1. Vulnerability Analysis

1.1. Analysis Criteria

1.1.1. CIA

[CONFIDENTIALITY] - compromises confidentiality
[INTEGRITY] - compromises integrity
[AVAILABILITY] - compromises availability

1.1.2. Approach (what we did to find the vulnerability)

manual

| [REVIEW/CODE] - done by reviewing code itself
| [REVIEW/DESIGN] - done by reviewing documents or code
| [FUZZING] - done by running fuzzing tools
v [STATIC] - done by running static analysis tools

toolly

1.1.3. Attack vector (pathway, what attacker obtained)

high opportunity

| [SERVERINFO] - can attack only with server ip, port
| [NETWORK] - can attack over the network communicating
| [CLIENT/ACCESS] - can attack with client accessible
| [CLIENT/SOURCE] - can attack with client source exposed
| [CLIENT/PRIVILEGED] - can attack with client accessible with privileged
| [SERVER/ACCESS] - can attack with server accessible
v [SERVER/SOURCE] - can attack with server source exposed

low opportunity

1.1.4. Exploit technique

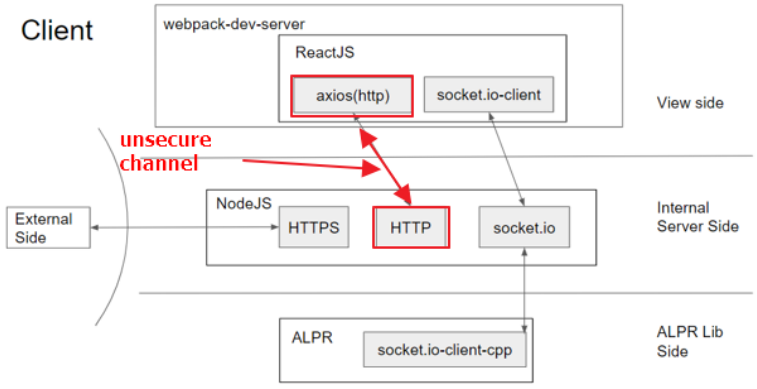
[SQLINJECTION] - for by-passing authentication
[BUFFEROVERFLOW] - reading or writing beyond legitimate area
[WRAPAROUND] - making use of unsinged type wraparound
[FORMATSTRING] - mainly used for leaking data on the stack
[REVEALEDKEY] - decrypting secure data using revealed keys
[SNIFFING] - sniffing packets over the network
[SPOOFING] - so-called, man in the middle attack
[BRUTEFORCE] - trying all possible input until success
[CRAFTPACKET] - crafting and sending a customized packet
[TAMPERING] - modifying system components for a purpose
[WEBDEBBER] - sniffing credentials using web debugger.
[NOSPECIFIED] - no special techniques specified

1.2. Vulnerability List

1.2.1. V01 – UI Webapp in client side send user credential as raw string, not encrypted

Description	[V01] UI Webapp in client side send user credential as raw string, not encrypted		
CIA	[CONFIDENTIALITY]	Attack vector	[CLIENT/ACCESS]
Approach	[REVIEW/DESIGN]	Exploit technique	[SNIFFING]
Vulnerabilities			

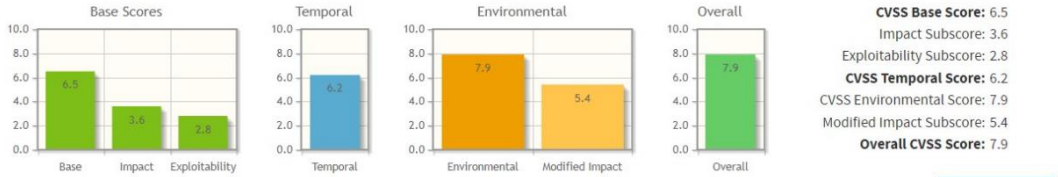
1. In client side, there are three parts; webapp for UI, local nodejs server and ALPR
2. To communicate between webapp UI and local server, HTTP which is not secured channel is used
3. It causes information disclosure, because data is not encrypted



CVSS Score	7.9	Severity	High

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS v3.1 Vector

AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/E:P/RL:X/RC:C/CR:H/IR:L/AR:L/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:H/MI:N/MA:N

Consequence / Impact analysis
The officer's credential is not only for ALPR service. It may be SSO(Single Sign On) account for police office. So, this officer's credential can be used for following attack like penetrate police office server and connect to police office portal and then steal target data and so on. In addition, store this user credential and use this for another site's dictionary attack. Because people sometimes use same ID/PW on various site. Also, attacker can sell user credential to illegal group and make money
Recommended mitigation
Use secure communication channel protocol like HTTPS(above TLS version 1.2) between module, instead of HTTP

Tools needed	Wireshark(Packet capture)
Relative component / source code	
Local Server - UI webapp	

client/web/src/images/server.js

```
13
14  // const server = https.createServer(options, app);
15  const server = http.createServer(app);
16  const cors = require('cors');
17 const io = require('socket.io')(server,{
18      cors : {
214 server.listen(4000, function () {
215     var host = server.address().address
216     var port = server.address().port
217 })
```

client/web/src/pages/login.js

```
38     onSubmit: (values) => {
39         const formData = new URLSearchParams();
40         formData.append("email", values.email);
41         formData.append("password", values.password);
42         formData.append("otp", values.otp);
43
44         axios.post('http://localhost:4000/login', null, {
45             params: formData
46         }).then((response => {
47             if(response.data.result.status !== "fail"){
48               sessionStorage.setItem("key",response.data.result.token);
49               onCancel();
50             } else {
51                alert(response.data.result.message);
52             }
53         }));
54     }
55   });
```

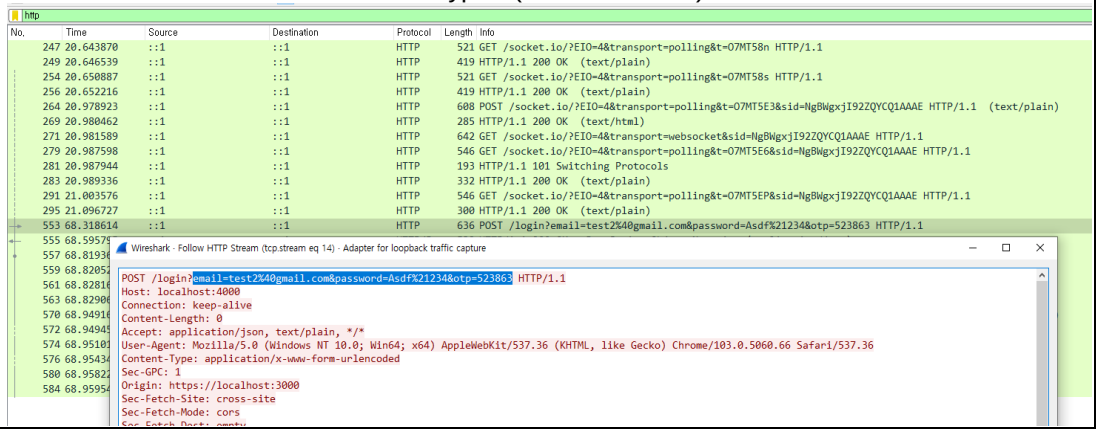
Proof of concept / How to attack

Constraints:

- Attacker can access client device and run Wireshark for packet capturing

Procedure:

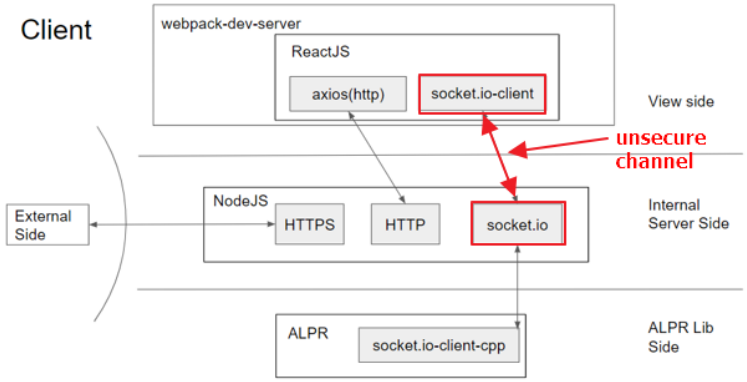
1. Run Wireshark for capturing packets and bind to localhost
2. Connect to officer UI webpage(https://localhost:3000)
3. Attempt to login with ID/PW/OTP which are provided
4. Check that user credential is not encrypted(HTTP Protocol) in Wireshark



1.2.2. V02 – Privacy is exposed in vehicle registration information

Description	[V02] Privacy is exposed in vehicle registration information		
CIA	[CONFIDENTIALITY]	Attack vector	[CLIENT/ACCESS]
Approach	[REVIEW/DESIGN]	Exploit technique	[SNIFFING]
Vulnerabilities			

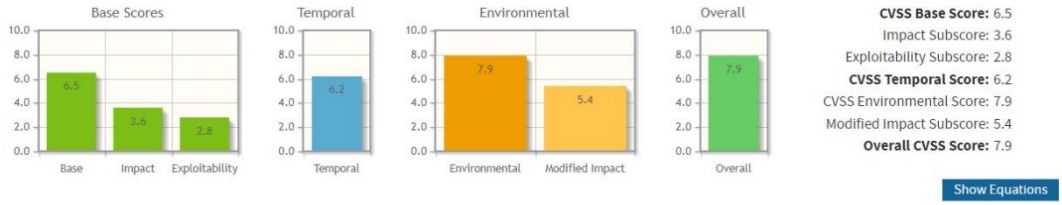
1. In client side, there are three parts; webapp for UI, local nodejs server and ALPR
2. To communicate between webapp UI and local server, websocket which is not secured channel is used
3. It causes information disclosure, because data is not encrypted



CVSS Score	7.9	Severity	High

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS v3.1 Vector

AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/E:P/RL:X/RC:C/CR:H/IR:L/AR:L/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:H/MI:N/MA:N

Consequence / Impact analysis
From vehicle registration, privacy is exposed and stolen by attacker. Attack can know vehicle user's home address and abuse for criminal. Also, attacker can sell privacy to illegal group and make money
Recommended mitigation
Use WSS(WebSockets over SSL/TLS) for secure data communication channel instead of websocket

Tools needed	Wireshark(Packet capture)

