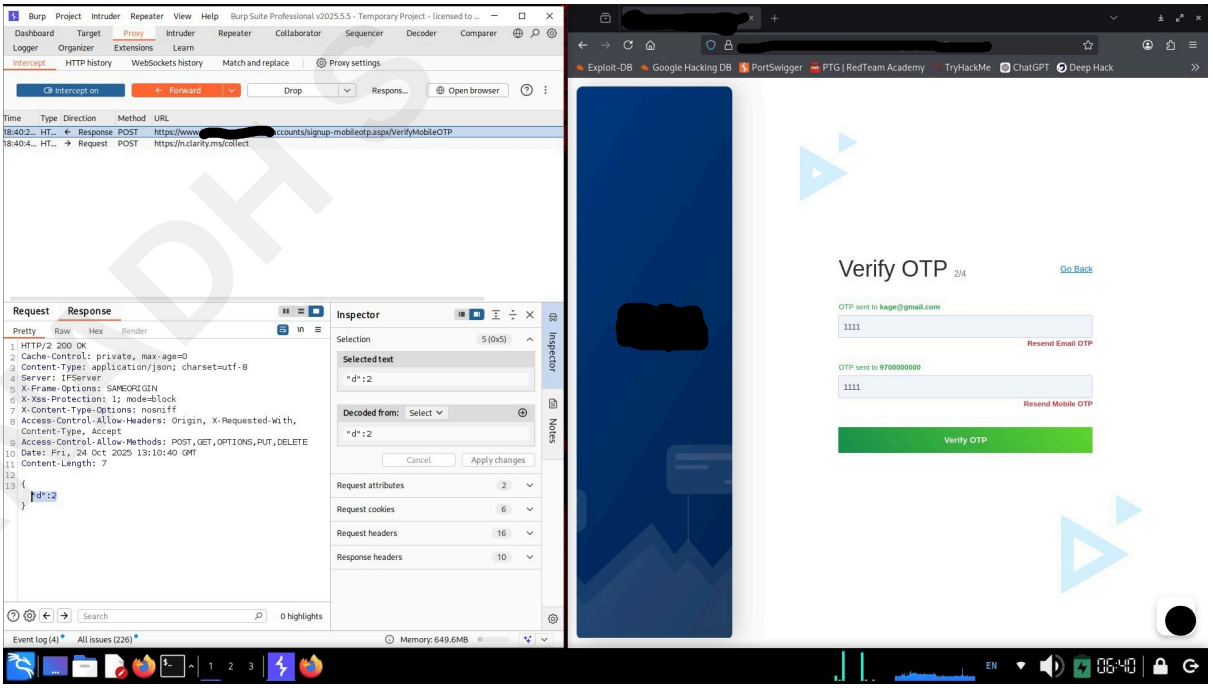
The response will show “d”:2 which means false change it to “d”:1

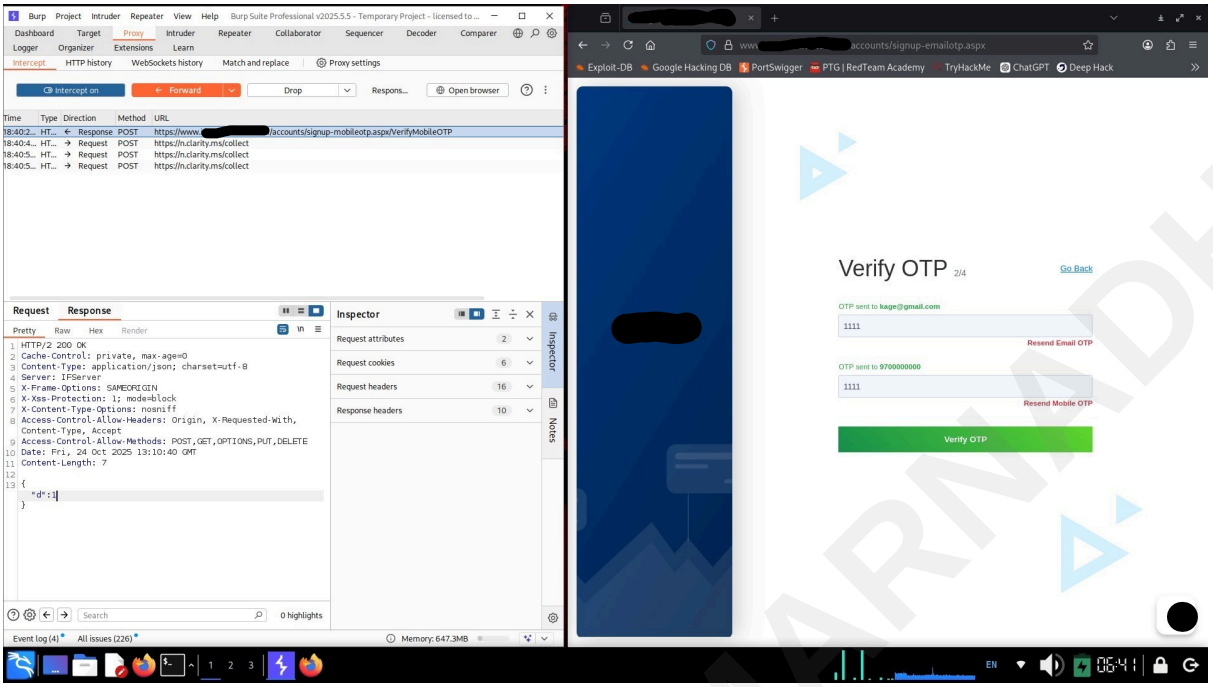
Now froward the response and OTP will be verified

