

**Zero bank webappsecurity**

**VAPT report**

**Target -** [http://zero.webappsecurity.com](http://zero.webappsecurity.com/index.html)

**Executive summary -** In this report, we dive into the security of Target Zero Bank's website, focusing on potential vulnerabilities outlined in the OWASP Top 10 2021 list. We've uncovered critical issues that need attention, along with practical steps to boost security and safeguard against risks.

Performed BY – Abhinay Singh (ayushrajput62687@gmail.com)  
  
Date – 20/03/2024 Assigned by – Nullclass

**CONTENTS**

* **Vulnerability and findings (owasp top 10 2021) -**

**AO1: Broken Access Control**

* + IDOR (Insecure direct object reference)
  + Vertical Privilege Escalation

**AO1: Cryptographic Failures**

* + SSL certificate Expired

**AO3: Injection**

* + Reflected XSS (account activity)
  + Reflected XSS (transfer funds)

**AO4: Insecure Design**

* + Lack HTTPs Connection
  + Missing Login Lockup Mechanism

**AO5: Security Misconfiguration**

* + Default login password found
  + Vulnerable directories found

**AO6: Vulnerable and Outdated Components**

* + Vulnerable components found

**AO7: Identification and Authentication Failures**

* + Authentication ID and password exposed
  + Session ID unchangeable

**AO8: Software and Data Integrity Failures**

**AO9: Security Misconfiguration**

**AO10: SSRF**

* + SSRF against the server
* **Conclusion**

**Vulnerabilities and findings (OWASP TOP 10)**

**AO1: Broken Access control**

|  |  |
| --- | --- |
| **1.1 IDOR (**I**nsecure direct object reference)** | |
| **Description** | |
| In the Account Activity section of the website, an IDOR (Insecure Direct Object Reference) flaw was discovered, allowing unauthorized users to access sensitive account data by manipulating URLs. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com/bank/account-activity.html?accountId=1 | **HIGH** |
| **Impact/Consequences** | |
| The IDOR (Insecure Direct Object Reference) finding poses significant risks, potentially granting malicious actors unauthorized access to sensitive account information. Exploitation could lead to unauthorized viewing, modification, or deletion of data across different user accounts, compromising privacy and security. Immediate action is imperative to prevent exploitation and safeguard user data | |
| **Recommendation** | |
| To address the IDOR vulnerability, ensure that users can only access their own data by setting up strict access controls. Use unique IDs and double-check user permissions on the server side to prevent unauthorized access. Regularly test for vulnerabilities to stay ahead of potential security risks. | |
| **Tools Used** | **References** |
| None | <https://portswigger.net/web-security/access-control/idor>  https://cheatsheetseries.owasp.org/cheatsheets/Insecure\_Direct\_Object\_Reference\_Prevention\_Cheat\_Sheet.html |
| **CWE** | **OWASP TOP 10 (2021)** |
| 639 | AO1: Broken Access Control |
| **Proof Of Vulnerability** | |
| **Step 1:** Enter the targeted website and click on the login page. Sign in with userrname“username” and password “password”. This is taken you to user page. | |
| **Step 2:** After successfully logging in, enter the username check account activity. This will open the section where all transactions performed by the user are present.    **Step 3:** Check the URL of this page and attempt to modify the account ID number from 1 to 2, 1234, etc. Verify if it allows access to another person's transactions.    **Step 4:** It’ll show transactions of different users. | |

|  |  |
| --- | --- |
| **1.2 Vertical Privilege Escalation** | |
| **Description** | |
| Vertical privilege escalation is akin to someone gaining higher clearance within a system, like being granted access to the admin page. This is achieved by exploiting weaknesses or adjusting one's user role, allowing them to surpass their initial access level and obtain elevated privileges. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com/admin | **HIGH** |
| **Impact/Consequences** | |
| Vertical privilege escalation allows someone to bypass normal access controls, granting them higher levels of authority. This means they could access sensitive information, make unauthorized changes, or disrupt services. It's like someone sneaking into a VIP area, gaining power they shouldn't have, potentially causing chaos. | |
| **Recommendation** | |
| To mitigate vertical privilege escalation, implement strict access controls, regularly review user permissions, and employ least privilege principles. Conduct frequent security audits to identify and patch vulnerabilities. Utilize robust authentication mechanisms like multi-factor authentication. Provide comprehensive training to users on security best practices and awareness of social engineering tactics to prevent exploitation of human error in granting elevated access. | |
| **Tools Used** | **References** |
| None | https://portswigger.net/web-security/access-control |
| **CWE** | **OWASP TOP 10 (2021)** |
| 250 | AO1: Broken Access Control |
| **Proof Of Vulnerability** | |
| **Step 1:** Enter the targeted website and check the URL.    **Step 2:** Try to manipulate the URL by adding /admin to the URL and see if you can access the admin page.    **Step 3:** You can see that it provided us with admin page with users and additional information | |