Relative component / source code
Local Server / UI webapp

client/web/src/images/server.js

```
115                 if(plateArr.length === 0 || !plateArr.includes(_data[1])){
116                     plateArr.push(_data[1]);
117                     const request = https.request(host+"/plate/get?plate_number="+_data[1], options, (response) => {
118                         let data = '';
119                         response.on('data', (chunk) => {
120                             data = data + chunk.toString();
121 
122                         });
123 
124                         response.on('end', () => {
125                             try{
126                                 const body = JSON.parse(data);
127                                 io.to(socketId).emit('queryTimeout', '');
128                                 io.to(socketId).emit('plateInfo', body);
129                             } catch(e){
130                                 console.log(e);
131                             }
132                         });
133                     });
```

client/web/src/pages/index.js

```
72    socket.on('plateInfo', (message) => {
73      let result = message;
74      let data = result.result;
75 
76      if(data.status !== "fail"){
77        setPlateInfo(oldArr => [data, ...oldArr]);
78      } else {
79        if(data.message === "Invalid token"){
80          alert(data.message + ". Please signin again.");
81          sessionStorage.clear();
82          setJwt("");
83          window.location.reload();
84        }
85      }
86    });
```

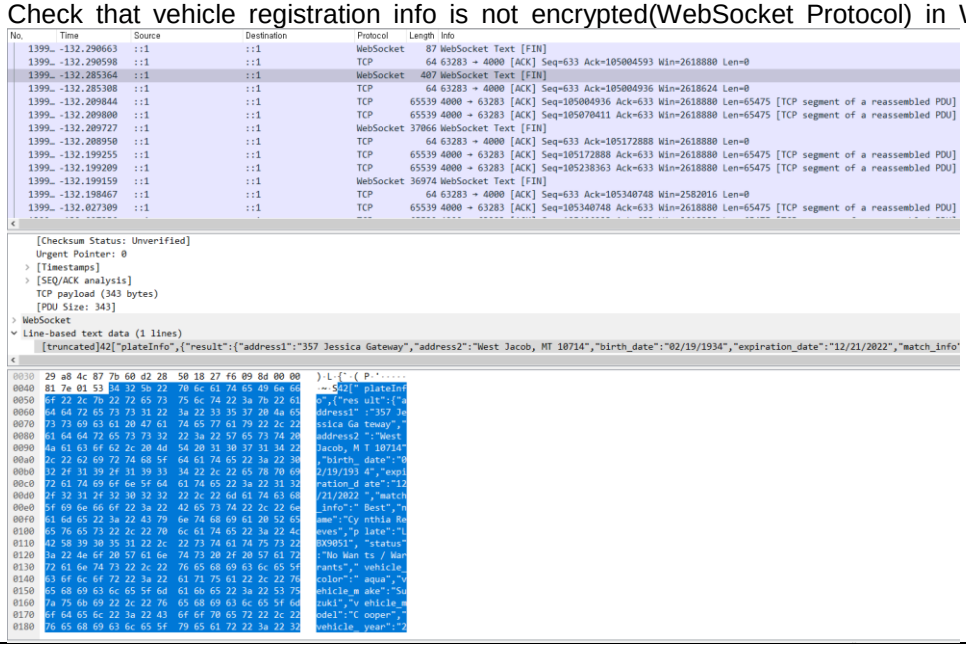
Proof of concept / How to attack

Constraints:

- Attacker can access client device and run Wireshark for packet capturing

Procedure:

1. Run Wireshark for capturing packets and bind to localhost
2. Connect to officer UI webpage(https://localhost:3000)
3. Attempt to login with ID/PW/OTP which are provided
4. Start ALPR service by opening video file "beaver1.avi" and clicking "START ALPR"
5. Check that vehicle registration info is not encrypted(WebSocket Protocol) in Wireshark



1.2.3. V03 - User credentials are exposed to URL params due to using the 'GET' method.

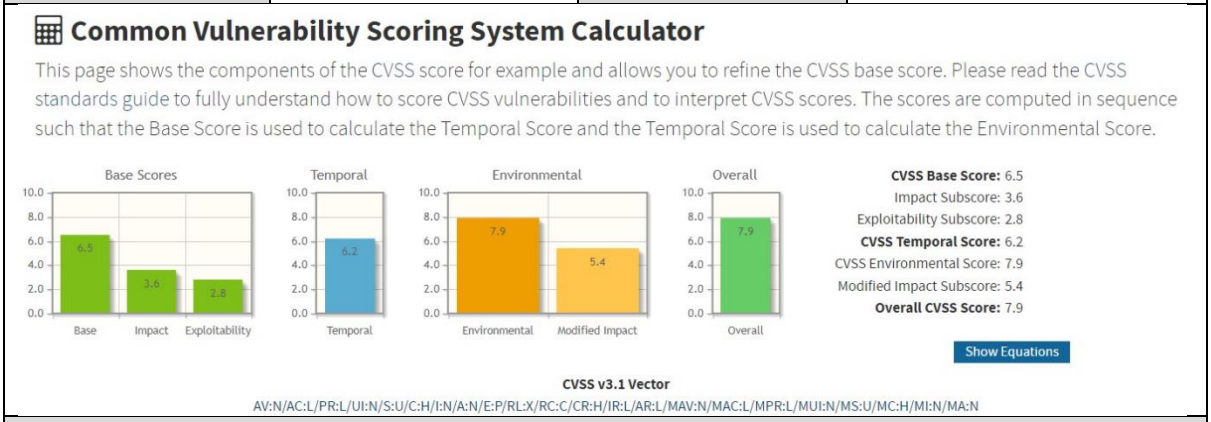
Description	[V03] User credentials are exposed to URL params due to using the 'GET' method.		
CIA	[CONFIDENTIALITY]	Attack vector	[NETWORK]
Approach	[REVIEW/CODE]	Exploit technique	[SNIFFING]

Vulnerabilities

Using 'Get' method for login API.
"https://team-server-dhzve.run.goorm.io/signin?username=test2@gmail.com&password=Asdf!234&otp=932468"

Relative component / source code

CVSS Score	7.9	Severity	High



Consequence / Impact analysis

an attacker can steal user credentials such as user name, password, otp seed.
using this information, an attacker can acquire the plate information.

Recommended mitigation

Use "POST" method instead of "GET" method in login process.

Tools needed	Postman

Proof of concept / How to attack

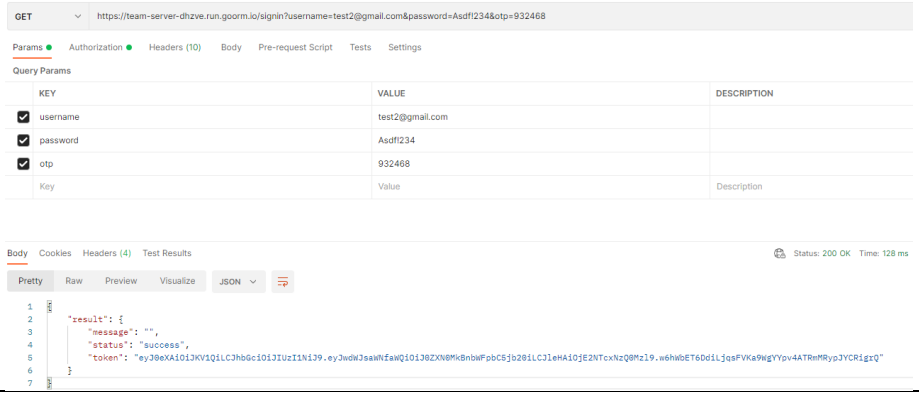
1. server.js

```
app.post('/login', function(req, res){
    let email = req.query.email;
    let password = req.query.password;
    let otp = req.query.otp;

    const url = host +"/signin?username="+email+"&password="+password+"&otp="+otp;
    const request = https.request(url, (response) => {
        let data = '';
        response.on('data', (chunk) => {
            data = data + chunk.toString();
        });
```

2. server.py

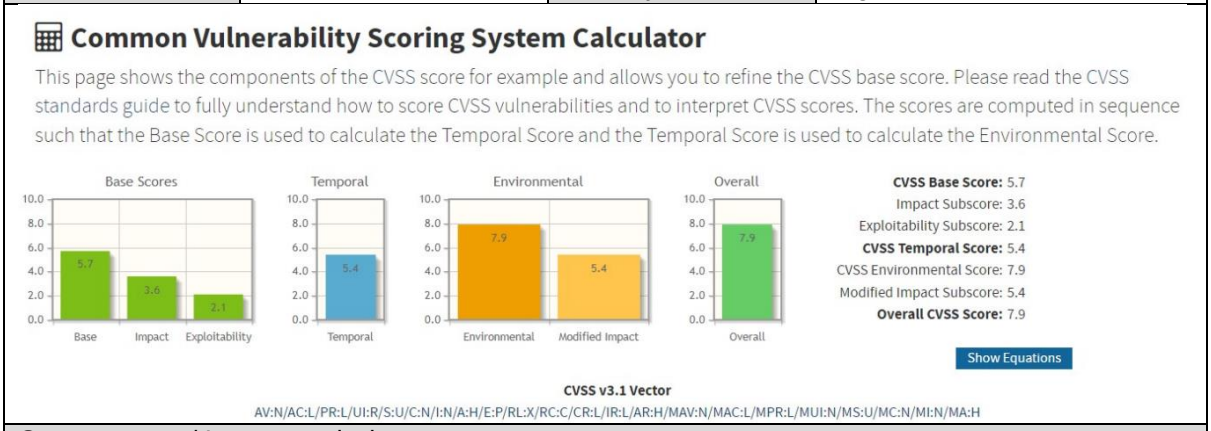
```python
user_management = Blueprint('usermanagement', __name__, url_prefix='/')

@user_management.route("/signin", methods=['GET','POST'])
def signin():

  if request.method == 'GET':
    form_user = request.args.get("username")
    form_pass = request.args.get("password")
    form_otp = request.args.get("otp")
  else:
    form_user = request.form.get("username")
    form_pass = request.form.get("password")
    form_otp = request.form.get("otp")
```

3. Run Postman
4. make Headers & Params
5. send packet to server
https://team-server-dhzve.run.goorm.io/signin?username=test2@gmail.com&password=Asdf!234&otp=932468
6. result (both http and https protocol possible)



1.2.4. V04 - Attacker can disable playback input operation of client's Server.js

Description	[V04] Attacker can disable playback input operation of client's Server.js		
CIA	[AVAILABILITY]	Attack vector	[CLIENT/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[NOSPECIFIED]
Vulnerabilities			
Abnormal selection of input playback file can compromise the Server.js as client broker. Once Server.js doesn't work, it will be never recovered until restarting client system.			
Relative component / source code			
Local Server			
CVSS Score	7.9	Severity	High



Consequence / Impact analysis
Attacker can disable ALPR system
Recommended mitigation
Input validation can be a choice to prevent selecting abnormal playback file.
Tools needed: NA
Proof of concept / How to attack
1. Click "파일 선택" and select the normal playback file 2. "START ALPR" click 3. Check displaying playback : normal operation 4. Click "파일 선택" without "STOP ALPR" 5. Select "liblept-DLL.dll" 6. Click "START ALPR" after click "STOP ALPR" 7. Check displaying nothing : normal operation 8. Click "파일 선택" and select the normal playback file 9. Click "START ALPR" 10. Check display nothing : abnormal operation - From now on, Playback doesn't work even though selecting normal playback file. - Server.js seems to be malfunctioning and never recovered until killing and restarting.

