

**PUBLIC ACCESS**

# **CYBERSECURITY AUDIT REPORT**

## **Version v1.0**

*This document details the process and results of the penetration test performed by CyStack on behalf of Locker from 13/04/2022 to 30/07/2022.*

*Prepared for*

**Locker Password Manager**

*Prepared by*

**Vietnam CyStack Joint Stock Company**

**© 2022 CyStack. All rights reserved.**

Portions of this document and the templates used in its production are the property of CyStack and cannot be copied (in full or in part) without CyStack's permission.

While precautions have been taken in the preparation of this document, CyStack the publisher, and the author(s) assume no responsibility for errors, omissions, or for damages resulting from the use of the information contained herein. Use of CyStack's services does not guarantee the security of a system, or that computer intrusions will not occur.

# Contents

<b>1 Executive Summary</b>	<b>4</b>
1.1 Key Findings . . . . .	4
1.2 Limitations . . . . .	4
1.3 Assessment Components . . . . .	5
<b>2 Dashboard</b>	<b>6</b>
<b>3 Code Review Details</b>	<b>8</b>
3.1 Master Password . . . . .	8
3.2 Public Key Verification . . . . .	8
<b>4 Asset Metadata</b>	<b>10</b>
<b>5 Vulnerability Details</b>	<b>20</b>
<b>6 Appendix</b>	<b>27</b>
Appendix A - Vulnerability Severity Ratings . . . . .	27
Appendix B - Vulnerability Categories . . . . .	28
Appendix C - Security Assessment Based On OWASP Web Security Testing Guide v4.2 . . . . .	29
Appendix D - Security Assessment Based On OWASP Mobile Security Testing Guide v1.2 . . . . .	34
Mobile Application Security Requirements . . . . .	34
Resiliency against Reverse Engineering . . . . .	40

## Confidentiality Statement

This document is the exclusive property of **Locker Password Manager (Locker)** and **CyStack Vietnam Joint Stock Company (CyStack)**. This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both **Locker** and **CyStack**.

**CyStack** may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

## Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. CyStack prioritized the assessment to identify the weakest security controls an attacker would exploit. CyStack recommends Locker conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

## Version History

Version	Date	Release notes
1.0	30/07/2022	Report on security posture of Locker Password Manager on multiple platforms

## Pentesters

Fullname	Role	Email address
Nguyen Huu Trung	Head of Security	trungnh@cystack.net
Nguyen Trung Huy Son	Pentester	
Vu Hai Dang	Pentester	
Ha Minh Chau	Pentester	
Nguyen Van Huy	Pentester	
Nguyen Ba Anh Tuan	Pentester	

# Executive Summary

From 13/04/2022 to 30/07/2022, Locker engaged CyStack to evaluate the security posture of its infrastructure compared to current industry best practices. This security audit includes two parts that were conducted simultaneously. The first part was source code review, performed by CyStack, and strictly followed *OWASP Code Review Guide*. The second part was external penetration testing, conducted by CyStack. Test cases for penetration testing were based on the *NIST SP 800-115 Technical Guide to Information Security Testing and Assessment*, *OWASP Testing Guide (v4)*, and customized testing frameworks from CyStack.

CyStack's security assessment for Locker focused on evidence, which confirmed that Locker Password Manager securely functions as a password vault that DOES NOT store either Master Password or raw credential data of their users, and DOES NOT change or share any users' data without their consent. CyStack also searched for security issues that might exist in Locker, especially those related to Cryptographic Flaws, Sensitive Data Exposure and Broken Authentication/Authorization. The assessment emphasized remediation over analyzing exploitability, including issues reported by tools. This means that less time was spent determining how specific security flaws might be exploited and more time identifying as many possible security issues and associated remediation as time allowed. The audit results also included a cursory review of dependent libraries and recommendations for improving software assurance practices at Locker Password Manager.

## 1.1 Key Findings

CyStack did not find any proves that indicates vulnerable usage and storage of users' Master Password and credential data, nor any critical severity issues that would undermine the cryptography design for Locker Password Manager. CyStack identified a number of misconfigurations and broken authentication/authorization that could result in informational leakage, potential social engineering, or malicious fraud application activity. No third-party open source library dependencies were identified as being out of date. Key findings from the engagement included:

- Failure to invalidate sessions.
- Broken authentication and access control.

## 1.2 Limitations

Because of the quantity of static and dynamic analysis diagnostics, some findings were not fully analyzed during the assessment, especially, DoS or DDoS-related issues, and some security vulnerabilities in third-party open source library dependencies might have not been discovered. Some effort was redirected to propose detailed remediations to the development team to ensure that the repairs would be made before the initial release of the product.

## 1.3 Assessment Components

### Source Code Review

Source code contains the most detailed information about an application. Source code review allows security researchers to understand thoroughly how an application operates and performs. Researchers then can search for design flaws and security vulnerabilities in the application.

The safety and security assessment for application source code includes automated and manual tests. For automated tests, static code analysis tools are used to identify dead code, unsafe coding patterns and the usage of libraries or plugins with publicly known vulnerabilities. Automated tests also search for the the existence of hard-coded sensitive information such as passwords, database connection strings, private keys for third-party services, etc.

Manual tests focus on analyzing the implementation of the application's operational logic and functional components, in order to detect critical vulnerabilities, which are possibly related to user input validation, unsafe database querying, unsafe file handling, etc. or business logic flows. People who perform manual tests are security researchers.

### External Penetration Test

An external penetration test emulates the role of an attacker attempting to gain access to an internal network without internal resources or inside knowledge. A CyStack engineer attempts to gather sensitive information through open-source intelligence (OSINT), including employee information, historical breached passwords, and more that can be leveraged against external systems to gain internal network access. The engineer also performs scanning and enumeration to identify potential vulnerabilities in hopes of exploitation.

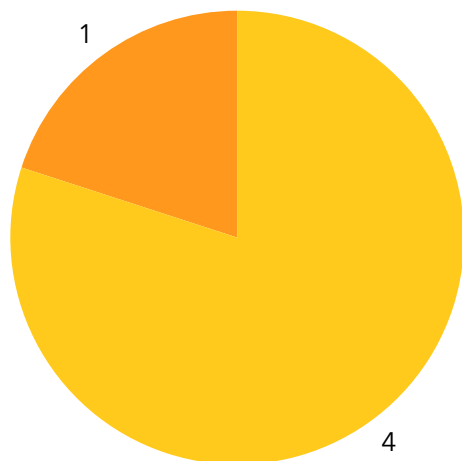
### Scope

Assessment	Details	Type
Source Code Review	<a href="#">Locker internal API</a>	Source code
Source Code Review	<a href="#">Locker Chrome extension</a>	Source code
Source Code Review	<a href="#">Locker mobile applications for Android and iOS</a>	Source code
Source Code Review	<a href="#">Locker Support website</a>	Source code
Source Code Review	<a href="#">Locker Web application</a>	Source code
External Penetration Test	*.locker.io	Websites and API
External Penetration Test	<a href="#">Locker mobile application for Android</a>	Android
External Penetration Test	<a href="#">Locker mobile application for iOS</a>	iOS
External Penetration Test	<a href="#">Locker extension for Chrome</a>	Browser extension

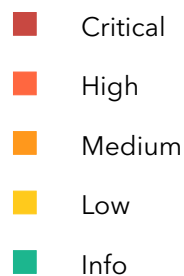
# Dashboard

Maintaining a healthy security posture requires constant review and refinement of existing security processes. Running a CyStack Pentest allows Locker's internal security team to not only uncover specific vulnerabilities but gain a better understanding of the current security threat landscape.