**AO2: Cryptographic Failures**

|  |  |
| --- | --- |
| **2.1 SSL Certificate Expired** | |
| **Description** | |
| An SSL certificate is a digital certificate that authenticates the identity of a website and enables secure connections through encryption. However, when SSL certificates expire, they pose a vulnerability, leaving websites susceptible to attacks like man-in-the-middle interceptions and data breaches. Regularly monitoring certificate expiration dates, renewing certificates promptly, and implementing automated alerts can mitigate this risk, ensuring continuous protection of sensitive data transmitted over the web. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com | **CRITICAL** |
| **Impact/Consequences** | |
| SSL certificate expiration disrupts secure connections, leaving websites vulnerable to data breaches and interception of sensitive information by cyber attackers. It erodes user trust, damages reputation, and can result in financial losses due to regulatory penalties and potential lawsuits. Additionally, it may lead to a loss of website functionality, decreased traffic, and negatively impact SEO rankings, highlighting the critical need for timely certificate renewal. | |
| **Recommendation** | |
| Regularly monitor SSL certificate expiration dates and set up automated alerts for renewal reminders. Implement a robust certificate management system to track certificates across all servers and domains. Establish a proactive renewal process to ensure certificates are updated before expiration. Utilize reputable Certificate Authorities for SSL certificate issuance. Conduct periodic security assessments to identify and address any vulnerabilities related to SSL certificate management. | |
| **Tools Used** | **References** |
| OpenSSL | https://cwe.mitre.org/data/definitions/298.html |
| **CWE** | **OWASP TOP 10 (2021)** |
| 298 | AO2: Cryptographic Failures |
| **Proof Of Vulnerability** | |
| **Step 1:** Open the target website and copy the URL. | |
| **Step 2:** Open Kali Linux on VMware or VirtualBox and open command prompt. | |
| **Step 3:** Type the below command for using OpenSSL tool to manually retrieve information about certificate, protocol, cipher suite of the target website. openssl s\_client -connect zero.webappsecurity.com:443  You’ll see the below information. | |
| **Step 4:** You’ll see information about the certificate and note that it has expired. | |

**AO3: Injection**

|  |  |
| --- | --- |
| **3.1 Reflected XSS Injection (account activity)** | |
| **Description** | |
| Reflected XSS occurs when a malicious script is injected into a web application's input and then reflected to the user within the application's response. Typically, this occurs via URLs with query parameters or form submissions. When users interact with these crafted URLs or forms, the injected script executes within their browser, potentially leading to unauthorized actions, data theft, or other security breaches. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com/bank/account-activity.html?accountId=1 | **CRITICAL** |
| **Impact/Consequences** | |
| Reflected XSS can lead to severe consequences, including unauthorized access to sensitive user data, theft of credentials or financial information, manipulation of account transactions, and even complete website takeover. Such exploits pose significant risks to both users and organizations, necessitating immediate mitigation efforts to prevent exploitation. | |
| **Recommendation** | |
| To mitigate Reflected XSS vulnerabilities, implement input validation and output encoding to sanitize user inputs effectively. Utilize security mechanisms like Content Security Policy (CSP) to restrict the execution of scripts and prevent unauthorized code injection. Regular security audits, code reviews, and web application firewalls are also crucial for detecting and addressing potential vulnerabilities proactively. | |
| **Tools Used** | **References** |
| None | https://fossa.com/blog/all-about-cwe-79-cross-site-scripting/ |
| **CWE** | **OWASP TOP 10 (2021)** |
| 79 | AO3: Injection |
| **Proof Of Vulnerability** | |
| **Step 1:** Enter the target website and sign in with your Username and password | |
| **Step 2:** After signing in, click on more services and select account summary. It’ll bring you to this page. | |
| **Step 3:** Go to savings. It’ll display all the savings and transactions you’ve made. | |
| **Step 4:** Go to the above URL and add the script highlighted below.  <http://zero.webappsecurity.com/bank/account-activity.html?accountId=1);alert(1);//1> | |
| **Step 5:** Add the script and run the website again. It’ll show the running script demonstrating reflected XSS vulnerability. | |