1.2.5. V05 - User credentials(username, password, otp) are exposed when using the debug tool such as chrome inspector.

Description	[V05] User credentials(username, password, otp) are exposed when using the debug tool such as chrome inspector.		
CIA	[CONFIDENTIALITY]	Attack vector	[CLIENT/SOURCE]
Approach	[REVIEW/CODE]	Exploit technique	[WEBDEBBER]

Vulnerabilities

User credentials(username, password, otp) are exposed when using the debug tool such as chrome inspector.

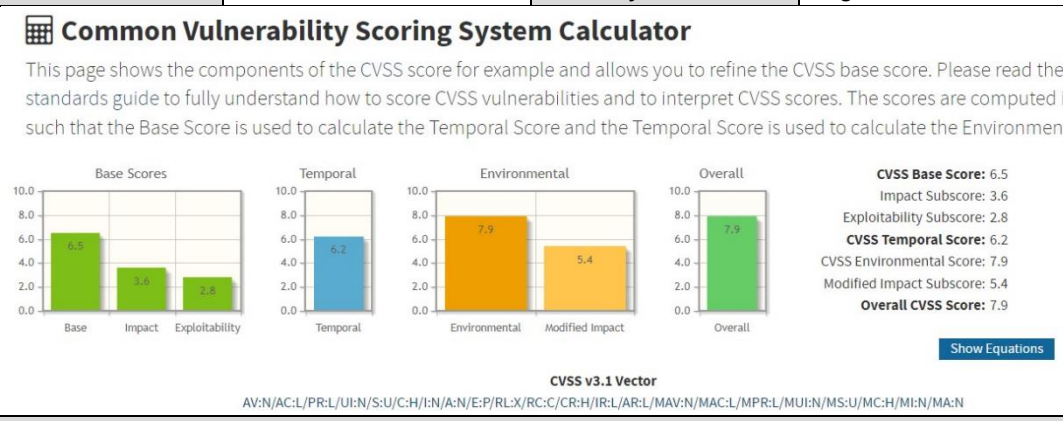
1. login.js source file

```
function Login(props) {
    const navigate = useNavigate();
    const onCancel = () =>{
        navigate(-1);
    }

  const formik = useFormik({
    initialValues: {
      email: 'test@gmail.com',
      password: 'test123',
      otp: '123456'
    },
    validationSchema: Yup.object({
    email: Yup
        .string()
        .email(
          'Must be a valid email')
        .max(80)
        .required(
          'Email is required'),
    password: Yup
        .string()
        .max(255)
        .required(
          'Password is required'),
    otp: Yup
        .number()
        .required(
                'OTP is required')
        }),
    onSubmit: (values) => {
        const formData = new URLSearchParams();
        formData.append("email", values.email);
        formData.append("password", values.password);
        formData.append("otp", values.otp);

        axios.post('http://localhost:4000/login', null, {
            params: formData
        }).then((response => {
            if(response.data.result.status !== "fail"){
              sessionStorage.setItem("key",response.data.result.token);
              onCancel();
            } else {
                alert(response.data.result.message);
            }
        }));
    }
  });
```

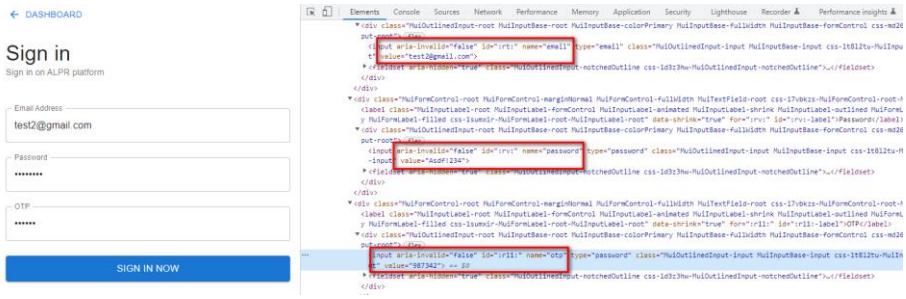
Relative component / source code

UI webapp

CVSS Score	7.9	Severity	High



Consequence / Impact analysis

an attacker can steal user credentials such as user name, password, otp seed. using this information, an attacker can acquire the plate information.
Recommended mitigation
1. Provide the client interface in the form of WebApp because it is possible to block external web debug tools at the WebApp level. 2. Using Javascript Obfuscation technology
Tools needed | Chrome Debug Mode
Proof of concept / How to attack
1. Chrome > More Tool > Developer Tools 2. Select an element in the page to inspect it 3. User credentials(username, password, otp) are exposed 

1.2.6. V06 - Token is exposed when using the debug tool such as chrome inspector or client console log

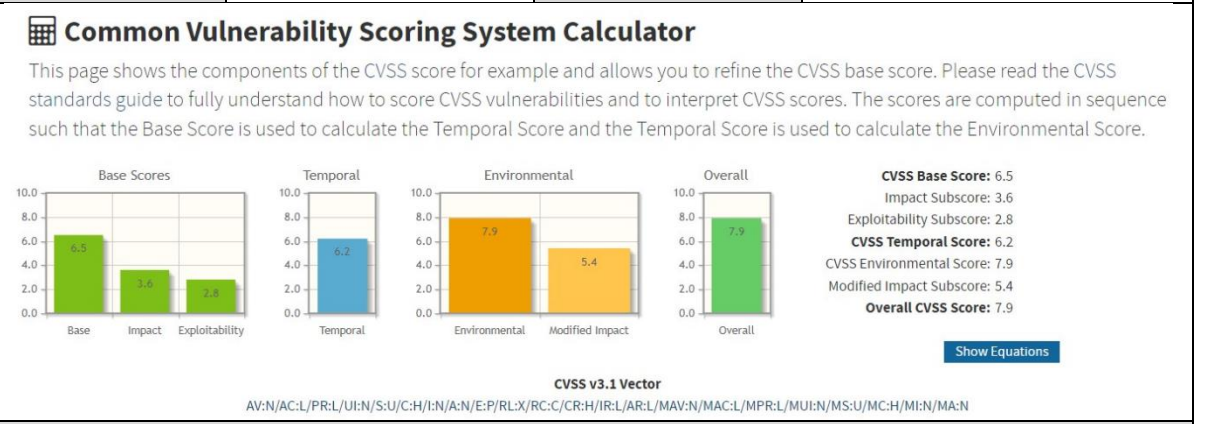
Description	[V06] Token is exposed when using the debug tool such as chrome inspector or client console log		
CIA	[CONFIDENTIALITY]	Attack vector	[CLIENT/SOURCE] [CLIENT/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[WEBDEBBER]
Vulnerabilities			

Token is exposed when using the debug tool such as chrome inspector or client console log

see Usermanagement.py

```
@user_management.route("/signin", methods=['GET','POST'])
def signin():

  if request.method == 'GET':
    form_user = request.args.get("username")
    form_pass = request.args.get("password")
    form_otp = request.args.get("otp")
  else:
    form_user = request.form.get("username")
    form_pass = request.form.get("password")
    form_otp = request.form.get("otp")

  if form_user is None or form_pass is None or form_otp is None:
    return {"result":{
              "status" : "fail",
              "message": "Invalid signin info."
            }}

  result, value = valid_username(form_user)
  if not result:
    return jsonify(value)

  data = User.query.filter_by(username=form_user).first()   # ID 조회Query 실행

  if data is not None:
    if data.password == form_pass:
      totp_instance = pyotp.TOTP(data.otp)
      valid = totp_instance.verify(form_otp)
      if valid:
        data.otpfcnt = 0
        data.passwordfcnt = 0
        db.session.commit()
        token = jwt.encode({'public_id': data.username, 'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=45)}, config._JWT_SECRET_KEY_, "HS256")
        value = {"result":{
          "status" : "success",
          "token" : token,
          "message": ""
        }}
```

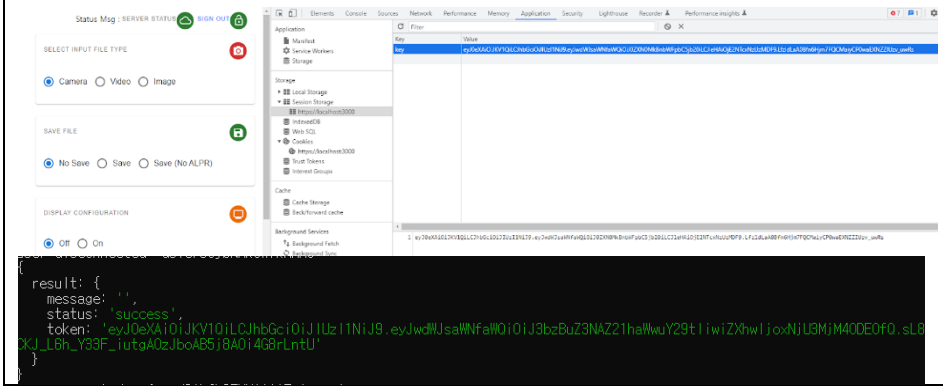
Relative component / source code			
UI webapp			
CVSS Score	7.9	Severity	High