## Vulnerabilities by severities



## Legend



## Vulnerabilities by assets

\*.locker.io

5



## Vulnerabilities by CWE

Security Misconfiguration (CWE-16)

1













Broken Authentication And Session Management (CWE-930)

3



## Vulnerabilities by OWASP Top 10

OWASP Top 10 Category	Test result	Findings
A1 - Broken Access Control		3
A2 - Cryptographic Failures		0
A3 - Injection		0
A4 - Insecure Design		0
A5 - Security Misconfiguration		2
A6 - Vulnerable and Outdated Components		0
A7 - Identification and Authentication Failures		0
A8 - Software and Data Integrity Failures		0
A9 - Security Logging and Monitoring Failures		0
A10 - Server-Side Request Forgery		0

## Table of vulnerabilities

ID	Status	Vulnerability	Severity
#locker-002	Fixed	No rate limit for the function Forgot Password	LOW
#locker-006	Fixed	Clickjacking on Login page may lead to account takeover	LOW
#locker-007	Fixed	Designated emergency contact with View privilege can change Master Password of the linked user	MEDIUM
#locker-008	Fixed	Failure to invalidate session on Password Change	LOW
#locker-010	Fixed	Failure to invalidate Intercom chat session after Logout	LOW



# Code Review Details

CyStack found no vulnerabilities related to cryptography design during the source code review phase for Locker Password Manager.

Locker ensures the privacy and confidentiality of user credential data with the following technologies:

## 3.1 Master Password

One of the two components of [two-secret key derivation](#), [Master Password](#), is created and memorized by each Locker user. The major limitation of secret data memorized and used by humans is the fact that this data can be guessed using algorithms or automatic word guessing systems, based on a set of simple or meaningful word-number sequences (dictionary attack/rainbow table attack).

Locker implements multi-layer protection and multi-factor authentication with a high level of security, through secure algorithms such as [AES-256-CBC Encryption](#), [Hash Function](#) combined with [End-to-end Encryption](#) and [Zero-knowledge Encryption](#). However, there still exists a high risk that the Master Password becomes exposed on the Client for various reasons. Besides the user's self-disclosure, it can be hackers' common techniques such as key loggers or malware to infiltrate the Client and steal data, and complex algorithms run on supercomputers with high processing speed. Therefore, we strongly recommend that every Locker user set a strong enough Master Password with a sufficient length and complexity to resist these types of attacks.

## 3.2 Public Key Verification

During the [Organization Data Sharing](#) process, the organization member is required to send the **RSA Public Key** to the organization owner for **Org Symmetric Key** transfer.

Currently, there is no perfect method for the organization owner to verify that the **Public Key** received from a member genuinely belongs to the member with whom data should be shared. For that reason, when the Locker servers are hacked or compromised, attackers can provide a fake Public Key to the users to encrypt and successfully execute a man-in-the-middle (MITM) attack. As a result, hackers obtain the **Org Symmetric Key** for the Organization [Vault](#) and can decrypt it to access the organization's confidential data, without leaving any anomalies for the users to detect or prevent.

However, it is worth noting that the threat becomes reality only when the hackers have infiltrated and taken control of the Locker database. There are several possible solutions: the user's Public Key is authenticated by a trusted third party before being encrypted with any data, or the Public Key authentication process is directly and separately performed through a secure communication channel other than that provided by the Locker servers.

Key recommendations	
Issues	<p>After the source code review and penetration testing processes for Locker Password Manager on multiple platforms, CyStack confirmed that users' credential data are safely handled and all are encrypted before storing to Locker database. However, CyStack still discovered few vulnerabilities in Locker system. All the issues are immediately resolved right after their discovery.</p>
Recommendations	<ul style="list-style-type: none"> <li>• Review the listed vulnerabilities to anticipate where similar risks may occur, thereby deploying early remediation plans.</li> <li>• Check the dependencies and, if possible, have plans on dropping deprecated libraries and moving to those more up-to-date.</li> </ul>
References	<ul style="list-style-type: none"> <li>• <a href="https://www.appsealing.com/react-native-security/">https://www.appsealing.com/react-native-security/</a></li> <li>• <a href="https://developer.android.com/training/articles/security-tips">https://developer.android.com/training/articles/security-tips</a></li> <li>• <a href="https://developer.apple.com/library/archive/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html">https://developer.apple.com/library/archive/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html</a></li> </ul>

# Asset Metadata

## 1. locker.io

### Basic Information

Type	Website
FQDN	locker.io
IP address	172.67.70.16, 104.26.0.10, 104.26.1.10, 2606:4700:20::681a:10a, 2606:4700:20::681a:a, 2606:4700:20::ac43:4610
Operating system	Linux 2.6.18 - 2.6.22
Last location	US

### Open Ports

Port	Protocol	Service	Product	Version
80	tcp	http	Cloudflare	
443	tcp	https	Cloudflare	
2052	tcp	clearvisn		
2053	tcp	http	nginx	
2082	tcp	infowave		
2083	tcp	http	nginx	
2086	tcp	gnunet		
2087	tcp	http	nginx	
2095	tcp	nbx-ser		
2096	tcp	http	nginx	
8080	tcp	http-proxy	Cloudflare	
8443	tcp	https-alt	Cloudflare	
8880	tcp	cddbp-alt		

**Installed Applications**

<b>Program</b>	<b>Vendor</b>	<b>Installed on</b>	<b>Size</b>	<b>Version</b>	<b>URL location</b>	<b>Port</b>
Cloudflare	Cloudflare, Inc.				https://locker.io	443
Google Tag Manager	Google LLC				https://locker.io	443
Vue.js	Yuxi (Evan) You				https://locker.io	443
Nuxt.js	Nuxt Team				https://locker.io	443
Node.js	OpenJS Foundation				https://locker.io	443

## 2. support.locker.io

### Basic Information

Type	Website
FQDN	support.locker.io
IP address	104.26.1.10, 104.26.0.10, 172.67.70.16, 2606:4700:20::ac43:4610, 2606:4700:20::681a:a, 2606:4700:20::681a:10a
Operating system	Linux 2.6.18 - 2.6.22
Last location	US

### Open Ports

Port	Protocol	Service	Product	Version
80	tcp	http	Cloudflare	
443	tcp	https	Cloudflare	
2052	tcp	clearvisn		
2053	tcp	http	nginx	
2082	tcp	infowave		
2083	tcp	http	nginx	
2086	tcp	gnunet		
2087	tcp	http	nginx	
2095	tcp	nbx-ser		
2096	tcp	http	nginx	
8080	tcp	http-proxy	Cloudflare	
8443	tcp	https-alt	Cloudflare	
8880	tcp	cddbp-alt		