|  |  |
| --- | --- |
| **3.2 Reflected XSS Injection (transfer funds)** | |
| **Description** | |
| Reflected XSS occurs when a malicious script is injected into a web application's input and then reflected to the user within the application's response. Typically, this occurs via URLs with query parameters or form submissions. When users interact with these crafted URLs or forms, the injected script executes within their browser, potentially leading to unauthorized actions, data theft, or other security breaches. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com/bank/transfer-funds.html | **CRITICAL** |
| **Impact/Consequences** | |
| Reflected XSS can lead to severe consequences, including unauthorized access to sensitive user data, theft of credentials or financial information, manipulation of account transactions, and even complete website takeover. Such exploits pose significant risks to both users and organizations, necessitating immediate mitigation efforts to prevent exploitation. | |
| **Recommendation** | |
| To mitigate Reflected XSS vulnerabilities, implement input validation and output encoding to sanitize user inputs effectively. Utilize security mechanisms like Content Security Policy (CSP) to restrict the execution of scripts and prevent unauthorized code injection. Regular security audits, code reviews, and web application firewalls are also crucial for detecting and addressing potential vulnerabilities proactively. | |
| **Tools Used** | **References** |
| None | https://fossa.com/blog/all-about-cwe-79-cross-site-scripting/ |
| **CWE** | **OWASP TOP 10 (2021)** |
| 79 | AO3: Injection |
| **Proof Of Vulnerability** | |
| **Step 1:** Enter the target website and sign in with your Username and password | |
| **Step 2:** After signing in, click on more services and select Transfer funds. It’ll bring you to this page. | |
| **Step 3:** Fill the details with random things and write this script below in the description box. <svg onload="alert(1)"> “><svg onload="alert(1)//"> | |
| **Step 4:** After submitting, you can see successful XSS injection. | |

**AO4: Insecure Design**

|  |  |
| --- | --- |
| **4.1 Lack of HTTPs connection** | |
| **Description** | |
| HTTP (Hypertext Transfer Protocol) is a foundational protocol for communication on the web, facilitating the exchange of data between clients and servers. However, its lack of encryption leaves data vulnerable to interception and manipulation. HTTPS (HTTP Secure) adds a layer of security through encryption, safeguarding against such vulnerabilities. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com | **HIGH** |
| **Impact/Consequences** | |
| The lack of HTTPS connection exposes users to numerous risks. Attackers can intercept sensitive data, such as login credentials and payment information, leading to identity theft and financial losses. Without encryption, the integrity of data is compromised, enabling manipulation or injection of malicious content. Additionally, absence of HTTPS diminishes user trust, damages reputation, and can result in legal liabilities for businesses, highlighting the critical need for secure communication protocols on the web. | |
| **Recommendation** | |
| Implement HTTPS across all web communications to ensure data security and user privacy. Obtain SSL/TLS certificates from reputable providers and configure web servers to enforce HTTPS encryption. Regularly update and monitor SSL configurations to maintain robust security posture. Educate users about the importance of HTTPS and encourage secure browsing practices. | |
| **Tools Used** | **References** |
| None | N/A |
| **CWE** | **OWASP TOP 10 (2021)** |
| 311 | AO4: Insecure Design |
| **Proof Of Vulnerability** | |
| **Step 1:** Enter the targeted website and check the URL.  **Step 2:** As you can see it’s using HTTP instead of HTTPs (secure).  **Step 3:** If you try to access it and sign in, you get a warning for insecure connection. | |

|  |  |
| --- | --- |
| **4.2 Missing Login lockup mechanism** | |
| **Description** | |
| Login lockup mechanism prevents unauthorized access by locking out users after a specified number of failed login attempts, enhancing security. However, the absence of this mechanism exposes systems to vulnerabilities, allowing attackers to launch brute force attacks without restrictions, potentially compromising sensitive data and system integrity. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com/login.html | **HIGH** |
| **Impact/Consequences** | |
| The impact of a missing login lockup mechanism can be severe. Without such a safeguard, systems are vulnerable to brute force attacks where attackers can repeatedly attempt to guess passwords without restriction. This can lead to unauthorized access, data breaches, compromised user accounts, loss of sensitive information, financial damage, reputational harm, and legal consequences for organizations. | |
| **Recommendation** | |
| To mitigate the risk of unauthorized access, implement a robust login lockout mechanism. Configure the system to temporarily lock user accounts after a specified number of failed login attempts. Customize lockout durations and consider implementing exponential backoff to deter brute force attacks. Enhance security further by deploying multi-factor authentication. Regularly monitor login attempts for anomalies and suspicious activity to promptly identify and mitigate potential security threats. | |
| **Tools Used** | **References** |
| None | N/A |
| **CWE** | **OWASP TOP 10 (2021)** |
| - | AO4: Insecure Design |
| **Proof Of Vulnerability** | |
| **Step 1:** Enter the targeted website and go to the sign-up button to get access to the login page. | |
| **Step 2:** Try to login with username and password. Make it wrong intentionally. | |
| **Step 3:** If the login failed, Try again. You’ll see that there is no limit to the number of login attempts. | |