Consequence / Impact analysis
an attacker can steal user credentials such as jwt token value. using this information, an attacker can acquire the plate information.
Recommended mitigation
1. Provide the client interface in the form of WebApp because it is possible to block external web debug tools at the WebApp level. 2. Using Javascript Obfuscation technology

Tools needed	Chrome Debug Mode
Proof of concept / How to attack	

1. Chrome > More Tool > Developer Tools
2. Select an Application Tab
3. Storage > Session Storage
4. find key
5. or check to client console
6. result



1.2.7. V07 - Username is exposed in the decoded token information.

Description	[V07] Username is exposed in the decoded token information.		
CIA	[CONFIDENTIALITY]	Attack vector	[CLIENT/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[SNIFFING]

Vulnerabilities

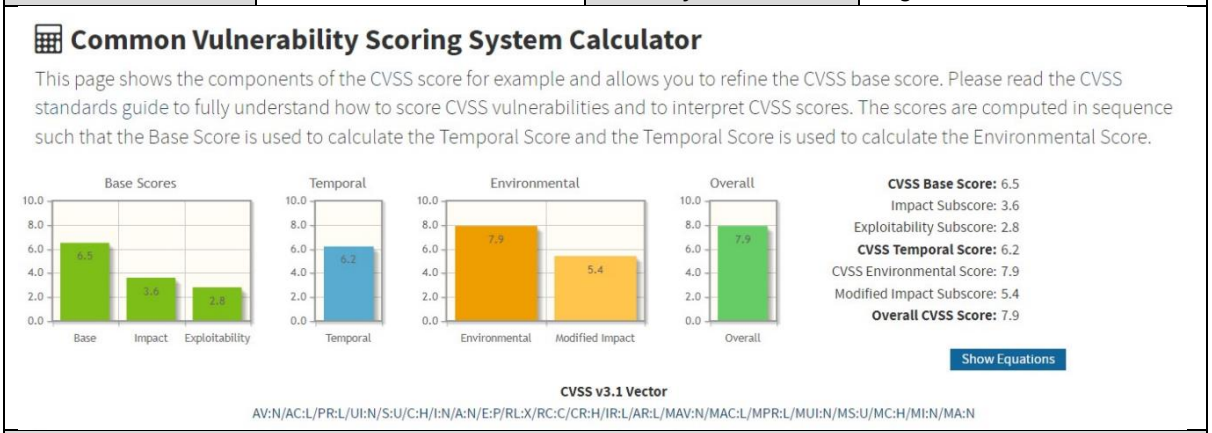
Username is exposed in the decoded token information.
See Usermanagement.py

```
if data is not None:
  if data.password == form_pass:
    totp_instance = pyotp.TOTP(data.otp)
    valid = totp_instance.verify(form_otp)
    if valid:
      data.otpfcnt = 0
      data.passwordfcnt = 0
      db.session.commit()
      token = jwt.encode({'public_id': data.username, 'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=45)}, config._JWT_SECRET_KEY_, "HS256")
      value = {"result":{
        "status" : "success",
        "token" : token,
        "message": ""
      }}
```

Relative component / source code

UI webapp

CVSS Score	7.9	Severity	High



Consequence / Impact analysis

an attacker can steal user credentials such as user name.
using this information, an attacker can try to access to login server.

Recommended mitigation

Avoid exposing user accounts to jwt token payload.
Instead, use a user id number that can replace public_id.

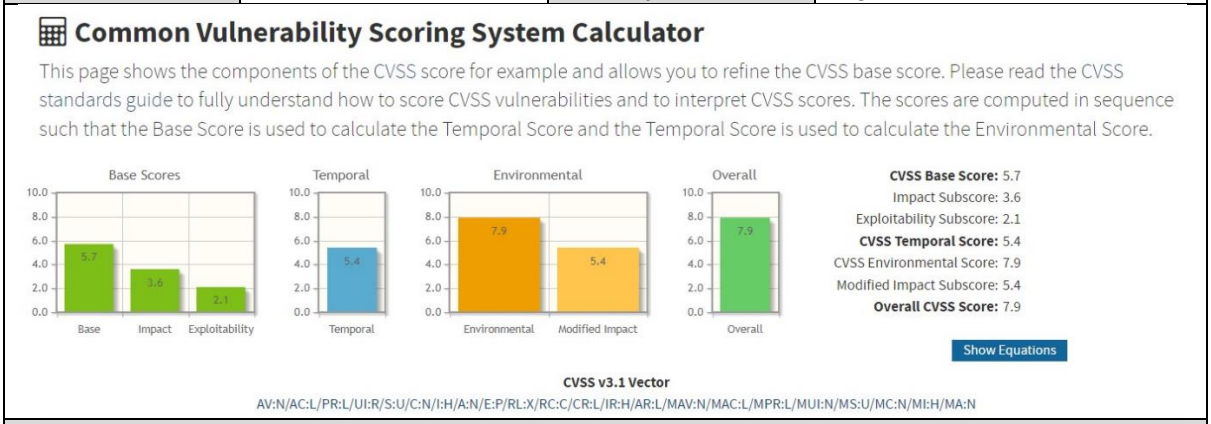
Tools needed	JWT Decoder online

Proof of concept / How to attack

1. get token using V06 Vulnerabilities
2. decode token using Base64

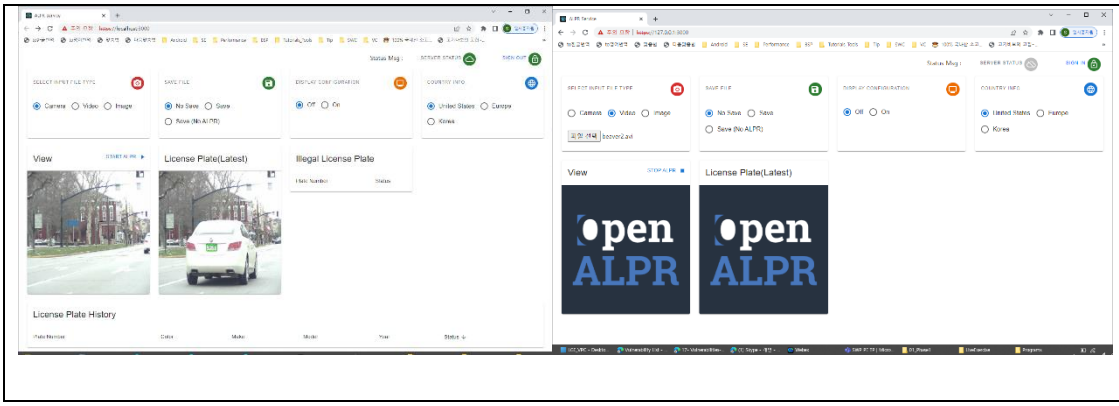
1.2.8. V08 - Attacker can inject a fake play-back to officer's browser

Description	[V08] Attacker can inject a fake play-back to officer's browser		
CIA	[INTEGRITY]	Attack vector	[CLIENT/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[SPOOFING]
Vulnerabilities			
Attacker can inject malicious playback file instead of normal playback file.			
Relative component / source code			
Local Server, ALPR.exe, UI webapp			
CVSS Score	7.9	Severity	High



Consequence / Impact analysis
An attacker can use the corrupted file to deceive the officer.
Recommended mitigation
Change design of the interface between local server and alpr.exe (ALPR.exe instance should be corresponded to UI webapp with 1:1 manner.)
Tools needed: NA
Proof of concept / How to attack

Pre-Condition
1. Officer did log-in normally.
Procedure
1. Attacker executes another browser.
2. Attacker accesses local client's server : "https://127.0.0.1:3000"
3. Attacker selects playback file with skipping log-in.
4. Attacker clicks "START ALPR."
5. Playback file selected by attacker is played on officer's browser and corresponding vehicle information are displayed also.
- Officer's browser - Attacker's browser


View
License Plate(Latest)
Illegal License Plate
License Plate History
View
License Plate(Latest)
open
ALPR
open
ALPR

1.2.9. V09 - All Username were exposed when entering a specific page

Description	[V09] All Username were exposed when entering a specific page		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN] [REVIEW/CODE]	Exploit technique	[NOSPECIFIED]

Vulnerabilities

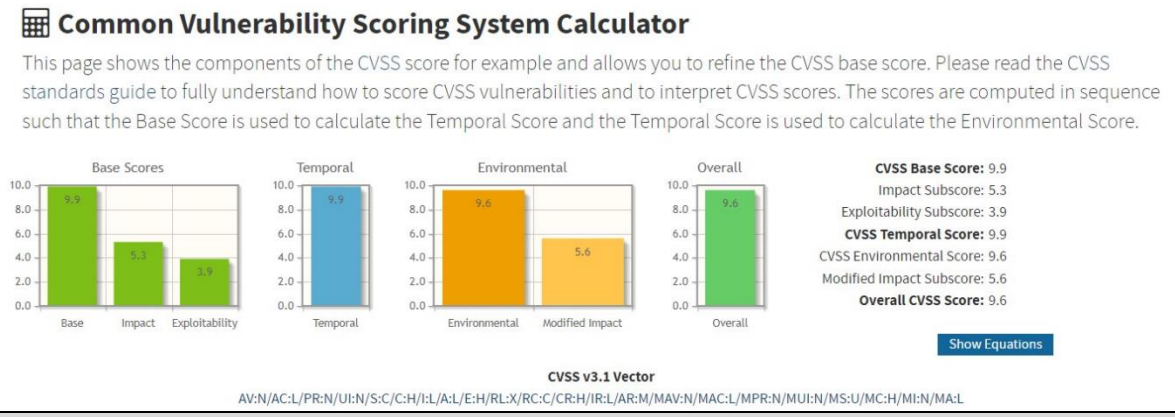
All Username were exposed when entering a specific page.
- no restrictions on API access.
- GET team-server-dhzve.run.goorm.io/admin/log

see usermanagement.py : no api access restrictions.

```
@user_management.route('/admin/log')
def adminlog():
  logs = Log.query.all()

  #sum_best = db.session.query(db.func.sum(Log.bestcnt)).first()[0]
  #sum_partial = db.session.query(db.func.sum(Log.partialcnt)).first()[0]
  #print(sum_best)
  #print(sum_partial)

  with  open("log.txt", "w+") as file:
    for log in logs:
      line = f"{log.username} | {log.qps} | {log.bestcnt} | {log.partialcnt} \n"
      file.write(line)
    file.close()

  return render_template('log.html', title='User Log', logs=logs)
```

Relative component / source code

CVSS Score	9.6	Severity	Critical



Consequence / Impact analysis

An attacker can steal user credentials such as user name in e-mail format.
using this information, an attacker can try to access to login server.