**Installed Applications**

<b>Program</b>	<b>Vendor</b>	<b>Installed on</b>	<b>Size</b>	<b>Version</b>	<b>URL location</b>	<b>Port</b>
Cloudflare	Cloudflare, Inc.				<a href="https://support.locker.io">https://support.locker.io</a>	443
Google Tag Manager	Google LLC				<a href="https://support.locker.io">https://support.locker.io</a>	443
Vue.js	Yuxi (Evan) You				<a href="https://support.locker.io">https://support.locker.io</a>	443
Nuxt.js	Nuxt Team				<a href="https://support.locker.io">https://support.locker.io</a>	443
Node.js	OpenJS Foundation				<a href="https://support.locker.io">https://support.locker.io</a>	443

### 3. Locker mobile application for Android

#### Basic Information

<b>Type</b>	Mobile Application
<b>Operating system</b>	Android
<b>File name</b>	com.cystack.locker.apk
<b>Size</b>	20.63 MB
<b>MD5</b>	d25eb7f1d5cffa38473c73912baf5838
<b>SHA1</b>	ef8021f0d2b43123e1fcd6fde6e54105ff61a5df
<b>SHA256</b>	3b4c6873bd10dfcb51163bb6583c5af14ce508e98ffe4f1bf285fa014299b8db
<b>Application name</b>	Locker
<b>Package name</b>	com.cystack.locker
<b>Main activity</b>	com.cystack.locker.MainActivity
<b>Target SDK</b>	31
<b>Min SDK</b>	21
<b>Android version name</b>	1.23
<b>Android version code</b>	4070004

**Application certificates**

<b>v1 signature</b>	True
<b>v2 signature</b>	True
<b>v3 signature</b>	True
<b>Subject</b>	C=US, ST=California, L=Mountain View, O=Google Inc., OU=Android, CN=Android
<b>Signature Algorithm</b>	rsassa_pkcs1v15
<b>Valid From</b>	2021-09-04 04:00:44+00:00
<b>Valid To</b>	2051-09-04 04:00:44+00:00
<b>Issuer</b>	C=US, ST=California, L=Mountain View, O=Google Inc., OU=Android, CN=Android
<b>Serial Number</b>	0xde63e72d0cbbcd870cb06cd2ca9612d07df8c6d9
<b>Hash Algorithm</b>	sha256
<b>MD5</b>	310b4ac87ef3515f02556726da7e7e76
<b>SHA1</b>	23d4880dbf7c4aaedd5f61669b2a35e5510ef0db
<b>SHA256</b>	4e771c44034a71a726baf79e6351f045c8574d12a7489f8cb737cac9e1017db7
<b>SHA512</b>	fcf4231cf1fb3e2d8532048744d8233036ed3a5f4c48242a860773f8e08fcfe76d7ef424e6cd18db52974e59474f086e19ba584d2ac0cbc4e7f65ba558f2ff97
<b>Public Key Algorithm</b>	rsa
<b>Bit Size</b>	4096
<b>Fingerprint</b>	1e48464064b9820208858d45f5d282137626386810db1927f584118d7628fecb



## Application permissions

Permission	Severity	Definition	Description
android.permission.CAMERA	Dangerous	Take pictures and videos	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.
android.permission.READ_EXTERNAL_STORAGE	Dangerous	Read external storage contents	Allows an application to read from external storage.
android.permission.SYSTEM_ALERT_WINDOW	Dangerous	Display system-level alerts	Allows an application to show system-alert windows. Malicious applications can take over the entire screen of the phone.
android.permission.WRITE_EXTERNAL_STORAGE	Dangerous	Read/modify/delete external storage contents	Allows an application to write to external storage.
android.permission.ACCESS_NETWORK_STATE	Normal	View network status	Allows an application to view the status of all networks.
android.permission.ACCESS_WIFI_STATE	Normal	View Wi-Fi status	Allows an application to view the information about the status of Wi-Fi.
android.permission.CHANGE_WIFI_MULTICAST_STATE	Normal	Allow Wi-Fi Multicast reception	Allows an application to receive packets not directly addressed to your device. This can be useful when discovering services offered nearby. It uses more power than the non-multicast mode.
android.permission.FOREGROUND_SERVICE	Normal		Allows a regular application to use Service.startForeground.
android.permission.INTERNET	Normal	Full Internet access	Allows an application to create network sockets.

android.permission.RECEIVE_BOOT_COMPLETED	Normal	Automatically start at boot	Allows an application to start itself as soon as the system has finished booting. This can make it take longer to start the phone and allow the application to slow down the overall phone by always running.
android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS	Normal		Permission an application must hold in order to use Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS.
android.permission.SCHEDULE_EXACT_ALARM	Normal		Allows an app to use exact alarm scheduling APIs to perform timing sensitive background work.
android.permission.USE_BIOMETRIC	Normal		Allows an app to use device supported biometric modalities.
android.permission.USE_FINGERPRINT	Normal	Allow use of fingerprint	This constant was deprecated in API level 28. Applications should request USE_BIOMETRIC instead.
android.permission.USE_FULL_SCREEN_INTENT	Normal		Required for apps targeting Build.VERSION_CODES.Q that want to use notification full screen intents.
android.permission.VIBRATE	Normal	Control vibrator	Allows the application to control the vibrator.
android.permission.WAKE_LOCK	Normal	Prevent phone from sleeping	Allows an application to prevent the phone from going to sleep.
com.google.android.c2dm.permission.RECEIVE	Signature	C2DM permissions	Permission for cloud to device messaging.
android.permission.INSTALL_GRANT_RUNTIME_PERMISSIONS	Unknown	Unknown permission	Unknown permission from android reference.
com.android.vending.BILLING	Unknown	Unknown permission	Unknown permission from android reference.

com.google.android.finsky.permission.BIND_GET_INST ALL_REFERRER_SERVICE	Unknown	Unknown permission	Unknown permission from android reference.
com.google.android.gms.p ermission.AD_ID	Unknown	Unknown permission	Unknown permission from android reference.

## 4. Locker mobile application for iOS

### Basic Information

<b>Type</b>	Mobile Application
<b>Operating system</b>	iOS
<b>Filename</b>	Locker 1.23.ipa
<b>Size</b>	27.27 MB
<b>MD5</b>	7aec89e7291cb1606c578d57b9bdb0d
<b>SHA1</b>	d1f9d2567c2e29549251d5b9f5423719aab095de
<b>SHA256</b>	10ad5ff98fc3a588c5dcfb68fe9387a2c9edb6859bc57a1452bd31aae31806e4
<b>Application name</b>	Locker
<b>Application type</b>	Swift
<b>Identifier</b>	com.cystack.lockerapp
<b>SDK name</b>	iphoneos15.2
<b>Application version</b>	1.23
<b>Build</b>	72
<b>Platform version</b>	15.2

### Application permissions

Permission	Severity	Definition	Description
NSCameraUsageDescription	Dangerous	Access camera	Scan QR code
NSLocationWhenInUseUsageDescription	Dangerous	Access location information when app is in the foreground	
NSPhotoLibraryUsageDescription	Dangerous	Access the user's photo library	Send photos to support center
NSFaceIDUsageDescription	Normal	Access the ability to authenticate with Face ID	Authentication