**AO5: Security Misconfiguration**

|  |  |
| --- | --- |
| **5.1 Default Login Password Found** | |
| **Description** | |
| Default login passwords are preset credentials often used by manufacturers for initial access to devices or systems. However, the presence of default login passwords poses a significant security vulnerability. Attackers exploit this weakness to gain unauthorized access, potentially compromising sensitive data and system integrity. It's imperative to change default passwords immediately after setup to mitigate this vulnerability and enhance security. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com/login.html | **CRITICAL** |
| **Impact/Consequences** | |
| Discovering default login passwords poses grave consequences, enabling unauthorized access to systems. This vulnerability compromises sensitive data, risks financial loss, damages reputation, and incurs legal ramifications. Immediate action, such as changing default passwords and implementing robust authentication measures, is essential to mitigate these severe impacts on security and integrity. | |
| **Recommendation** | |
| Address the default login password found vulnerability by promptly changing default credentials upon setup. Utilize strong, unique passwords for each device/system and employ multi-factor authentication. Regularly audit and update passwords to mitigate risks. Educate users on password security practices, emphasizing the importance of strong, unique passwords and regular changes to bolster overall security posture. | |
| **Tools Used** | **References** |
| None | https://cwe.mitre.org/data/definitions/1393.html |
| **CWE** | **OWASP TOP 10 (2021)** |
| 1393 | AO5: Security Misconfiguration |
| **Proof Of Vulnerability** | |
| **Step 1:** Enter the targeted website and go to the sign-up button to get access to the login page. | |
| **Step 2:** Take your cursor to the question mark(?) icon after the username text field. | |
| **Step 3:** You can see that the default username “username” and password ‘password’ is visible. | |

|  |  |
| --- | --- |
| **5.2 Vulnerable Directories found** | |
| **Description** | |
| Directories are hierarchical structures used to organize files on computer systems. Vulnerable directories are those prone to security breaches due to misconfigurations or lax access controls. Common vulnerabilities include unrestricted file access, insufficient authentication measures, and directory traversal attacks. Such vulnerabilities can lead to unauthorized data access, modification, or deletion, posing significant risks to system integrity and confidentiality. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com | **HIGH** |
| **Impact/Consequences** | |
| Finding vulnerable directories can have severe consequences, including unauthorized access to sensitive data, data breaches, and loss of data integrity. This can result in legal liabilities, financial losses, damage to reputation, and disruption of operations. Prompt remediation is essential to mitigate risks and safeguard the organization's assets and reputation. | |
| **Recommendation** | |
| To uncover vulnerable directories, conduct regular security assessments using tools like vulnerability scanners and web application scanners. Implement strong access controls, ensuring directories are only accessible to authorized users. Employ security best practices such as encryption, secure coding practices, and regular patch management. Conduct thorough code reviews to identify and address potential vulnerabilities. Additionally, educate staff on security awareness to prevent inadvertent exposure of directories. Regular audits and monitoring are essential for maintaining a secure directory environment. | |
| **Tools Used** | **References** |
| Gobuster | https://www.aquasec.com/cloud-native-academy/supply-chain-security/security-misconfigurations/ |
| **CWE** | **OWASP TOP 10 (2021)** |
| 16 | AO5: Security Misconfiguration |
| **Proof Of Vulnerability** | |
| **Step 1:** I used my kali Linux machine and used Gobuster tool to scan for any unprotected directories that might be vulnerable. | |
| **Step 2:** This is what I found. | |
| **Step 3:** I then change the url and add /admin to see if I can access the admin page and I was successful in accessing it. | |