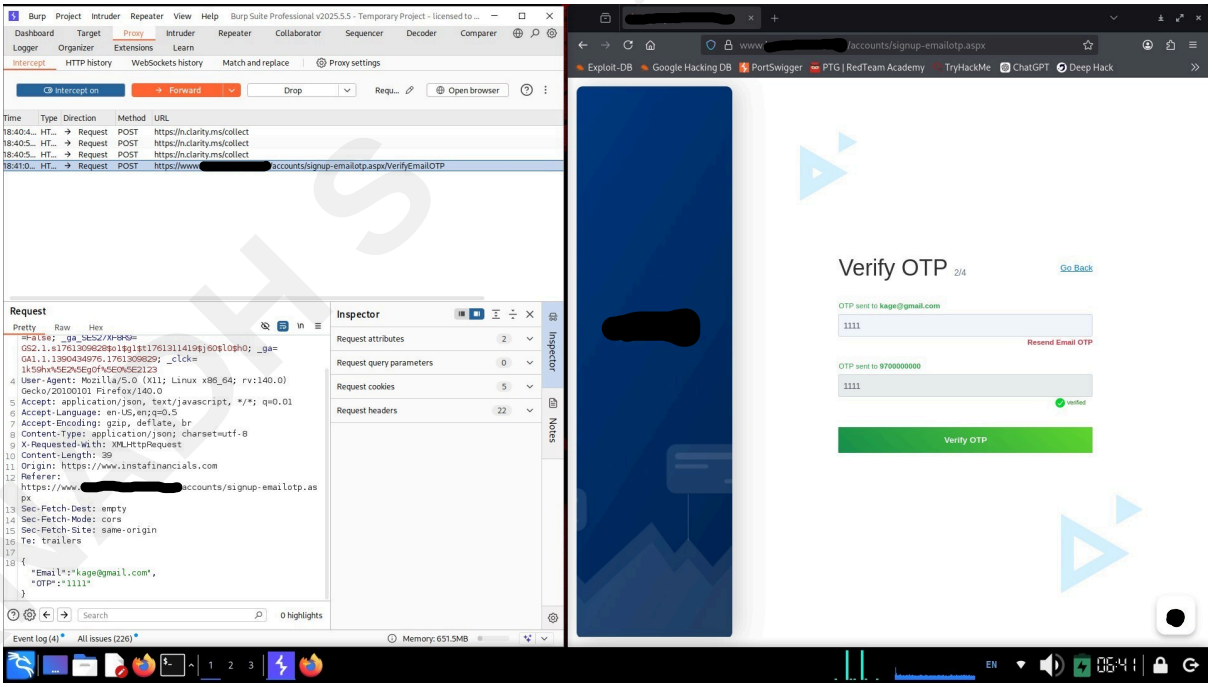
Screenshots

Number OTP Verification

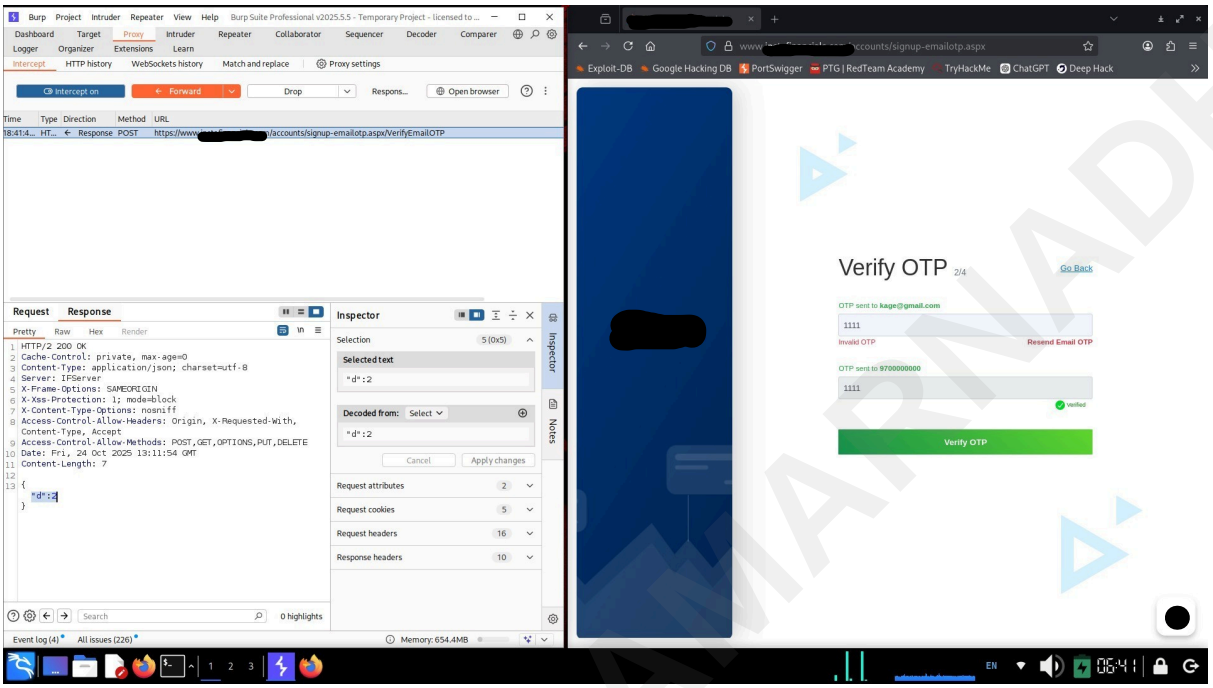


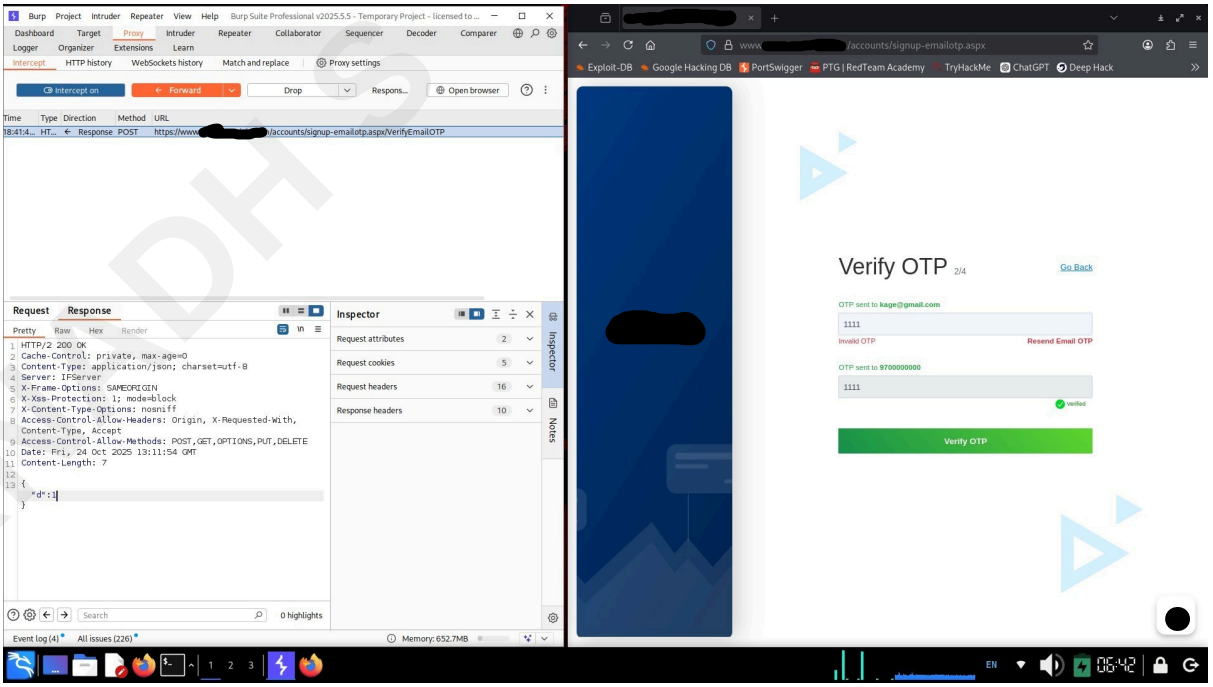


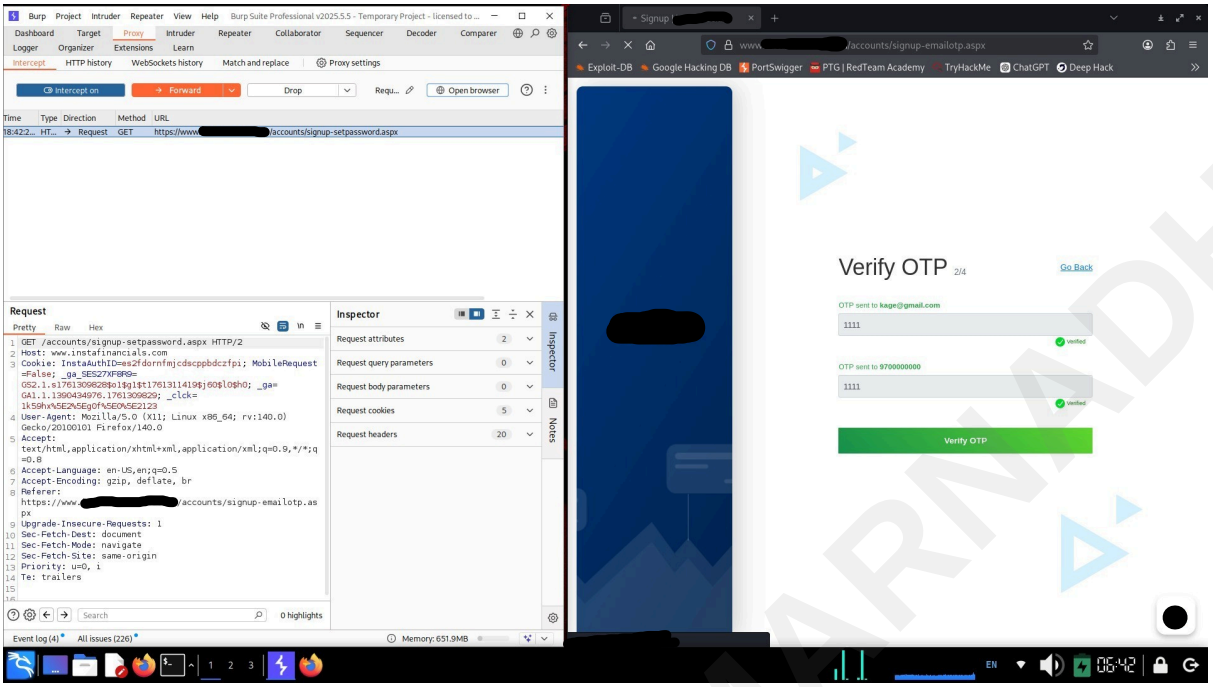

Verify OTP 2/4
Go Back
OTP sent to kage@gmail.com
1111
Resend Email OTP
OTP sent to 9700000000
1111
Resend Mobile OTP
Verify OTP

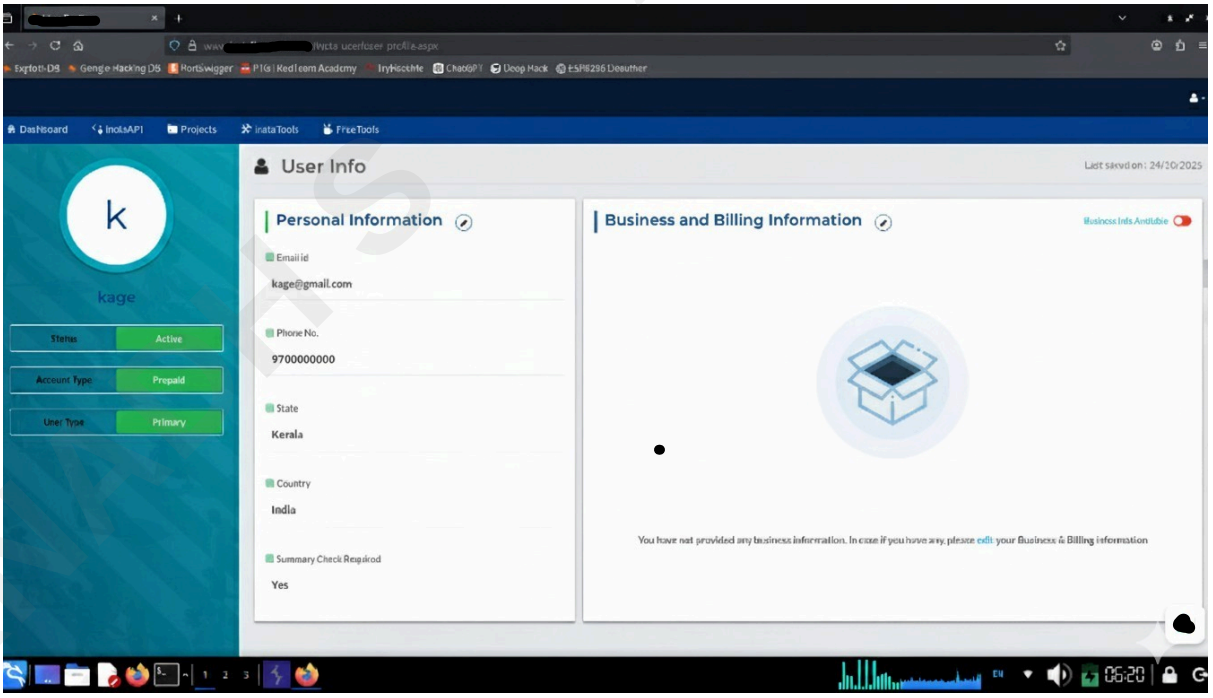

Verify OTP 2/4
Go Back
OTP sent to kage@gmail.com
1111
Resend Email OTP
OTP sent to 9700000000
1111
Verify OTP

Email OTP Verification






Verify OTP 2/4
Go Back
OTP sent to kage@gmail.com
1111
OTP sent to 9700000000
1111
Verify OTP


Dashboard
Projects
User Info
Personal Information
Email id
kage@gmail.com
Phone No.
9700000000
State
Kerala
Country
India
Summary Check Required
Yes
Business and Billing Information
kage
Status
Active
Account Type
Prepaid
User Type
Primary

Recommendations :

1. Prioritize Phishing-Resistant 2FA Factors:

- Authenticator Apps (TOTP): Encourage users to use Time-based One-Time Password (TOTP) apps (like Google/Microsoft Authenticator) instead of SMS. TOTP codes are generated on the user's device and are not susceptible to SIM swap or network interception attacks.

- Security Keys (FIDO2/WebAuthn): Implement support for WebAuthn/FIDO2 security keys (e.g., YubiKey). This is the most secure and phishing-resistant form of multi-factor authentication.

2. Adopt Adaptive MFA Policies:

- Implement a system where the authentication factor required is based on context. For example, a user logging in from a new IP address or device requires a stronger factor (like TOTP) than a user logging in from a known, trusted device.

3. Enhance User Awareness:

- Educate Users: Continuously educate users that legitimate institutions will never ask for their OTP over a phone call or unprompted message. OTPs are only to be entered by the user on the official website/app during a transaction they initiated.