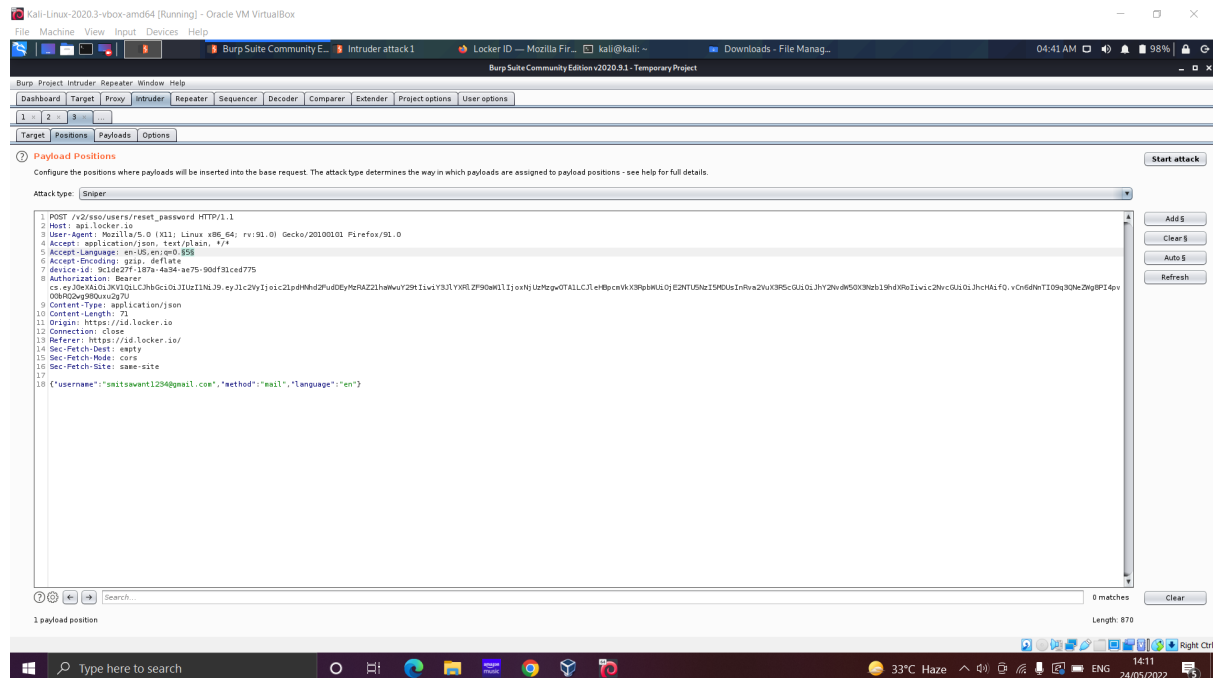
# Vulnerability Details

## 1. No rate limit for the function Forgot Password

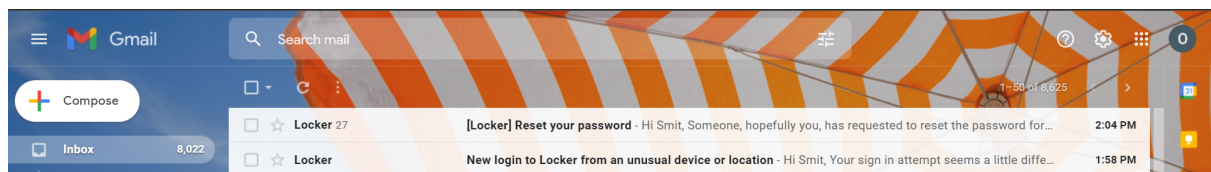
<b>ID</b>	#locker-002
<b>Category</b>	Security Misconfiguration
<b>Description</b>	<p>A rate limiting algorithm is used to check if the user session (or IP-address) has to be limited based on the information in the session cache. In case a client made too many requests within a given timeframe, HTTP-Servers may respond with an error status code, usually 429: Too Many Requests. When Forgetting Password for an account in Locker Password Manager, it is detected that this request has no mechanism of rate limit. This type of attack can result in financial loss by exceeding allowed number of sent requests supported by the mail service, and also, it can slow down running services. It might cause bulk of storage in sent mail and cause users' annoyance.</p>
<b>Severity</b>	<span>LOW</span>
<b>CVSS 3.0 base score</b>	CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:L (3.7)
<b>Target</b>	<a href="https://id.locker.io/forgot?SERVICE_URL=%2Fvault&amp;SERVICE_SCOPE=pwdmanager&amp;lang=en">https://id.locker.io/forgot?SERVICE_URL=%2Fvault&amp;SERVICE_SCOPE=pwdmanager&amp;lang=en</a>
<b>Status</b>	Fixed
<b>Reference</b>	OWASP A5 - Security Misconfiguration
<b>Remediation</b>	<p>Limit with either ReCaptcha or any mechanism of manual human interaction or limit the number of allowed times to request Forgot Password, for example, 5 times/day.</p>

## Step to reproduce

1. Go to the link <https://locker.io/>, then click on the option Login.
2. Select Forgot Password, then enter the registered email and send the request.
3. Once done, intercept the request with Burp Suite and forward every intercepted except until a request with the following form is found: {"email": "your email here", [...]}.
  - 4. Send this request to Intruder and repeat it 100 times by iterating any arbitrary payload on to a place that does not affect request. For example, the value of  $q$  in the header Accept-Language:



5. Every request is responded with the status code 200, and all of them can be found in the mail box.



## 2. Clickjacking on Login page may lead to account takeover

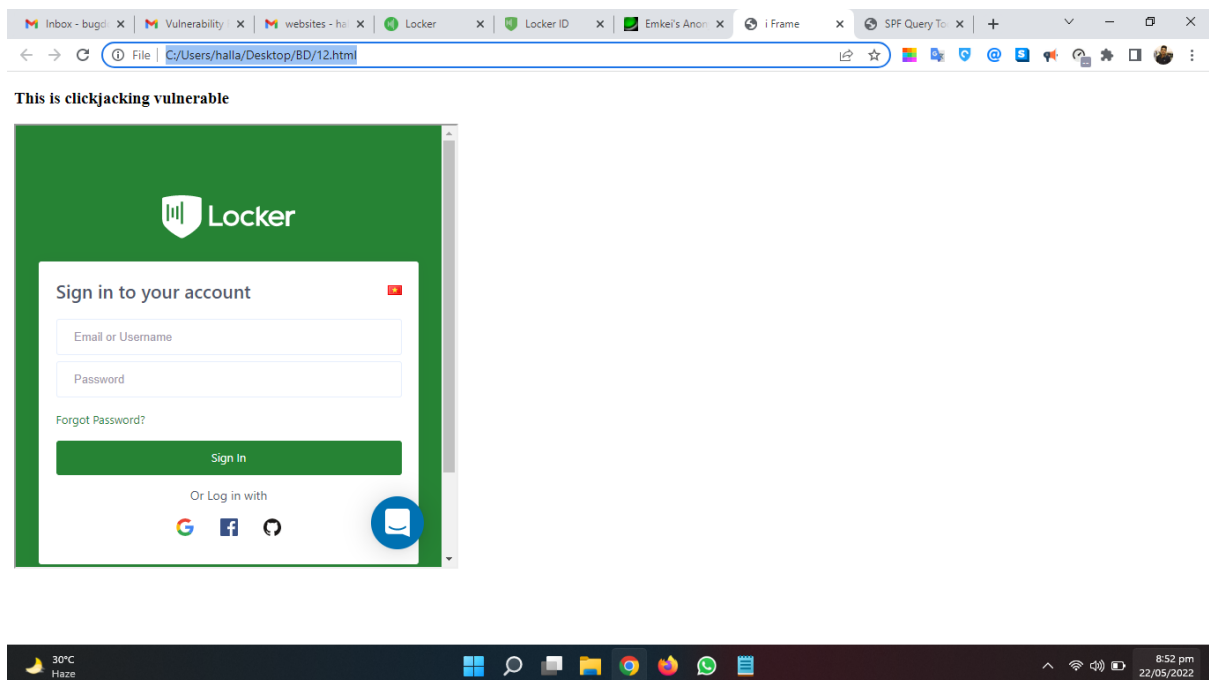
<b>ID</b>	#locker-006
<b>Category</b>	Security Misconfiguration
<b>Description</b>	Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages. The server of Locker didn't return an X-Frame-Options header. This means that this website could be at risk of a clickjacking attack. Sites can use X-Frame-Options to avoid clickjacking attacks, by ensuring that their content is not embedded into other sites within any <frame> or <iframe>.
<b>Severity</b>	LOW
<b>CVSS 3.0 base score</b>	CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N (3.1)
<b>Target</b>	https://id.locker.io/login?SERVICE_URL=%2Fvault&SERVICE_SCOPE=pwdmanager&lang=en
<b>Status</b>	Fixed
<b>Reference</b>	OWASP A5 - Security Misconfiguration
<b>Remediation</b>	<ul style="list-style-type: none"> <li>• Add the header X-Frame-Options or implement other mechanism to restrict &lt;frame&gt; and &lt;iframe&gt; usage to render websites of Locker on cross domain sites.</li> <li>• Use CAPTCHA or element randomization.</li> </ul>

## Step to reproduce

- Prepare a malicious page that use <iframe> to embed Locker website inside it:

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta charset="UTF-8">
<title>i Frame</title>
</head>
<body>
<h3>This is clickjacking vulnerable</h3>
<iframe src="https://id.locker.io/login?SERVICE_URL=%2Fvault&SERVICE_SCOPE=pwdmanager&lang=en" height="500px" width="500px"></iframe>
</body>
</html>
```

- Open the created webpage and see the result:





### 3. Designated emergency contact with View privilege can change Master Password of the linked user

<b>ID</b>	#locker-007
<b>Category</b>	Broken Authentication And Session Management
<b>Description</b>	A privilege escalation vulnerability exists in Locker applications, by which attackers can leverage to change the Master Password of victims. However, attackers must be authenticated as a designated emergency contact of the victims, in order to successfully conduct the attack.
<b>Severity</b>	MEDIUM
<b>CVSS 3.0 base score</b>	CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:N (6.4)
<b>Target</b>	https://api.locker.io/v3/cystack_platform/pm/emergency_access/3912603e-e626-4925-981f-69a9f04d3xxx/password
<b>Status</b>	Fixed
<b>Reference</b>	OWASP A1 - Broken Access Control
<b>Remediation</b>	Review the permission control table and apply correct check on each role for the function Change Master Password of shared assets.

#### Step to reproduce

1. Create 2 accounts: one will act as the victim, and the other will be attacker.
2. Log in Locker with the victim account.
3. Catch the request which is used for Changing Master Password with Burp Suite, and save it for later use. The request has the following structure:

```
POST /v3/cystack_platform/pm/emergency_access/3912603e-e626-4925-981f-69a9f04d3xxx
/password HTTP/2
Host: api.locker.io
Content-Length: 264
Content-Type: application/json
Authorization: Bearer xxxxxxxx