**AO6: Vulnerable and Outdated Components**

|  |  |
| --- | --- |
| **6.1 Vulnerable and outdated components found** | |
| **Description** | |
| Vulnerable and outdated components refer to software elements within a system that are susceptible to exploitation due to unpatched security flaws or obsolescence. These components pose significant risks to digital infrastructure, potentially facilitating cyberattacks or system failures. Timely updates and patches are crucial to mitigate these vulnerabilities, ensuring the resilience and security of the overall system against evolving threats. | |
| **Affected Resources/Parameters** | **Severity** |
| http://zero.webappsecurity.com | **CRITICAL** |
| **Impact/Consequences** | |
| Vulnerable and outdated components can lead to severe consequences for organizations, including security breaches, system compromise, financial losses, and reputational damage. Breaches may result in unauthorized access, data theft, and operational disruptions, while financial losses can arise from remediation efforts, regulatory fines, and legal fees. Moreover, compromised systems may violate compliance standards, further exacerbating risks. These impacts can erode trust, lead to customer attrition, and perpetuate vulnerability cycles if left unaddressed. | |
| **Recommendation** | |
| To address vulnerable and outdated systems, prioritize a comprehensive cybersecurity strategy. Conduct regular assessments to identify weaknesses, apply patches promptly, and update software regularly. Implement robust access controls, network segmentation, and intrusion detection systems to fortify defenses. Invest in employee training to enhance awareness of security best practices. Consider adopting automation and AI-driven solutions for proactive threat detection and response. Collaborate with industry peers and stay abreast of emerging threats to ensure a resilient security posture. | |
| **Tools Used** | **References** |
| Whatweb | https://owasp.org/Top10/A06\_2021-Vulnerable\_and\_Outdated\_Components/ |
| **CWE** | **OWASP TOP 10 (2021)** |
| 1352 | AO6: Vulnerable and Outdated compnents |
| **Proof Of Vulnerability** | |
| **Step 1:** After opening the website, I first tried to check if the website is hosted by WordPress by adding /wp-admin. It showed error but it told me the server version. This version of tomcat possesses vulnerabilities http request smuggling. | |
| **Step 2:** After opening kali linux, I run the Whatweb tool to gather information about the resources. And I found all this. | |
| **Step 3:** You can see that the jQuery version has been outdated. This version commonly possesses vulnerabilities like XSS. | |
| **Step 4:** Check for latest updates on jQuery and update. | |

**AO7: Identification and Authentication Failures**

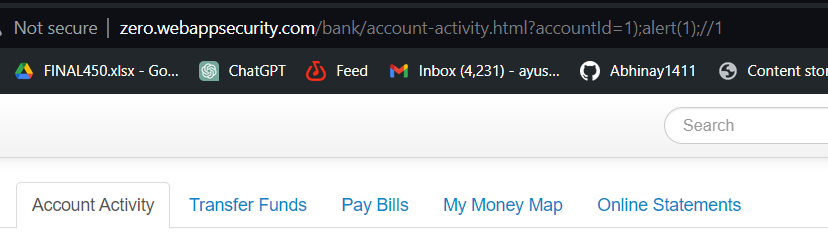
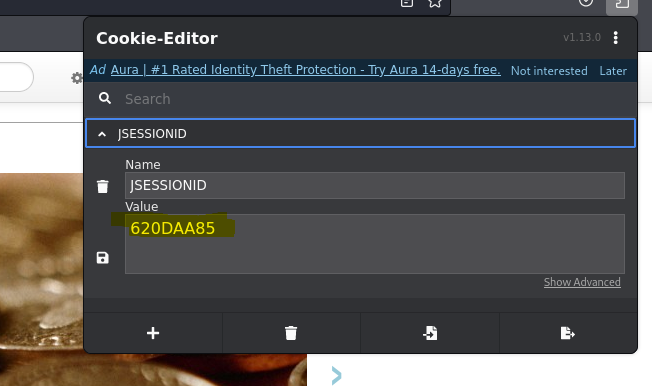
|  |  |
| --- | --- |
| **7.1 Authentication ID and password exposed** | |
| **Description** | |
| Authentication ID, or username, uniquely identifies users in systems, while passwords verify their identity. Exposing these credentials poses significant vulnerabilities. Threats include phishing attacks, data breaches, weak password practices, shoulder surfing, and insider threats. Breaches led to unauthorized access, financial losses, and reputational damage. Mitigation strategies include multifactor authentication, strong password policies, user education on phishing, and encryption for credential protection. Prompt detection and response to breaches are crucial for minimizing security risks and preserving user privacy. | |
| **Affected Resources/Parameters** | **Severity** |
| <http://zero.webappsecurity.com/login.html> | **CRITICAL** |
| **Impact/Consequences** | |
| Exposure of authentication IDs and passwords results in unauthorized access, data breaches, financial losses, and reputational damage. It enables phishing attacks, compromises sensitive information, and facilitates malicious activities. Weak password practices exacerbate vulnerabilities. Failure to address these risks jeopardizes system security, user privacy, and organizational integrity, highlighting the critical importance of safeguarding authentication credentials. | |
| **Recommendation** | |
| To mitigate risks, implement multifactor authentication, robust password policies, and regular security training. Encourage strong, unique passwords and discourage password reuse. Employ encryption for stored credentials and monitor for suspicious activity. Promptly update passwords after breaches and employ password managers for secure storage. Educate users on phishing awareness and encourage reporting of suspicious emails. Regularly audit access logs and conduct security assessments to identify and address potential vulnerabilities proactively. | |
| **Tools Used** | **References** |
| - | https://owasp.org/Top10/A07\_2021-Identification\_and\_Authentication\_Failures/ |
| **CWE** | **OWASP TOP 10 (2021)** |
| 297 | AO7: Identification and Authentication Failures |
| **Proof Of Vulnerability** | |
| **Step 1:** Open target website and go to sign up page to access the login page. | |
| **Step 2:** If you hover to the question mark icon(?) after username text field. You see the only access to the website username ‘username’ and password ‘password’. | |