Recommended mitigation

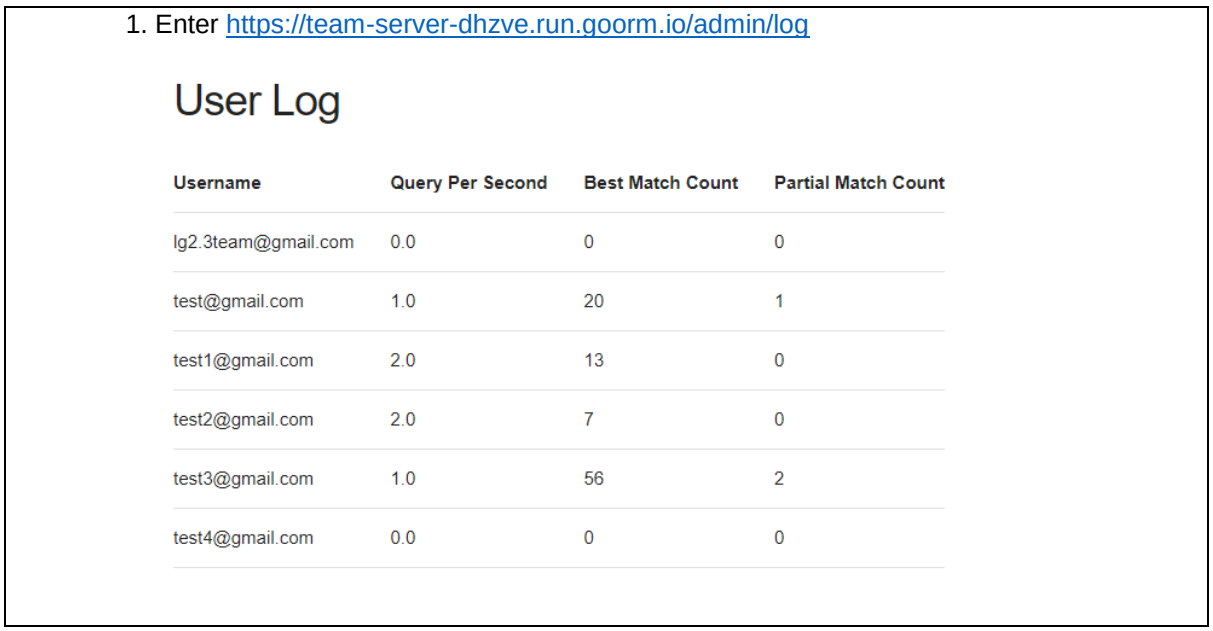
1. Add access permissions to the API provided to the administrator.

2. Minimize attackable URLs such as "team-server-dhzve.run.goorm.io/admin/log”

Tools needed	NA

Proof of concept / How to attack

1. Enter https://team-server-dhzve.run.goorm.io/admin/log



User Log

Username	Query Per Second	Best Match Count	Partial Match Count
lg2.3team@gmail.com	0.0	0	0
test@gmail.com	1.0	20	1
test1@gmail.com	2.0	13	0
test2@gmail.com	2.0	7	0
test3@gmail.com	1.0	56	2
test4@gmail.com	0.0	0	0

1.2.10. V10 - All OTP seeds were exposed when entering a specific page.

Description	[V10] All OTP seeds were exposed when entering a specific page. As a result, 2 factor authentication becomes useless.		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN] [REVIEW/CODE]	Exploit technique	[NOSPECIFIED]

Vulnerabilities

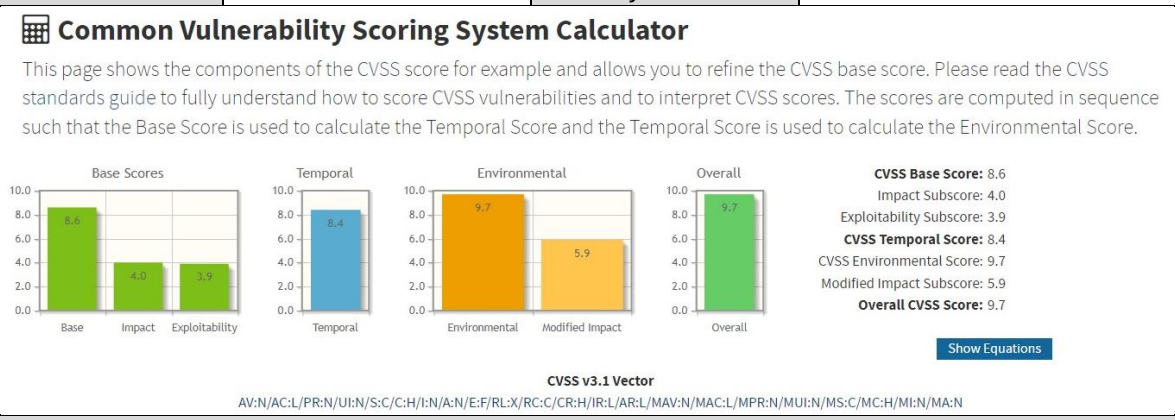
All OTP seeds were exposed when entering a specific page.
As a result, 2 factor authentication becomes useless.
- no restrictions on API access.
- GET team-server-dhzve.run.goorm.io/auth/<user>

see Usermanagement.py : no api access restrictions.

```python
@user_management.route("/auth/<user>", methods=['GET'])
def OTP_auth(user):

  data = User.query.filter_by(username=user).first()

  if data is not None:
    return render_template('auth.html', title='User OTP Info.', secret_key=data.otp,
                           prov_uri=pyotp.TOTP(data.otp).provisioning_uri(name=user, issuer_name='3team Studio Project'))
  else:
    flash("Invalid user. Please try again.")
    return redirect(url_for("usermanagement.index"))
```

Relative component / source code

CVSS Score	9.7	Severity	Critical



Consequence / Impact analysis

An attacker can steal user credentials such as user otp seed.
using this information, an attacker can try to access to login server.

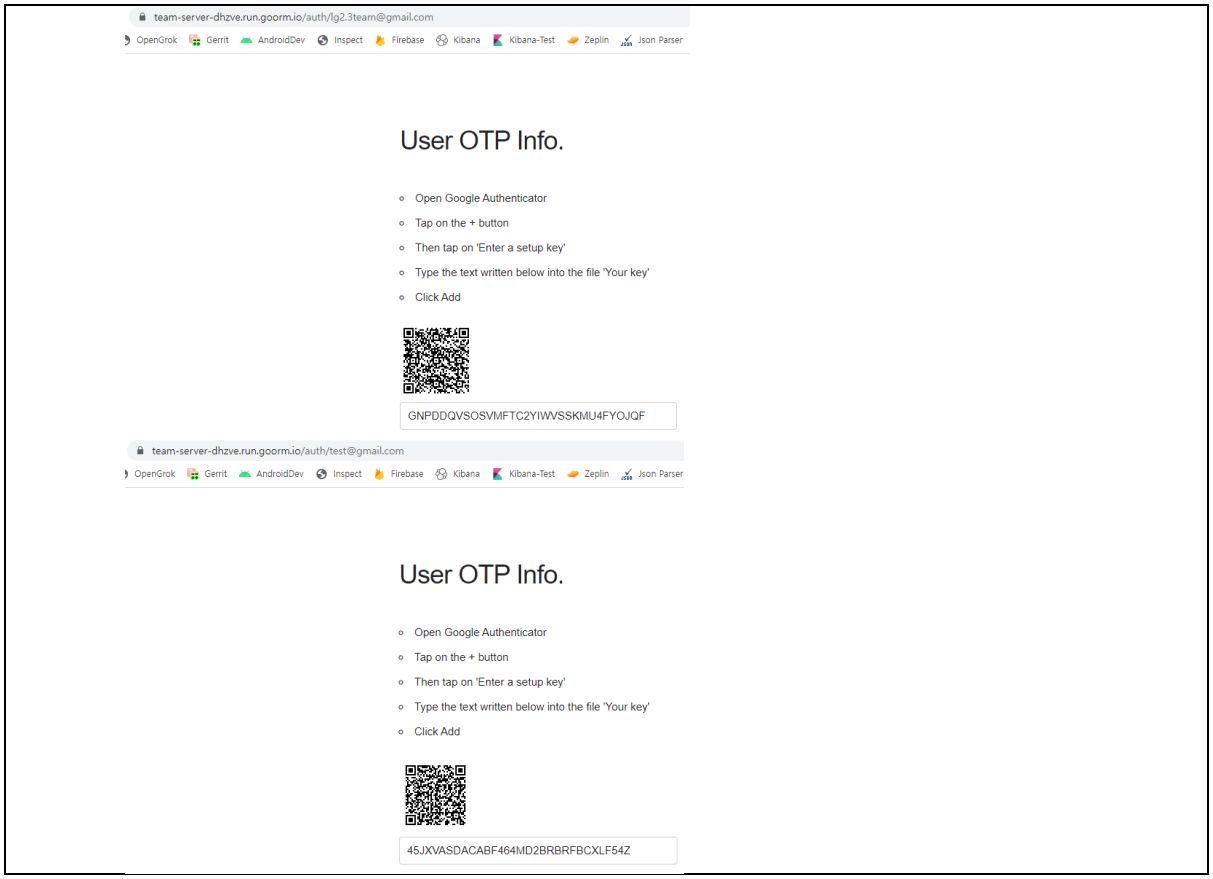
Recommended mitigation

1. Add access permissions to the API provided to the administrator.
2. Minimize attackable URLs such as "team-server-dhzve.run.goorm.io/auth/<user>"

Tools needed	NA

Proof of concept / How to attack

1. Get username from “https://team-server-dhzve.run.goorm.io/admin/log” page without authentication.
2. Enter
“https://team-server-dhzve.run.goorm.io/auth/lg2.3team@gmail.com”
“https://team-server-dhzve.run.goorm.io/auth/test@gmail.com”
“https://team-server-dhzve.run.goorm.io/auth/test1@gmail.com”
“https://team-server-dhzve.run.goorm.io/auth/test2@gmail.com”
“https://team-server-dhzve.run.goorm.io/auth/test3@gmail.com”
“https://team-server-dhzve.run.goorm.io/auth/test4@gmail.com”


team-server-dhzve.run.goorm.io/auth/lg2.3team@gmail.com
OpenGrok
Gerrit
AndroidDev
Inspect
Firebase
Kibana
Kibana-Test
Zeplin
Json Parser
User OTP Info.
Open Google Authenticator
Tap on the + button
Then tap on 'Enter a setup key'
Type the text written below into the file 'Your key'
Click Add
GNPDDQVSOSVMFTC2YIWVSSKMU4FYOJQF
team-server-dhzve.run.goorm.io/auth/test@gmail.com
OpenGrok
Gerrit
AndroidDev
Inspect
Firebase
Kibana
Kibana-Test
Zeplin
Json Parser
User OTP Info.
Open Google Authenticator
Tap on the + button
Then tap on 'Enter a setup key'
Type the text written below into the file 'Your key'
Click Add
45JXVASDACABF464MD2BRBRFBCXLF54Z

1.2.11. V11 - The administrator account has been hijacked and can login with the admin`s credentials(username, password, otp).