{"key": "2.5kdxLiGgBmAflMSVtzerDA==|J1D8zfbRRPsadv28tNDOLaUKr/7Uv7a184Mypg
/Bzi37zM1sBLEYmF1VI40ujeVgFIi4Y7Y1z22wGE6aaTFd63FCR5XfK3gEkxeQk0aFao=|ebgaJ8fEwVRKfRxs
wO
/sZ8WRjmTrF4pVRBqMwq5Y30M=", "new_master_password_hash": "x6CYc+m7xyKK0CdZV22iYjxj3X1kBbI
1WA906AMWzbA="}
```

4. Add the other account as Trusted emergency contact with View privilege.
5. Log in to that account in other browser or in private tab.

6. Approve to become the Designated emergency account for the victim.
7. Reuse the request that has been saved, but apply with the attacker authorization token instead.
8. Response return with status code 200 and no error message.
9. Check on the victim account by re-login or refresh the page. Enter the original Master Password, the victim cannot access to his own vault.

For more details on exploitation, please login WhiteHub and follow [this link](#).

## 4. Failure to invalidate session on Password Change

<b>ID</b>	#locker-008
<b>Category</b>	Broken Authentication And Session Management
<b>Description</b>	Locker application does not invalidate users' session after successful requests of Password Change. With this design, the victim cannot revoke access of any attackers if account has been compromised. If attacker have user password and logged in different places, as other sessions are not destroyed after password changes, the attackers still can log in and have a complete access to the victim account until these sessions are expired.
<b>Severity</b>	LOW
<b>CVSS 3.0 base score</b>	CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N (3.1)
<b>Target</b>	https://id.locker.io/
<b>Status</b>	Fixed
<b>Reference</b>	OWASP A1 - Broken Access Control
<b>Remediation</b>	Blacklist or destroy authorization tokens of users after they change their passwords.

### Step to reproduce

1. Create an account and go to My profile.
2. Change password in the current browser.
3. Log in this account in other browser and edit any data.
4. Save change, the result shows that the new data has been saved and changed.

For more details on exploitation, please login WhiteHub and follow [this link](#).

## 5. Failure to invalidate Intercom chat session after Logout

<b>ID</b>	#locker-010
<b>Category</b>	Broken Authentication And Session Management
<b>Description</b>	Locker application does not invalidate users' Intercom chat session after Logout.
<b>Severity</b>	LOW
<b>CVSS 3.0 base score</b>	CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:U/C:L/I:N/A:N (3.1)
<b>Target</b>	https://locker.io/
<b>Status</b>	Fixed
<b>Reference</b>	OWASP A1 - Broken Access Control
<b>Remediation</b>	No cookies data from the last session should be accessible after logging out.

### Step to reproduce

1. Log in to an account at <https://locker.io/>.
2. Copy the session cookies from the export button on the extension. Attackers can use any Cookie Grabber to grab the cookies from users, depending on the attack scenario.
3. Log out from the account.
4. Load the page <https://id.locker.io/login>.
5. Delete all the related cookies from the browser, then add the old cookies from the extension with the import button.
6. Go to the page <https://id.locker.io/login>. The page will take the user back to the Login page with a little difference: it contains the same chat in the chat screen (click on the bottom right button).

For more details on exploitation, please login WhiteHub and follow [this link](#).

# Appendix

## Appendix A - Vulnerability Severity Ratings

Severity	CVSS 3.0 score range	Definition
<b>CRITICAL</b>	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
<b>HIGH</b>	7.0-8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
<b>MEDIUM</b>	4.0-6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
<b>LOW</b>	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
<b>INFO</b>	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

## Appendix B - Vulnerability Categories

CyStack uses CWE (Common Weakness Enumeration) for the vulnerability categorization. Common Weakness Enumeration (CWE) is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

CWE categories used by CyStack are listed in the following table:

CWE ID	Name
CWE-16	Security Misconfiguration
CWE-77, CWE-259	Insecure OS Firmware
CWE-79	Cross-Site Scripting (XSS)
CWE-310	Broken Cryptography
CWE-311, CWE-319	Insecure Data Transport
CWE-352	Cross-Site Request Forgery (CSRF)
CWE-359	Privacy Concerns
CWE-400	Application Level Denial Of Service (DoS)
CWE-601	Unvalidated Redirects And Forwards
CWE-693	Lack Of Binary Hardening
CWE-723	Broken Access Control
CWE-729, CWE-922	Insecure Data Storage
CWE-919	Mobile Security Misconfiguration
CWE-929	Injection
CWE-930	Broken Authentication And Session Management
CWE-934	Sensitive Data Exposure
CWE-937	Using Components With Known Vulnerabilities

## Appendix C - Security Assessment Based On OWASP Web Security Testing Guide v4.2

Test ID	Test name	Status
<b>WSTG-INFO</b>	<b>Information Gathering</b>	<b>Pass</b>
WSTG-INFO-01	Conduct Search Engine Discovery and Reconnaissance for Information Leakage	Pass
WSTG-INFO-02	Fingerprint Web Server	Pass
WSTG-INFO-03	Review Webserver Metafiles for Information Leakage	Pass
WSTG-INFO-04	Enumerate Applications on Webserver	Pass
WSTG-INFO-05	Review Webpage Content for Information Leakage	Pass
WSTG-INFO-06	Identify Application Entry Points	Pass
WSTG-INFO-07	Map Execution Paths Through Application	Pass
WSTG-INFO-08	Fingerprint Web Application Framework	Pass
WSTG-INFO-09	Fingerprint Web Application	Pass
WSTG-INFO-10	Map Application Architecture	Pass
<b>WSTG-CONF</b>	<b>Configuration and Deploy Management Testing</b>	<b>Pass</b>
WSTG-CONF-01	Test Network Infrastructure Configuration	Pass
WSTG-CONF-02	Test Application Platform Configuration	Pass
WSTG-CONF-03	Test File Extensions Handling for Sensitive Information	Pass
WSTG-CONF-04	Review Old Backup and Unreferenced Files for Sensitive Information	Pass
WSTG-CONF-05	Enumerate Infrastructure and Application Admin Interfaces	Pass
WSTG-CONF-06	Test HTTP Methods	Pass
WSTG-CONF-07	Test HTTP Strict Transport Security	Pass
WSTG-CONF-08	Test RIA Cross Domain Policy	Pass
WSTG-CONF-09	Test File Permission	Pass
WSTG-CONF-10	Test for Subdomain Takeover	Pass
WSTG-CONF-11	Test Cloud Storage	Pass

WSTG-CONF-12	Testing for Content Security Policy	Pass
<b>WSTG-IDNT</b>	<b>Identity Management Testing</b>	<b>Pass</b>
WSTG-IDNT-01	Test Role Definitions	Pass
WSTG-IDNT-02	Test User Registration Process	Pass
WSTG-IDNT-03	Test Account Provisioning Process	Pass
WSTG-IDNT-04	Testing for Account Enumeration and Guessable User Account	Pass
WSTG-IDNT-05	Testing for Weak or Unenforced Username Policy	Pass
<b>WSTG-ATHN</b>	<b>Authentication Testing</b>	<b>Pass</b>
WSTG-ATHN-01	Testing for Credentials Transported over an Encrypted Channel	Pass
WSTG-ATHN-02	Testing for Default Credentials	Pass
WSTG-ATHN-03	Testing for Weak Lock Out Mechanism	Pass
WSTG-ATHN-04	Testing for Bypassing Authentication Schema	Pass
WSTG-ATHN-05	Testing for Vulnerable Remember Password	Pass
WSTG-ATHN-06	Testing for Browser Cache Weakness	Pass
WSTG-ATHN-07	Testing for Weak Password Policy	Pass
WSTG-ATHN-08	Testing for Weak Security Question Answer	Pass
WSTG-ATHN-09	Testing for Weak Password Change or Reset Functionalities	Pass
WSTG-ATHN-10	Testing for Weaker Authentication in Alternative Channel	Pass
<b>WSTG-ATHZ</b>	<b>Authorization Testing</b>	<b>Pass</b>
WSTG-ATHZ-01	Testing Directory Traversal File Include	Pass
WSTG-ATHZ-02	Testing for Bypassing Authorization Schema	Pass
WSTG-ATHZ-03	Testing for Privilege Escalation	Pass
WSTG-ATHZ-04	Testing for Insecure Direct Object References	Pass
<b>WSTG-SESS</b>	<b>Session Management Testing</b>	<b>Pass</b>
WSTG-SESS-01	Testing for Session Management Schema	Pass
WSTG-SESS-02	Testing for Cookies Attributes	Pass