|  |  |
| --- | --- |
| **7.2 Session ID unchangeable** | |
| **Description** | |
| The "Session ID Not Changing" vulnerability poses a security risk by allowing unauthorized access due to stagnant session IDs. Without periodic regeneration, attackers may exploit intercepted IDs for prolonged unauthorized entry, compromising user data and violating regulatory standards. This flaw underscores the importance of dynamic session management to safeguard authentication and user privacy. | |
| **Affected Resources/Parameters** | **Severity** |
| <http://zero.webappsecurity.com/login.html> | **HIGH** |
| **Impact/Consequences** | |
| The "Session ID Not Changing" vulnerability can lead to unauthorized access to user accounts and sensitive information. Attackers may exploit stagnant session IDs to gain prolonged entry, compromising user privacy and data integrity. This can result in financial losses, reputational damage, and legal consequences for the affected organization. Failure to address this vulnerability promptly can erode user trust and lead to regulatory non-compliance. | |
| **Recommendation** | |
| Implement a robust session management system that periodically regenerates session IDs upon authentication and after a set period of inactivity. Employ encryption techniques for session data transmission and storage to enhance security. Regularly audit and monitor session activities for suspicious behavior. Ensure compliance with relevant security standards and regulations to mitigate risks associated with vulnerability. | |
| **Extension Used** | **References** |
| Cookie editor | <https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/>  <https://cwe.mitre.org/data/definitions/613.html> |
| **CWE** | **OWASP TOP 10 (2021)** |
| 613 | AO7: Identification and Authentication Failures |
| **Proof Of Vulnerability** | |
| **Step 1:** Open target website and go to sign up page to access the login page. | |
| **Step 2:** Configure cookie editor and give desired permission for the target website. After that log in with username ‘username’ and password ‘password’. It'll generate session id. | |
| **Step 3:** After that keep this website logged in and in hibernate for several hours. | |
| **Step 4:** After that, open the website and cookie editor. You’ll see that you’re still logged in and the session ID hasn’t changed after refreshing the website either. | |

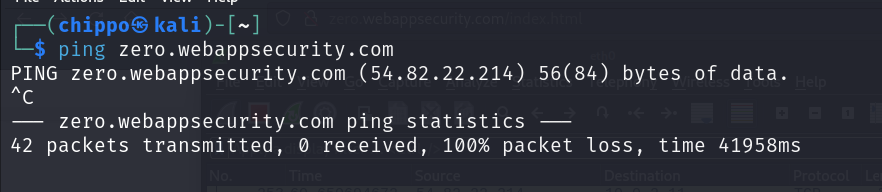
**AO8: Software and Data Integrity Failures**

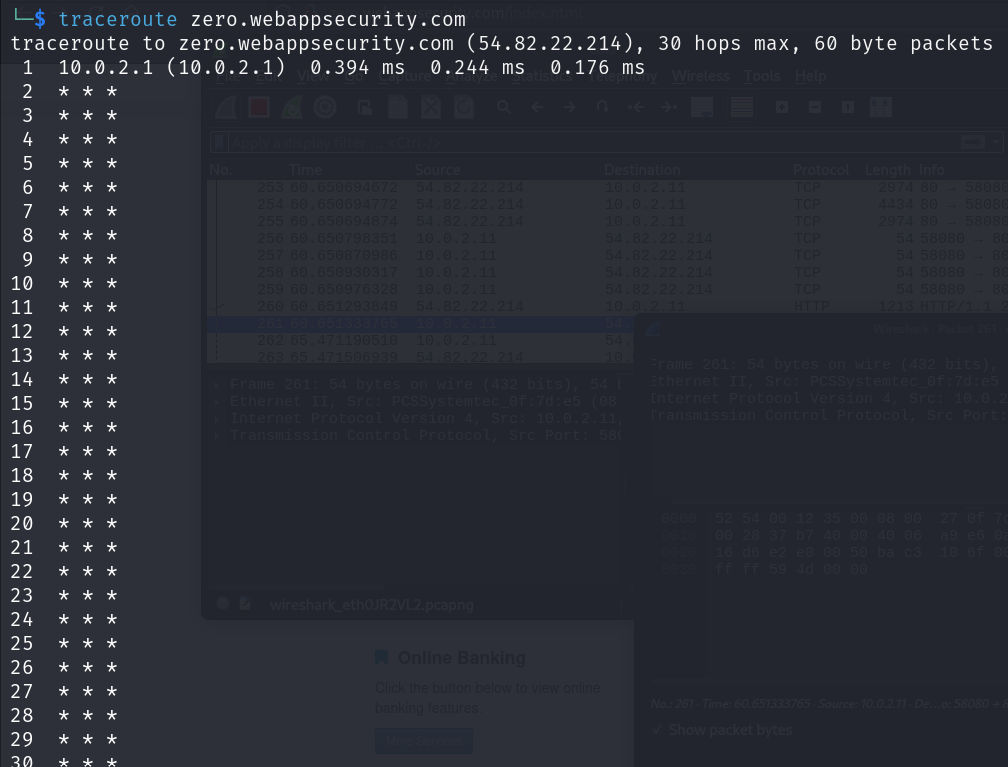
There are several vulnerabilities that fall under software and data Integrity failures which have already been found in the above testing which violates CIA triad of the targeted website.