Reference :

https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/04-Testing_for_Bypassing_Authentication_Schema

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/11-Testing_Multi-Factor_Authentication

https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

https://www.authgear.com/post/otp-bots-bypass-sms-2fa

https://wpscan.com/vulnerability/eedb0d3f-6dc2-4f08-af21-767acc4d1301/

Vulnerability #3 : Reflected XSS

Severity : Medium (4.9)

Vulnerable URL : [redacted]

Security Impact : Medium

Reflected Cross-Site Scripting occurs when an application takes unsanitized user input (usually from a query parameter, form field, or URL) and reflects it immediately in an HTTP response (HTML, JS, or other contexts). If that reflected content contains executable script content, a victim who opens the crafted URL executes that script in their browser context.

Impacts :

- Account takeover (via CSRF chaining, token capture)
- Credential & data exfiltration (reads DOM, sends data to attacker)
- Drive-by malware / phishing (injects UI that looks legitimate)
- Trust & compliance damage (legal/regulatory fallout, brand damage)

Mitigations :

- Context-Aware Output Encoding/Escaping (Primary Defense):
- The most critical defense is to treat all user-controllable data as data, not as executable code, when it's rendered in the HTML response.
- You must encode special characters based on where the data is being placed in the HTML (its context). For example:

- HTML Entity Encoding for data placed directly into the HTML body (< becomes <, > becomes >, " becomes ").

- HTML Attribute Encoding for data placed inside an HTML attribute (e.g., in a value attribute).

- JavaScript Encoding for data placed inside a script block.

- Use built-in framework features: Modern web frameworks (React, Angular, Vue.js, etc.) and template engines (like Jinja2, Thymeleaf) often include automatic output escaping by default, which is highly Recommended.

- Validation: Check that user input conforms to expected formats (e.g., a phone number field only contains numbers and dashes, an email field matches an email regex). Use an allow-list approach (allowing only known-safe input) over a block-list (blocking known-bad input).

- Sanitization (for user-authored HTML): If you must allow users to input HTML (e.g., in a WYSIWYG editor), use a robust library (like DOMPurify on the client-side) to filter out dangerous elements and attributes (e.g., <script>, onerror, onload).

- Implement a strong Content Security Policy via an HTTP response header.• CSP is a browser-side defense that acts as a second line of defense by restricting the sources from which a browser can load scripts, style sheets, and other resources.

- A strict CSP can prevent the execution of inline scripts and scripts loaded from untrusted domains, even if an XSS vulnerability exists.

- Set the HttpOnly flag on all cookies, especially session cookies. This prevents client-side scripts (including malicious ones injected via XSS)from accessing the cookies, which limits the attacker's ability to steal session information.

- Use the Secure flag to ensure cookies are only sent over HTTPS.

- While mostly deprecated in favor of CSP, some browsers previously implemented XSS filters (e.g., Chrome's XSS Auditor), but reliance on these is not a reliable primary mitigation.

Steps to Reproduced :

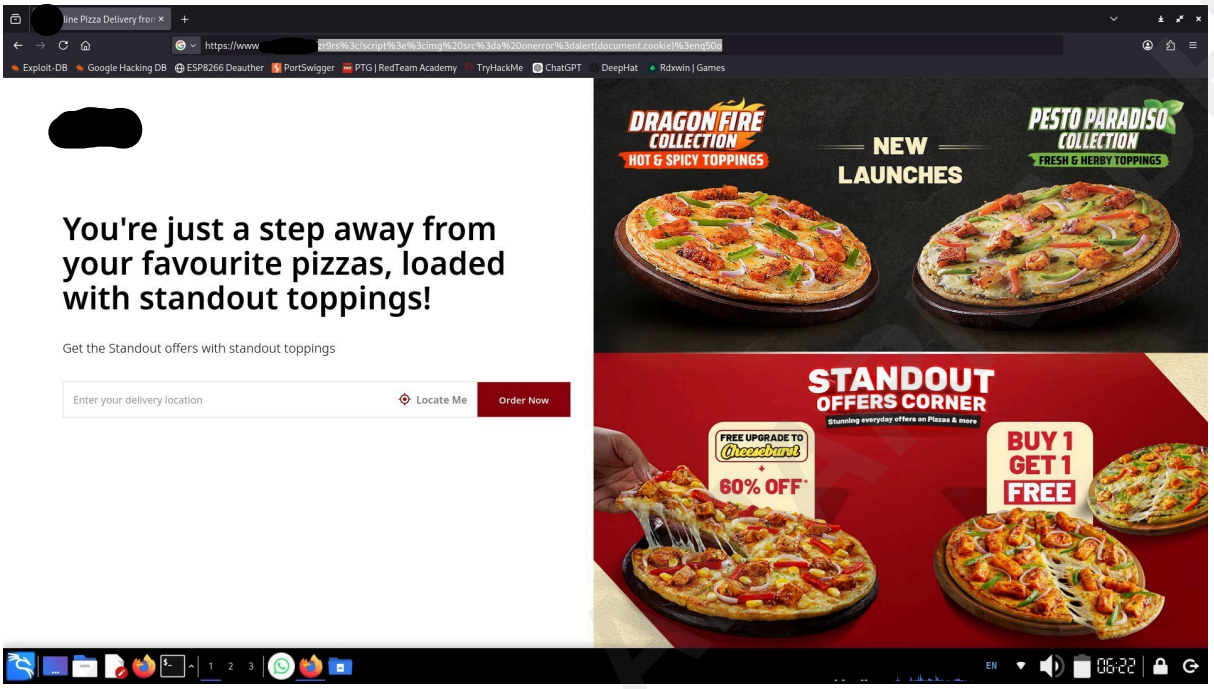
Go to target URL : [redacted]