Description	[V11] The administrator account has been hijacked and can login with the admin`s credentials(username, password, otp).		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN] [REVIEW/CODE]	Exploit technique	[CRAFTPACKET] [SPOOFING]

Vulnerabilities

The administrator account has been hijacked and can login with the admin`s credentials(username, password, otp).

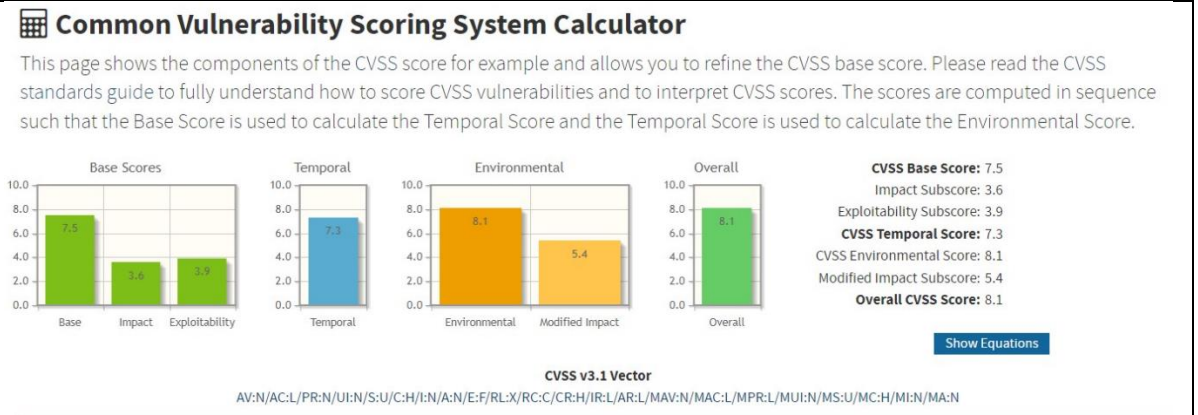
- “team-server-dhzve.run.goorm.io/admin/signin”

see Usermamagement.py : use POST method

```
@user_management.route("/admin/signin", methods=['GET', 'POST'])
def adminsignin():
  form_user = request.form.get("username")
  form_pass = request.form.get("password")
  form_otp = request.form.get("otp")

  if request.method == 'POST':
    data = User.query.filter_by(username=form_user).first()

    if data is not None:
      if data.password == form_pass :
        totp_instance = pyotp.TOTP(data.otp)
        valid = totp_instance.verify(form_otp)
        if valid:
          if data.admin :
            session['username'] = "admin"
            data.otpfcnt = 0
            data.passwordfcnt = 0
            db.session.commit()
            return redirect(url_for("usermanagement.page"))
          else :
            flash("You don't have admin authorizetion")
```

Relative component / source code

CVSS Score	8.1	Severity	High



Consequence / Impact analysis

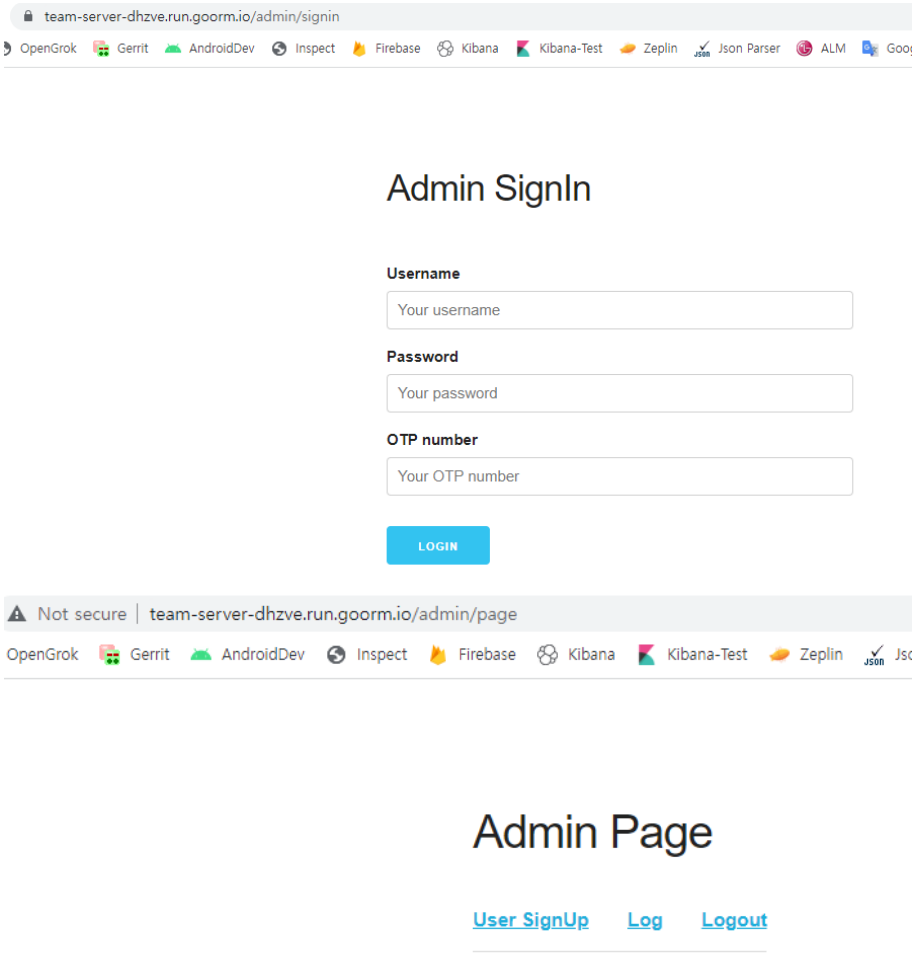
1. an attacker accesses the server with the stolen admin`s account information and gets a admin`s cookie from server.
2. Using this information, an attacker can add new illegal account.

Recommended mitigation

1. Add access permissions to the API provided to the administrator.
2. Minimize attackable URLs such as "team-server-dhzve.run.goorm.io/auth/<user>”
3. Block repeated failed login attempts for a certain period of time

4. When the number of failed logins is exceeded, the account is locked.	
Tools needed	Brutus Password Cracker
Proof of concept / How to attack	

1. to get admin`s username and password from login`s URL params or chrome inspector or sniff the client.
2.toget otp seeds form
"https://team-server-dhzve.run.goorm.io/auth/lg2.3team@gmail.com"
3. enter "https://team-server-dhzve.run.goorm.io/admin/signin"
- otp seed : "GNPDDQVSOSVMFTC2YIWVSSKMU4FYOJQF"
4. input Username : "lg2.3team@gmail.com" and Password: "qwer!234", OTP number : "xxxxxx"



1.2.12. V12 - Any user account can be added by obtaining administrator privileges

Description	[V12] Any user account can be added by obtaining administrator privileges (admin login or stolen cookie)		
CIA	[INTEGRITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN] [REVIEW/CODE]	Exploit technique	[CRAFTPACKET] [SPOOFING]

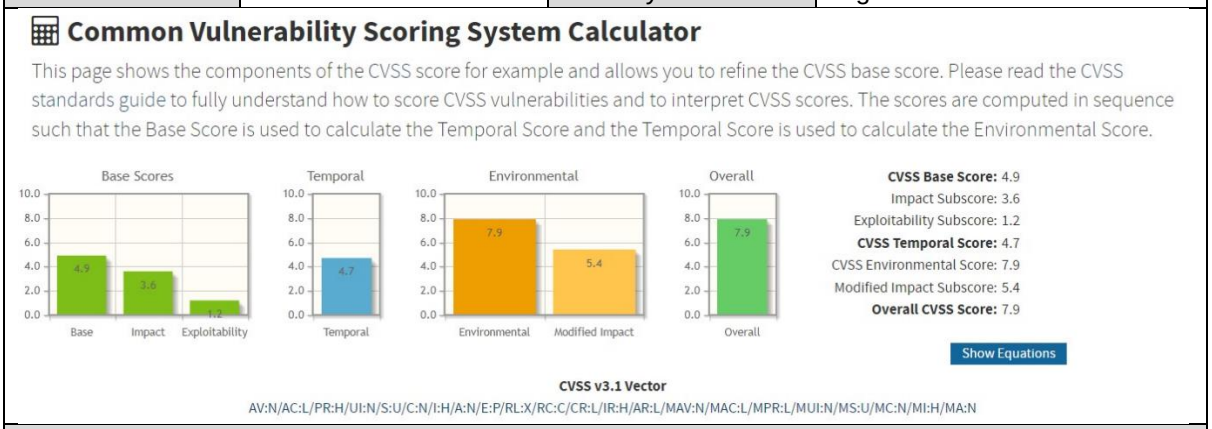
Vulnerabilities

Any user account can be added by obtaining administrator privileges.
(admin login or stolen cookie)
Usermanagement.py

```
@user_management.route("/admin/signup", methods=['GET', 'POST'])
def siginup():
  if not "username" in session:
    flash("You are not authorized to signup, Please sign in admin account")
    return render_template('index.html', title='Admin SignIn')

  if session['username'] != "admin":
    flash("You are not authorized to signup, Please sign in admin account")
    return render_template('index.html', title='Admin SignIn')

  form_user = request.form.get("username")
  form_pass = request.form.get("password")
  form_confirm_pass = request.form.get("confirm_password")

  if request.method == 'POST':
    if form_user is None or form_pass is None or form_confirm_pass is None:
      return {"result":{
               "status" : "fail",
               "message": "Invalid signup info."
              }}

    result, value = valid_username(form_user)
```