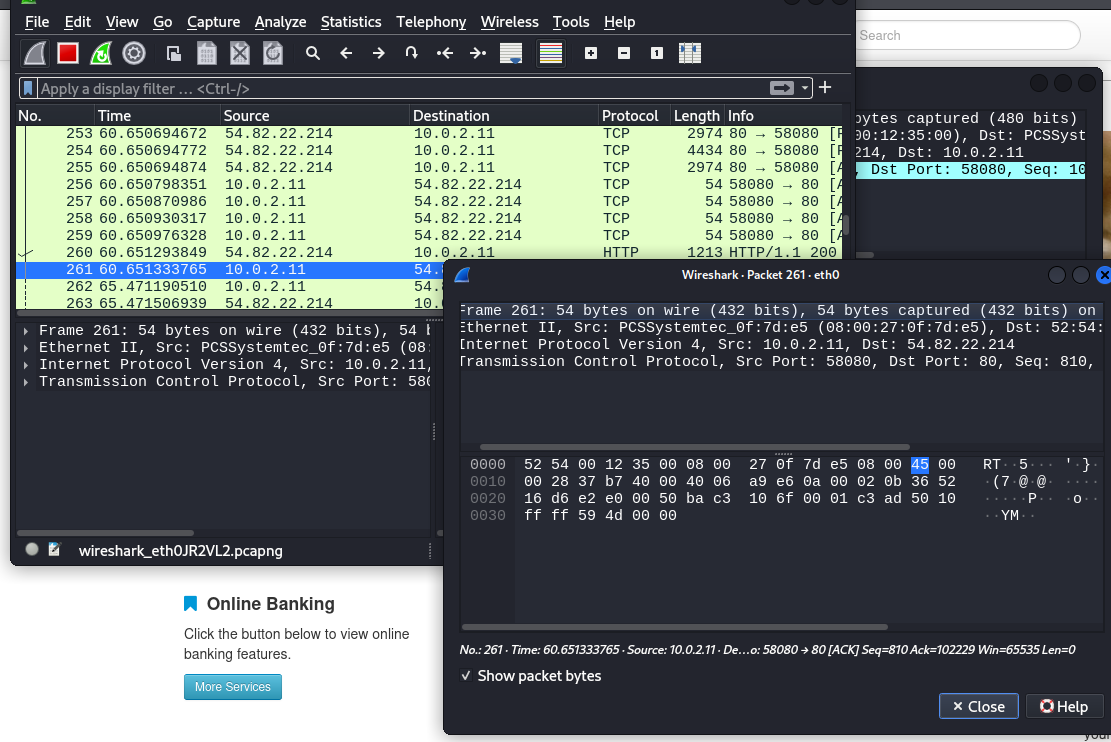
**These are the following vulnerabilities that Impact Software and Data Integrity Failures too-**

1. **IDOR -** IDOR occurs when an attacker can access or manipulate objects (such as files, records, or resources) directly without proper authorization.  
   - It undermines data integrity by allowing unauthorized access to sensitive information.  
   - For example, if an application allows users to view other users’ private data by manipulating URLs, it’s an IDOR vulnerability.  
   - **Refer to AO1: Broken access control for details for this vulnerability**  
   
2. **XSS - Outdated Components: Using outdated or vulnerable components (libraries, frameworks, plugins) poses a risk to software integrity.**   
   - These components may have known security flaws that attackers can exploit.   
   - Regularly updating components is essential to maintain integrity.  
   - **Refer to AO3: Injection for details for this vulnerability**  
   
3. **Unchanging Session IDs: If session IDs remain constant over extended periods, it can lead to session fixation attacks.**  
   - Attackers can set a known session ID and wait for users to authenticate, gaining unauthorized access.  
   - This undermines both data and session integrity.  
   - **Refer to AO7: Identification and authentication failures**   
   

**AO9: Security Logging and Monitoring Failures**

**Description-** Monitoring and logging failures can lead to compromised security, including delayed detection of threats, incomplete forensic evidence, and unauthorized access to sensitive data. Inadequate alerting, data integrity issues, and resource exhaustion further exacerbate vulnerabilities. Proper configuration, regular auditing, and proactive threat detection are essential to mitigate these risks and ensure the reliability and effectiveness of monitoring and logging systems.  
  