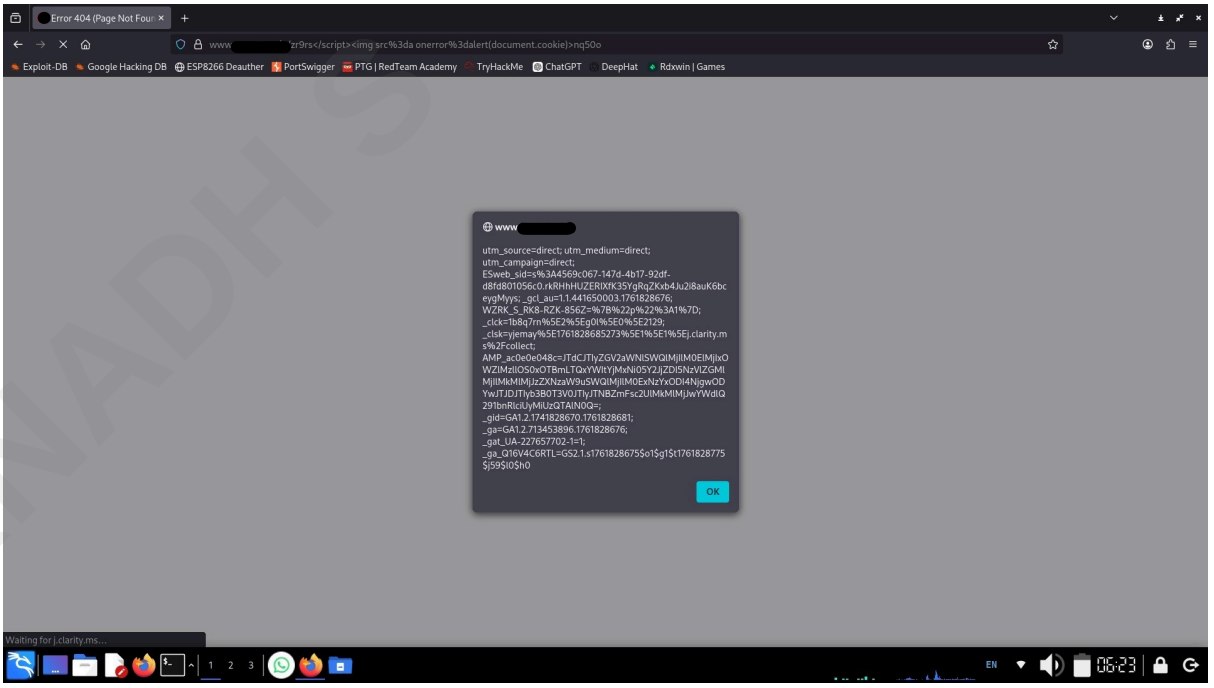
Give this payload in the URL :

zr9rs%3c/script%3e%3cimg%20src%3da%20onerror%3dalert(document.cookie)%3enq50o

Enter the payload and the cookies will be shown

Screenshots :





Recommendations :

1. Adopt a Secure Coding Policy

- Enforce coding standards (e.g., OWASP Secure Coding Practices).
- Include XSS prevention in development checklists.

2. Use a Modern Web Framework

- Prefer frameworks that automatically escape user input and restrict inline scripts (e.g., React, Angular, Django).

3. Implement Content Security Policy (CSP)

- Configure CSP to block inline scripts and restrict script sources to trusted domains.
- Use nonces or hashes for necessary inline scripts.

4. Conduct Regular Security Testing

- Perform vulnerability scanning and penetration testing on all input points.
- Include XSS testing in CI/CD pipelines using automated tools.

5. Educate Developers and Review Code

- Conduct periodic training on XSS and input sanitization.
- Mandate peer code reviews focusing on user input/output handling.

6. Harden Browser-Side Security

- Enable HttpOnly, Secure, and SameSite cookie flags.
- Use HTTPS exclusively to protect traffic from injection or Interception.

7. Implement Incident Response Measures

- Maintain logging for abnormal URL parameters or reflected payloads.
- Have a process for patching and disclosing XSS vulnerabilities Quickly.

8. Enforce Least Privilege Principle

- Limit privileges of web applications and APIs to reduce impact if exploitation occurs.

9. Continuous Monitoring & Compliance

- Monitor for attempted script injections via SIEM tools.
- Ensure compliance with OWASP Top 10 and industry standards.