Relative component / source code

CVSS Score	7.9	Severity	High



Consequence / Impact analysis

An attacker can pollute the user db by adding illegal users.
And An attacker can acquire the plate information using illegal users information

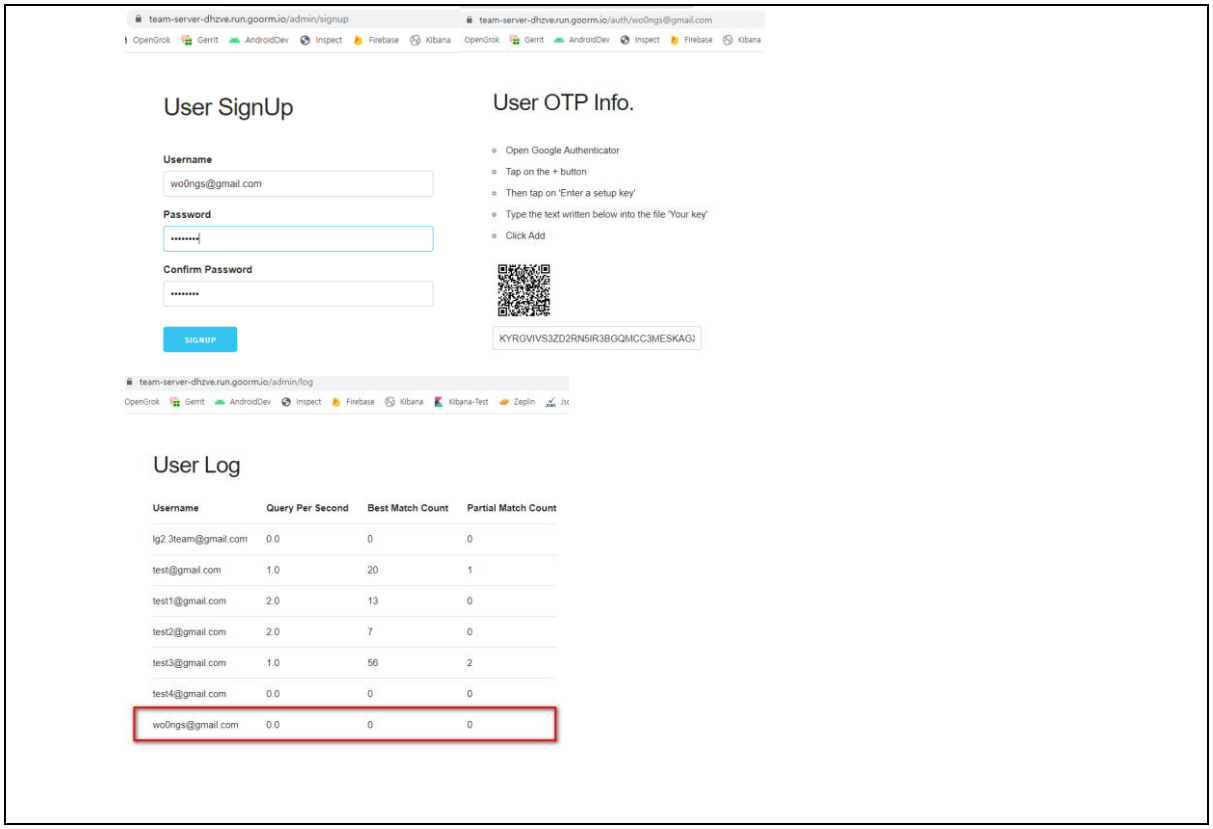
Recommended mitigation

Use trusted device and user authentication using the PKI with SSL/TLS

Tools needed	Postman

Proof of concept / How to attack

1. Log in through the page below
   "https://team-server-dhzve.run.goorm.io/admin/signin"
   - Administrator credentials can be acquired through ID12.
2. enter "https://team-server-dhzve.run.goorm.io/admin/signup"
3. add user


team-server-dhzve.run.goorm.io/admin/signup
team-server-dhzve.run.goorm.io/auth/wo0ngs@gmail.com
OpenGrok Gerrit AndroidDev Inspect Firebase Kibana
User SignUp
Username
wo0ngs@gmail.com
Password
Confirm Password
SIGNUP
User OTP Info.
Open Google Authenticator
Tap on the + button
Then tap on 'Enter a setup key'
Type the text written below into the file 'Your key'
Click Add
KYRGVIVS3ZD2RN5IR3BGQMCC3MESKAGJ
team-server-dhzve.run.goorm.io/admin/log
OpenGrok Gerrit AndroidDev Inspect Firebase Kibana Kibana-Test Zeplin
User Log
Username
Query Per Second
Best Match Count
Partial Match Count
lg2.3team@gmail.com 0.0 0 0
test@gmail.com 1.0 20 1
test1@gmail.com 2.0 13 0
test2@gmail.com 2.0 7 0
test3@gmail.com 1.0 56 2
test4@gmail.com 0.0 0 0
wo0ngs@gmail.com 0.0 0 0

1.2.13. V13 - Direct access to "https://team-server-dhzve.run.goorm.io/admin/signup" page is possible through the exposed admin cookie value.

Description	[V13] Direct access to "https://team-server-dhzve.run.goorm.io/admin/signup" page is possible through the exposed admin cookie value. and any account can be added possibly		
CIA	[CONFIDENTIALITY] [INTEGRITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/DESIGN] [REVIEW/CODE]	Exploit technique	[CRAFTPACKET] [WEBDEBBER]

Vulnerabilities

usermanagement.py

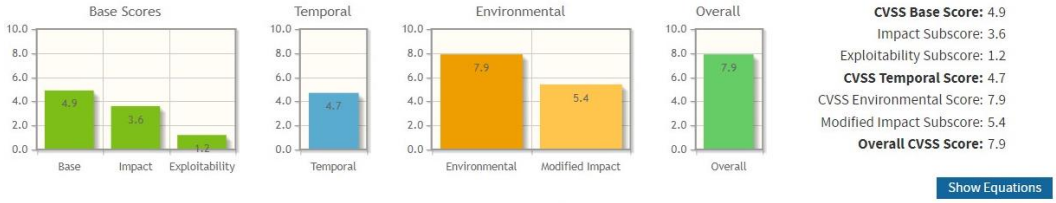
```
@user_management.route("/admin/signup", methods=['GET', 'POST'])
def siginup():
  if not "username" in session:
    flash("You are not authorized to signup, Please sign in admin account")
    return render_template('index.html', title='Admin SignIn')

  if session['username'] != "admin":
    flash("You are not authorized to signup, Please sign in admin account")
    return render_template('index.html', title='Admin SignIn')

  form_user = request.form.get("username")
  form_pass = request.form.get("password")
  form_confirm_pass = request.form.get("confirm_password")

  if request.method == 'POST':
    if form_user is None or form_pass is None or form_confirm_pass is None:
      return {"result":{
                "status" : "fail",
                "message": "Invalid signup info."
              }}

    result, value = valid_username(form_user)
    if not result:
      flash(value)
      return render_template('signup.html', title='User Signup')

    result, value = valid_password(form_pass)
    if not result:
      flash(value)
      return render_template('signup.html', title='User Signup')

    data = User.query.filter_by(username=form_user).first()

    if data is not None:
      flash("User Name already exists. Please try with another one")
      return render_template('signup.html', title='User Signup')

    else:
      if form_pass == form_confirm_pass:
        otptoken=pyotp.random_base32()
        user = User(username=form_user, password=form_pass, otp=otptoken, admin=False)
        log = Log(username=form_user)
        db.session.add(user)
        db.session.add(log)
        db.session.commit()

        return render_template('auth.html', title="User OTP Info.", secret_key=otptoken,
                               prov_uri=pyotp.TOTP(otptoken).provisioning_uri(name=form_user, issuer_name='3team Studio Project'))
```

Relative component / source code

CVSS Score	7.9	Severity	High

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS Base Score: 4.9
Impact Subscore: 3.6
Exploitability Subscore: 1.2
CVSS Temporal Score: 4.7
CVSS Environmental Score: 7.9
Modified Impact Subscore: 5.4
Overall CVSS Score: 7.9

Show Equations

CVSS v3.1 Vector

AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:H/A:N/E:P/RL:X/RC:C/CR:L/IR:H/AR:L/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:N/MI:H/MA:N

Consequence / Impact analysis

An attacker can steal Administrator`s credentials such as cookie and use this admin credentials to add new illegal account.

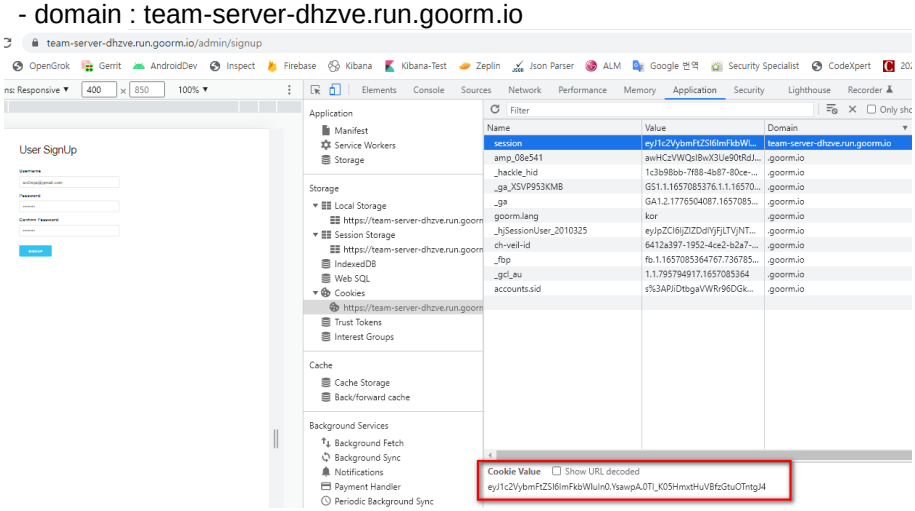
Recommended mitigation

1. Use trusted device and user authentication using the PKI with SSL/TLS
2. Using Javascript Obfuscation technology

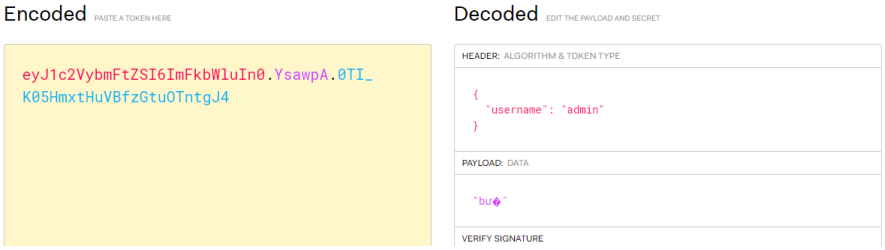
Tools needed	Postman or Chrome inspector,