  
**Result-** Haven't found any issues regarding logging and monitoring failures. Here are the things I did to check -   
  
**1. Ping –** I tried to ping the website, but the website was providing null response in receiving packets. Which was suspicious at first that the server might not able to receive the packets.  


**2. Traceroute –** I tried traceroute to see the hops this website takes in order to reach to the server but it seems cause of the result that it might takes more than 30 hops which is the limit is exceeded. And was hidden.  


**3. Wireshark –** I used wireshark to monitor all the traffic going in and out of the system and it seems that there is no problem as the website is successfuly synchorinizing with the server(apache coyote7.0.7.0) and receiving the acknowledgement.  


**Conclusion – Regular auditing of monitoring is still required for overall website welfare. Cause the website some components are outdated (check vulnerable and outdated components), and requests might be vulnerable because of it.**

**AO10: SSRF(Server Side Request Forge)**

|  |  |
| --- | --- |
| **10.1 SSRF against Server** | |
| **Description** | |
| SSRF, or Server-Side Request Forgery, is a type of web security vulnerability where an attacker can manipulate a server into making unauthorized requests on its behalf. By exploiting SSRF, attackers can access internal systems, retrieve sensitive data, or perform actions as the server itself. This vulnerability arises when a server processes user-supplied URLs without proper validation, allowing attackers to control the destination of the requests, posing significant risks to system integrity and confidentiality. | |
| **Affected Resources/Parameters** | **Severity** |
| <http://zero.webappsecurity.com/bank/account-activity.html> | **HIGH** |
| **Impact/Consequences** | |
| SSRF vulnerabilities can lead to unauthorized access to sensitive data, unauthorized actions, service disruption, and reputation damage. Attackers exploit SSRF to access internal systems, steal data, disrupt services, and escalate privileges, resulting in financial losses, legal consequences, and regulatory violations. Organizations must prioritize identifying and mitigating SSRF vulnerabilities to prevent exploitation and protect their assets. | |
| **Recommendation** | |
| To mitigate SSRF vulnerabilities, implement strict input validation, limiting URLs to trusted domains and protocols. Employ network-level protections like firewalls to restrict outgoing connections to trusted IPs and ports. Ensure server configurations are secure, with proper access controls and permissions. Regularly update and patch server software to address known vulnerabilities. Conduct comprehensive security testing, including SSRF-specific assessments, and educate developers on secure coding practices to prevent SSRF exploitation. | |
| **Tool Used** | **References** |
| Burp suite | <https://cwe.mitre.org/data/definitions/918.html> |
| **CWE** | **OWASP TOP 10 (2021)** |
| 918 | AO10: SSRF |
| **Proof Of Vulnerability** | |
| **Step 1:** Open target website and go to sign up page to access the login page. | |
| **Step 2:** Open the find transaction section under account activity. | |
| **Step 3:** Open burp suite, set proxy and capture the request. | |
| **Step 3:** Delete the rest of the parameters except the currentactionID, we’ll use it to send to the server and see if that provides any information. | |
| **Step 4:** Send this to the repeater and send the response to see if server reacts and provide any information with providing just one parameter. And it does provide the information of a transaction after requesting the response in browser. That means the server didn’t authenticate for complete parameters and provided information just based on one parameter. Which leads to successful SSRF. | |

**Conclusion**

In conclusion, the VAPT report has uncovered critical vulnerabilities across various aspects of your system, aligning closely with the OWASP Top 10 for 2021. These vulnerabilities, ranging from broken access controls to injection flaws, insecure design practices, security misconfigurations, and identification/authentication failures, pose significant risks to the confidentiality, integrity, and availability of your system and its data.

Addressing these vulnerabilities promptly is imperative to mitigate the risk of security breaches and safeguard your organization's assets and reputation. By prioritizing remediation efforts based on the severity and potential impact of each vulnerability, and implementing robust security measures such as access controls, secure design practices, regular patching and updates, and proactive monitoring, you can strengthen your defenses against cyber threats and ensure the resilience of your infrastructure.

Failure to address these vulnerabilities effectively could have severe consequences, including data breaches, financial losses, and damage to your organization's reputation. Therefore, it is crucial to allocate the necessary resources and commit to implementing sustainable security practices to mitigate future risks and enhance the overall security posture of your system.