WSTG-SESS-03	Testing for Session Fixation	Pass

WSTG-SESS-04	Testing for Exposed Session Variables	Pass
WSTG-SESS-05	Testing for Cross Site Request Forgery	Pass
WSTG-SESS-06	Testing for Logout Functionality	Pass
WSTG-SESS-07	Testing Session Timeout	Pass
WSTG-SESS-08	Testing for Session Puzzling	Pass
WSTG-SESS-09	Testing for Session Hijacking	Pass
WSTG-SESS-10	Testing JSON Web Tokens	Pass
<b>WSTG-INPV</b>	<b>Input Validation Testing</b>	<b>Pass</b>
WSTG-INPV-01	Testing for Reflected Cross Site Scripting	Pass
WSTG-INPV-02	Testing for Stored Cross Site Scripting	Pass
WSTG-INPV-03	Testing for HTTP Verb Tampering	Pass
WSTG-INPV-04	Testing for HTTP Parameter pollution	Pass
WSTG-INPV-05	Testing for SQL Injection	Pass
WSTG-INPV-06	Testing for LDAP Injection	Pass
WSTG-INPV-07	Testing for XML Injection	Pass
WSTG-INPV-08	Testing for SSI Injection	Pass
WSTG-INPV-09	Testing for XPath Injection	Pass
WSTG-INPV-10	Testing for IMAP SMTP Injection	Pass
WSTG-INPV-11	Testing for Code Injection	Pass
WSTG-INPV-12	Testing for Command Injection	Pass
WSTG-INPV-13	Testing for Format String Injection	Pass
WSTG-INPV-14	Testing for Incubated Vulnerabilities	Pass
WSTG-INPV-15	Testing for HTTP Splitting Smuggling	Pass
WSTG-INPV-16	Testing for HTTP Incoming Requests	Pass
WSTG-INPV-17	Testing for Host Header Injection	Pass
WSTG-INPV-18	Testing for Server-side Template Injection	Pass
WSTG-INPV-19	Testing for Server-Side Request Forgery	Pass
<b>WSTG-ERRH</b>	<b>Error Handling</b>	<b>Pass</b>



WSTG-ERRH-01	Testing for Improper Error Handling	Pass
WSTG-ERRH-02	Testing for Stack Traces	Pass
<b>WSTG-CRYP</b>	<b>Cryptography</b>	<b>Pass</b>
WSTG-CRYP-01	Testing for Weak Transport Layer Security	Pass
WSTG-CRYP-02	Testing for Padding Oracle	Pass
WSTG-CRYP-03	Testing for Sensitive Information Sent Via Unencrypted Channels	Pass
WSTG-CRYP-04	Testing for Weak Encryption	Pass
<b>WSTG-BUSLOGIC</b>	<b>Business Logic Testing</b>	<b>Pass</b>
WSTG-BUSL-01	Test Business Logic Data Validation	Pass
WSTG-BUSL-02	Test Ability to Forge Requests	Pass
WSTG-BUSL-03	Test Integrity Checks	Pass
WSTG-BUSL-04	Test for Process Timing	Pass
WSTG-BUSL-05	Test Number of Times a Function Can be Used Limits	Pass
WSTG-BUSL-06	Testing for the Circumvention of Work Flows	Pass
WSTG-BUSL-07	Test Defenses Against Application Misuse	Pass
WSTG-BUSL-08	Test Upload of Unexpected File Types	Pass
WSTG-BUSL-09	Test Upload of Malicious Files	Pass
<b>WSTG-CLIENT</b>	<b>Client-side Testing</b>	<b>Pass</b>
WSTG-CLNT-01	Testing for DOM based Cross Site Scripting	Pass
WSTG-CLNT-02	Testing for JavaScript Execution	Pass
WSTG-CLNT-03	Testing for HTML Injection	Pass
WSTG-CLNT-04	Testing for Client-side URL Redirect	Pass
WSTG-CLNT-05	Testing for CSS Injection	Pass
WSTG-CLNT-06	Testing for Client-side Resource Manipulation	Pass
WSTG-CLNT-07	Test Cross Origin Resource Sharing	Pass
WSTG-CLNT-08	Testing for Cross Site Flashing	Pass
WSTG-CLNT-09	Testing for Clickjacking	Pass
WSTG-CLNT-10	Testing WebSockets	Pass

WSTG-CLNT-11	Test Web Messaging	Pass
WSTG-CLNT-12	Test Browser Storage	Pass
WSTG-CLNT-13	Testing for Cross Site Script Inclusion	Pass
<b>WSTG-APIT</b>	<b>API Testing</b>	<b>Pass</b>
WSTG-APIT-01	Testing GraphQL	Pass

**LEGEND**

**Pass:** Requirement is applicable to Web application and implemented according to best practices.

**Fail:** Requirement is applicable to Web application but not fulfilled.

## Appendix D - Security Assessment Based On OWASP Mobile Security Testing Guide v1.2

### Mobile Application Security Requirements

Test ID	Test name	Status
	<b>Architecture, design and threat modelling</b>	
MSTG-ARCH-1	All app components are identified and known to be needed.	Pass
MSTG-ARCH-2	Security controls are never enforced only on the client side, but on the respective remote endpoints.	Pass
MSTG-ARCH-3	A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture.	Pass
MSTG-ARCH-4	Data considered sensitive in the context of the mobile app is clearly identified.	Pass
MSTG-ARCH-5	All app components are defined in terms of the business functions and/or security functions they provide.	Pass
MSTG-ARCH-6	A threat model for the mobile app and the associated remote services has been produced that identifies potential threats and countermeasures.	Pass
MSTG-ARCH-7	All security controls have a centralized implementation.	Pass
MSTG-ARCH-8	There is an explicit policy for how cryptographic keys (if any) are managed, and the lifecycle of cryptographic keys is enforced. Ideally, follow a key management standard such as NIST SP 800-57.	Pass
MSTG-ARCH-9	A mechanism for enforcing updates of the mobile app exists.	Pass
MSTG-ARCH-10	Security is addressed within all parts of the software development lifecycle.	Pass
MSTG-ARCH-11	A responsible disclosure policy is in place and effectively applied.	Pass
MSTG-ARCH-12	The app should comply with privacy laws and regulations.	Pass
	<b>Data Storage and Privacy</b>	

MSTG-STORAGE-1	System credential storage facilities need to be used to store sensitive data, such as PII, user credentials or cryptographic keys.	Pass
MSTG-STORAGE-2	No sensitive data should be stored outside of the app container or system credential storage facilities.	Pass
MSTG-STORAGE-3	No sensitive data is written to application logs.	Pass
MSTG-STORAGE-4	No sensitive data is shared with third parties unless it is a necessary part of the architecture.	Pass
MSTG-STORAGE-5	The keyboard cache is disabled on text inputs that process sensitive data.	Pass
MSTG-STORAGE-6	No sensitive data is exposed via IPC mechanisms.	Pass
MSTG-STORAGE-7	No sensitive data, such as passwords or pins, is exposed through the user interface.	Pass
MSTG-STORAGE-8	No sensitive data is included in backups generated by the mobile operating system.	Pass
MSTG-STORAGE-9	The app removes sensitive data from views when moved to the background.	Pass
MSTG-STORAGE-10	The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use.	Pass
MSTG-STORAGE-11	The app enforces a minimum device-access-security policy, such as requiring the user to set a device passcode.	Pass
MSTG-STORAGE-12	The app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app.	Pass
MSTG-STORAGE-13	No sensitive data should be stored locally on the mobile device. Instead, data should be retrieved from a remote endpoint when needed and only be kept in memory.	Pass
MSTG-STORAGE-14	If sensitive data is still required to be stored locally, it should be encrypted using a key derived from hardware backed storage which requires authentication.	Pass
MSTG-STORAGE-15	The app's local storage should be wiped after an excessive number of failed authentication attempts.	Pass
	<b>Cryptography</b>	

MSTG-CRYPTO-1	The app does not rely on symmetric cryptography with hardcoded keys as a sole method of encryption.	Pass
MSTG-CRYPTO-2	The app uses proven implementations of cryptographic primitives.	Pass
MSTG-CRYPTO-3	The app uses cryptographic primitives that are appropriate for the particular use-case, configured with parameters that adhere to industry best practices.	Pass
MSTG-CRYPTO-4	The app does not use cryptographic protocols or algorithms that are widely considered deprecated for security purposes.	Pass
MSTG-CRYPTO-5	The app doesn't re-use the same cryptographic key for multiple purposes.	Pass
MSTG-CRYPTO-6	All random values are generated using a sufficiently secure random number generator.	Pass
	<b>Authentication and Session Management</b>	
MSTG-AUTH-1	If the app provides users access to a remote service, some form of authentication, such as username/password authentication, is performed at the remote endpoint.	Pass
MSTG-AUTH-2	If stateful session management is used, the remote endpoint uses randomly generated session identifiers to authenticate client requests without sending the user's credentials.	Pass