Reference :

https://portswigger.net/web-security/cross-site-scripting/reflected

https://owasp.org/www-community/attacks/xss/

https://portswigger.net/web-security/cross-site-scripting/reflected/lab-html-context-nothing-encoded

https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting

https://brightsec.com/blog/reflected-xss/

Vulnerability #4 : DNS Misconfiguration

Severity : Medium (6.5)

Vulnerable URL : [redacted]

Security Impact : Medium

A DNS misconfiguration was identified during the assessment. Improperly configured DNS records can expose sensitive information about the organization's infrastructure or allow attackers to exploit subdomains. Examples include open zone transfers, dangling DNS entries, publicly accessible internal records, and the absence or misconfiguration of DNSSEC. Such issues can assist attackers in reconnaissance, spoofing, or hijacking attacks.

Impacts :

Information Disclosure: Exposure of internal subdomains, IP addresses, and server details allows attackers to map the internal network.

Subdomain Takeover: Dangling DNS records pointing to decommissioned resources (e.g., old cloud instances) can be exploited to gain control over subdomains.

Phishing and Brand Abuse: Misconfigured domains can be leveraged to impersonate legitimate services, facilitating phishing or fraud.

Email Spoofing: Absence or misconfiguration of SPF, DKIM, or DMARC records can allow unauthorized email spoofing, damaging domain reputation.

DNS Hijacking: Insecure configurations may allow manipulation or poisoning of DNS entries, redirecting users to malicious destinations.

Service Disruption: Incorrect DNS entries can lead to unavailability of legitimate applications or services.

Man-in-the-Middle Attacks: Lack of DNSSEC validation enables attackers to forge DNS responses, redirecting traffic to malicious servers.

Mitigations :

1. **Restrict Zone Transfers:**
    - Configure DNS servers to allow **zone transfers (AXFR)** only from authorized secondary servers or specific IP addresses.
    - Disable zone transfers entirely if not required.
2. **Disable Recursion on Public DNS Servers:**
    - Prevent public DNS servers from performing recursive queries to avoid DNS amplification and information disclosure.
3. **Implement DNSSEC:**
    - Enable and properly configure **DNS Security Extensions (DNSSEC)** to ensure DNS responses are validated and tamper-proof.
4. **Harden DNS Records:**
    - Review and remove **unused, outdated, or dangling DNS entries** that point to inactive cloud services or domains.
    - Avoid wildcard (*) DNS records unless absolutely necessary.
5. **Protect Internal Records:**
    - Ensure that **internal or private subdomains** (e.g., dev, test, staging) are not publicly resolvable via external DNS servers.
6. **Secure Mail-Related DNS Configurations:**
    - Properly configure **SPF**, **DKIM**, and **DMARC** records to prevent email spoofing and domain impersonation.
7. **Regular DNS Audits:**
    - Conduct **periodic reviews** of DNS configurations, zone files, and name servers to identify anomalies or unauthorized changes.
    - Use tools like dnsrecon, fierce, dig, or external scanners such as **MXToolbox** or **DNSdumpster**.

8. **Access Control and Logging:**
    - Restrict administrative access to DNS servers using **least privilege principles**.
    - Enable **detailed logging** to monitor zone transfers, record modifications, and suspicious DNS activities.
9. **Use Redundant and Secure DNS Infrastructure:**
    - Deploy **redundant authoritative name servers** in different networks or geographic regions for availability and resilience.
    - Keep DNS server software and operating systems **up to date** with the latest security patches.
10. **Monitor for Subdomain Takeovers:**
    - Regularly scan for **dangling subdomains** and decommissioned third-party assets (e.g., AWS S3, Azure, GitHub Pages).
    - Implement automated alerts for DNS record changes and certificate issuance events (e.g., via **Crt.sh** or **SecurityTrails**).

Steps to Reproduce :

Go to target URL : [redacted]