MSTG-AUTH-3	If stateless token-based authentication is used, the server provides a token that has been signed using a secure algorithm.	Pass
MSTG-AUTH-4	The remote endpoint terminates the existing session when the user logs out.	Pass
MSTG-AUTH-5	A password policy exists and is enforced at the remote endpoint.	Pass
MSTG-AUTH-6	The remote endpoint implements a mechanism to protect against the submission of credentials an excessive number of times.	Pass
MSTG-AUTH-7	Sessions are invalidated at the remote endpoint after a predefined period of inactivity and access tokens expire.	Pass

MSTG-AUTH-8	Biometric authentication, if any, is not event-bound (i.e. using an API that simply returns "true" or "false"). Instead, it is based on unlocking the keychain/keystore.	Pass
MSTG-AUTH-9	A second factor of authentication exists at the remote endpoint and the 2FA requirement is consistently enforced.	Pass
MSTG-AUTH-10	Sensitive transactions require step-up authentication.	Pass
MSTG-AUTH-11	The app informs the user of all sensitive activities with their account. Users are able to view a list of devices, view contextual information (IP address, location, etc.), and to block specific devices.	Pass
MSTG-AUTH-12	Authorization models should be defined and enforced at the remote endpoint.	Pass
	<b>Network Communication</b>	
MSTG-NETWORK-1	Data is encrypted on the network using TLS. The secure channel is used consistently throughout the app.	Pass
MSTG-NETWORK-2	The TLS settings are in line with current best practices, or as close as possible if the mobile operating system does not support the recommended standards.	Pass
MSTG-NETWORK-3	The app verifies the X.509 certificate of the remote endpoint when the secure channel is established. Only certificates signed by a trusted CA are accepted.	Pass
MSTG-NETWORK-4	The app either uses its own certificate store, or pins the endpoint certificate or public key, and subsequently does not establish connections with endpoints that offer a different certificate or key, even if signed by a trusted CA.	Pass
MSTG-NETWORK-5	The app doesn't rely on a single insecure communication channel (email or SMS) for critical operations, such as enrollments and account recovery.	Pass
MSTG-NETWORK-6	The app only depends on up-to-date connectivity and security libraries.	Pass
	<b>Platform Interaction</b>	
MSTG-PLATFORM-1	The app only requests the minimum set of permissions necessary.	Pass

MSTG-PLATFORM-2	All inputs from external sources and the user are validated and if necessary sanitized. This includes data received via the UI, IPC mechanisms such as intents, custom URLs, and network sources.	Pass
MSTG-PLATFORM-3	The app does not export sensitive functionality via custom URL schemes, unless these mechanisms are properly protected.	Pass
MSTG-PLATFORM-4	The app does not export sensitive functionality through IPC facilities, unless these mechanisms are properly protected.	Pass
MSTG-PLATFORM-5	JavaScript is disabled in WebViews unless explicitly required.	Pass
MSTG-PLATFORM-6	WebViews are configured to allow only the minimum set of protocol handlers required (ideally, only https is supported). Potentially dangerous handlers, such as file, tel and app-id, are disabled.	Pass
MSTG-PLATFORM-7	If native methods of the app are exposed to a WebView, verify that the WebView only renders JavaScript contained within the app package.	Pass
MSTG-PLATFORM-8	Object deserialization, if any, is implemented using safe serialization APIs.	Pass
MSTG-PLATFORM-9	The app protects itself against screen overlay attacks. (Android only)	Pass
MSTG-PLATFORM-10	A WebView's cache, storage, and loaded resources (JavaScript, etc.) should be cleared before the WebView is destroyed.	Pass
MSTG-PLATFORM-11	Verify that the app prevents usage of custom third-party keyboards whenever sensitive data is entered.	Pass
	<b>Code Quality and Build Settings</b>	
MSTG-CODE-1	The app is signed and provisioned with a valid certificate, of which the private key is properly protected.	Pass
MSTG-CODE-2	The app has been built in release mode, with settings appropriate for a release build (e.g. non-debuggable).	Pass
MSTG-CODE-3	Debugging symbols have been removed from native binaries.	Pass

MSTG-CODE-4	Debugging code and developer assistance code (e.g. test code, backdoors, hidden settings) have been removed. The app does not log verbose errors or debugging messages.	Pass
MSTG-CODE-5	All third party components used by the mobile app, such as libraries and frameworks, are identified, and checked for known vulnerabilities.	Pass
MSTG-CODE-6	The app catches and handles possible exceptions.	Pass
MSTG-CODE-7	Error handling logic in security controls denies access by default.	Pass
MSTG-CODE-8	In unmanaged code, memory is allocated, freed and used securely.	Pass
MSTG-CODE-9	Free security features offered by the toolchain, such as byte-code minification, stack protection, PIE support and automatic reference counting, are activated.	Pass



## Resiliency against Reverse Engineering

Test ID	Test name	Status
	<b>Impede Dynamic Analysis and Tampering</b>	
MSTG-RESILIENCE-1	The app detects, and responds to, the presence of a rooted or jailbroken device either by alerting the user or terminating the app.	Pass
MSTG-RESILIENCE-2	The app prevents debugging and/or detects, and responds to, a debugger being attached. All available debugging protocols must be covered.	Pass
MSTG-RESILIENCE-3	The app detects, and responds to, tampering with executable files and critical data within its own sandbox.	Pass
MSTG-RESILIENCE-4	The app detects, and responds to, the presence of widely used reverse engineering tools and frameworks on the device.	Pass
MSTG-RESILIENCE-5	The app detects, and responds to, being run in an emulator.	Pass
MSTG-RESILIENCE-6	The app detects, and responds to, tampering the code and data in its own memory space.	Pass
MSTG-RESILIENCE-7	The app implements multiple mechanisms in each defense category (8.1 to 8.6). Note that resiliency scales with the amount, diversity of the originality of the mechanisms used.	Pass
MSTG-RESILIENCE-8	The detection mechanisms trigger responses of different types, including delayed and stealthy responses.	Pass
MSTG-RESILIENCE-9	Obfuscation is applied to programmatic defenses, which in turn impede de-obfuscation via dynamic analysis.	Pass
	<b>Device Binding</b>	
MSTG-RESILIENCE-10	The app implements a 'device binding' functionality using a device fingerprint derived from multiple properties unique to the device.	Pass
	<b>Impede Comprehension</b>	

MSTG-RESILIENCE-11	All executable files and libraries belonging to the app are either encrypted on the file level and/or important code and data segments inside the executables are encrypted or packed. Trivial static analysis does not reveal important code or data.	Pass
MSTG-RESILIENCE-12	If the goal of obfuscation is to protect sensitive computations, an obfuscation scheme is used that is both appropriate for the particular task and robust against manual and automated de-obfuscation methods, considering currently published research. The effectiveness of the obfuscation scheme must be verified through manual testing. Note that hardware-based isolation features are preferred over obfuscation whenever possible.	Pass
<b>Impede Eavesdropping</b>		
MSTG-RESILIENCE-13	As a defense in depth, next to having solid hardening of the communicating parties, application level payload encryption can be applied to further impede eavesdropping.	Pass

**LEGEND**

**Pass:** Requirement is applicable to Mobile application and implemented according to best practices.

**Fail:** Requirement is applicable to Mobile application but not fulfilled.

**N/A:** Requirement is not applicable to Mobile application.