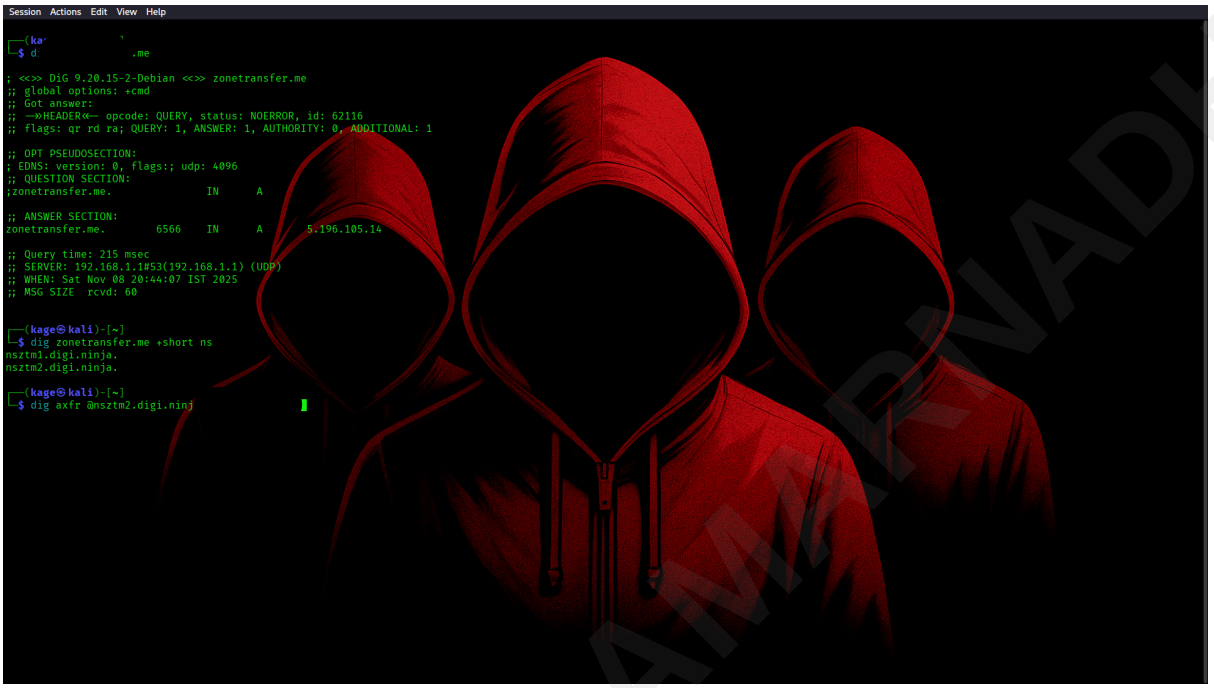
Open terminal in kali linux

Type the command - dig zonetransfer.me +short ns
(this will show potential vulnerable sub domains)

Then select the subdomain you want to execute (nsztm2.digi.ninja.)

Then type the command - dig axfr @nsztm2.digi.ninja zonetransfer.me

Screenshots :



Recommandations :

- Restrict DNS zone transfers (AXFR) to authorized IP addresses only.
- Disable recursion on publicly accessible DNS servers.
- Remove or update outdated and unused DNS records.
- Avoid exposing internal or non-public records through external DNS servers.
- Implement and correctly configure **DNSSEC** to ensure authenticity and integrity of DNS responses.
- Ensure **SPF**, **DKIM**, and **DMARC** records are properly configured to mitigate email spoofing.
- Conduct regular DNS audits and monitor for unauthorized or dangling records.
- Use monitoring tools such as **dig**, **dnsrecon**, **fierce**, or **DNSdumpster** for periodic verification.

Reference :

https://owasp.org/www-project-secure-configuration-guide/v1/DNS-Security.html

https://cwe.mitre.org/data/definitions/710.html

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-81r2.pdf

https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/securing-dns

https://www.cloudflare.com/learning/dns/dns-security/

APPENDIX : A

Tools Used

Burp suite Professional

Burp Suite is used for testing and securing web applications by finding and analyzing vulnerabilities.

Nuclei

Nuclei is used for fast, automated vulnerability scanning using templates to detect security issues in web applications, networks, and APIs.

Dirsearch

Dirsearch is used for discovering hidden directories and files on a website through brute-force scanning.

CVSS (Common Vulnerability Scoring System)

CVSS (Common Vulnerability Scoring System) is used to measure and rate the severity of security vulnerabilities using a standardized numerical score.

OWASP Resources

OWASP resources are used to improve web application security by providing guides, tools, best practices, and frameworks such as the OWASP Top 10, ASVS, and testing methodologies.

Kali Linux

Kali Linux is a operating system used for penetration testing, ethical hacking, and security auditing, providing a wide range of built-in security tools.

APPENDIX : B

ENGAGEMENT INFORMATION

For any queries, clarifications, or additional information related to the findings and recommendations presented in this Vulnerability Assessment and Penetration Testing (VAPT) report, please feel free to contact the undersigned. I will be available to provide further explanations, supporting evidence, or technical details as required to assist in understanding or resolving the identified issues.

Contact Information

NAME	**AMARNADH S**
PHONE	**9074023731**
EMAIL	**xamarnadh@gmail.com**

Conclusion

The primary objective of this assessment is to systematically identify, document, and prioritize critical vulnerabilities within a web application and ensure their timely resolution. In today's digital landscape, where organizations heavily rely on their online presence, securing systems is essential. As cyber threats grow more sophisticated, attackers constantly search for weaknesses that allow unauthorized access, data theft, or operational disruption. Therefore, identifying and resolving vulnerabilities is vital to protecting an organization's assets, data, and reputation.

Vulnerability assessments provide a proactive approach to uncover potential threats before they can be exploited. By thoroughly evaluating the security posture of a web application, organizations gain valuable insights into areas needing improvement. This enables informed decision-making and effective risk mitigation. Instead of reacting to breaches, assessments allow organizations to address weaknesses early, reducing exposure to cyber attacks.

Timely remediation of vulnerabilities significantly lowers exploitation risks. Implementing appropriate security controls and patching weaknesses strengthens application resilience, safeguards sensitive information, and ensures business continuity. In regulated industries, this also prevents legal, financial, and reputational consequences, reinforcing the trust of users and stakeholders.

The process of conducting a vulnerability assessment is not just about recognizing potential threats but also about evaluating the risks they pose to the organization's operations, infrastructure, and overall reputation Different vulnerabilities present varying levels of risk, depending on factors such as the type of data involved,the potential impact of exploitation, and the likelihood of an attack. A comprehensive vulnerability assessment allows organizations to prioritize their efforts by addressing the most critical and high-risk issues first. By doing so, they ensure that the most pressing security concerns are resolved before they can lead to more significant problems.

In summary, vulnerability assessments are essential for securing web applications and digital environments. They help organizations stay ahead of threats, mitigate risks, and maintain a strong security posture. Continuous assessment and remediation ensure the integrity, confidentiality, and availability of systems while preserving the trust and confidence of customers and stakeholders.