Proof of concept / How to attack

1. exposed admin cookies
 - domain : team-server-dhzve.run.goorm.io



2. decoded cookies



3. insert cookie to another pc browser

1.2.14. V14 - no log records for login or logout with an administrator account and looking up log on the server side

Description	[V14] no log records for login or logout with an administrator account and looking up log on the server side		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[NOSPECIFIED]

Vulnerabilities

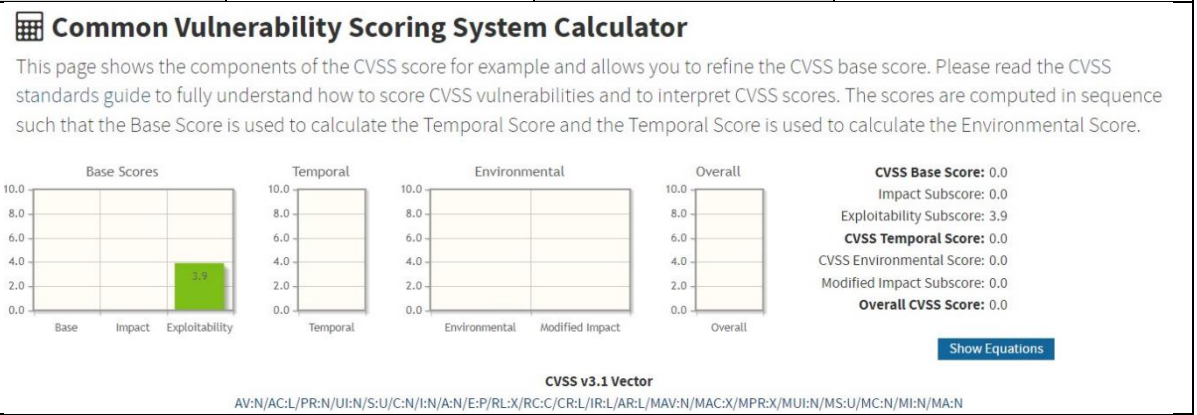
usermanagement.py - no log

```
@user_management.route("/admin/signin", methods=['GET', 'POST'])
def adminsignin():
  form_user = request.form.get("username")
  form_pass = request.form.get("password")
  form_otp = request.form.get("otp")

  if request.method == 'POST':
    data = User.query.filter_by(username=form_user).first()

    if data is not None:
      if data.password == form_pass :
        totp_instance = pyotp.TOTP(data.otp)
        valid = totp_instance.verify(form_otp)
        if valid:
          if data.admin :
            session['username'] = "admin"
            data.otpfcnt = 0
            data.passwordfcnt = 0
            db.session.commit()
            return redirect(url_for("usermanagement.page"))
          else :
            flash("You don't have admin authorizetion")
```

```
@user_management.route("/admin/signin", methods=['GET', 'POST'])
def adminsignin():
  form_user = request.form.get("username")
  form_pass = request.form.get("password")
  form_otp = request.form.get("otp")

  if request.method == 'POST':
    data = User.query.filter_by(username=form_user).first()

    if data is not None:
      if data.password == form_pass :
        totp_instance = pyotp.TOTP(data.otp)
        valid = totp_instance.verify(form_otp)
        if valid:
          if data.admin :
            session['username'] = "admin"
            data.otpfcnt = 0
            data.passwordfcnt = 0
            db.session.commit()
            return redirect(url_for("usermanagement.page"))
          else :
            flash("You don't have admin authorization")
```

Relative component / source code

CVSS Score	0.0	Severity	None



Consequence / Impact analysis

An attacker accesses the server with the stolen account, but the server does not record. So, it is impossible to track the attacker.

Recommended mitigation

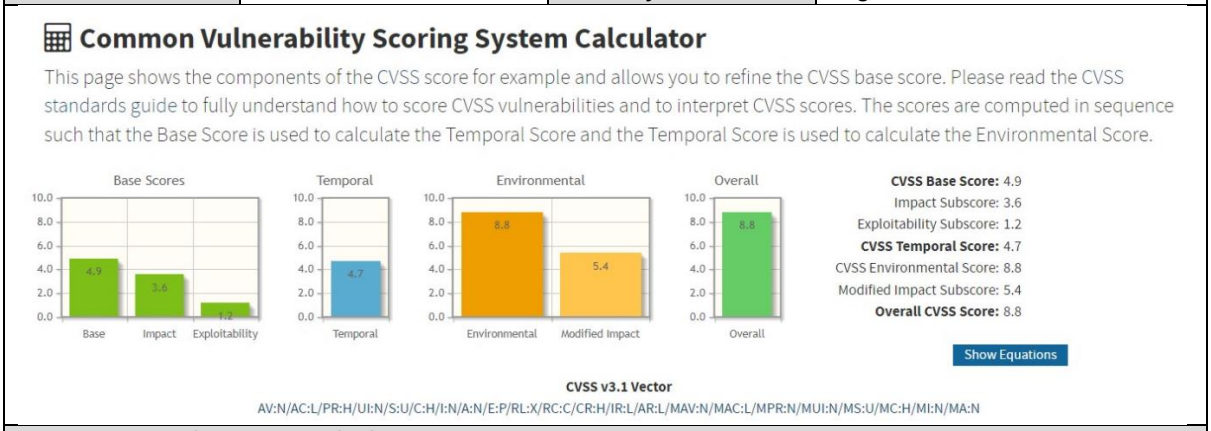
Logging

Tools needed	NA

Proof of concept / How to attack

1.2.15. V15 - An attacker can retrieve plate information with a newly added user account by the stolen administrator account.

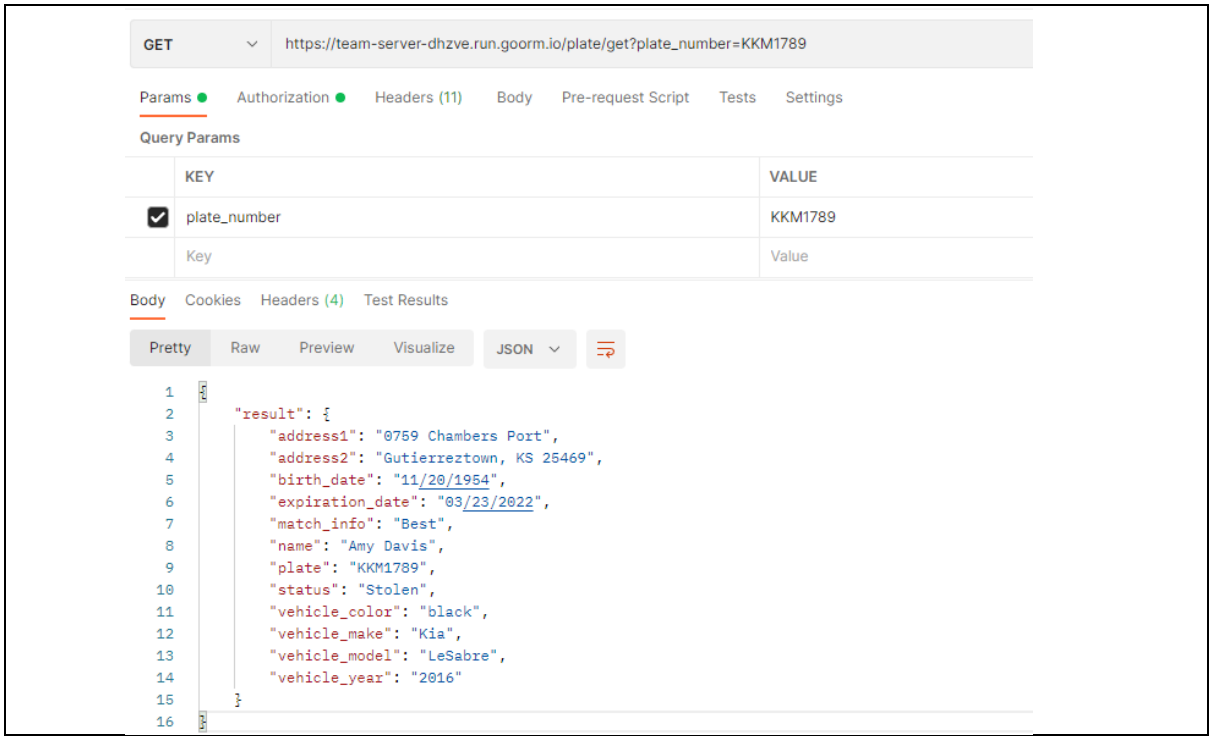
Description	[V15] An attacker can retrieve plate information with a newly added user account by the stolen administrator account.		
CIA	[CONFIDENTIALITY] [INTEGRITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[CRAFTPACKET] [SPOOFING]
Vulnerabilities			
Attacker can add new user account.			
Relative component / source code			
CVSS Score	8.8	Severity	High



Consequence / Impact analysis
An attacker can acquire the plate information An Attacker can add new account arbitrarily
Recommended mitigation
1. Use trusted device and user authentication using the PKI with SSL/TLS 2. Using Javascript Obfuscation technology 3. Strengthen the security of the OTP issuance process

Tools needed	Postman

Proof of concept / How to attack
1. The attacker adds user accounts according to V12. - wo0ngs@gmail.com is added 2. Login with account “wo0ngs@gmail.com” 3. We can obtain plate and license information by inquiring about any plate number.


GET
https://team-server-dhzve.run.goorm.io/plate/get?plate_number=KKM1789
Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
plate_number
KKM1789
Key
Value
Body
Cookies
Headers (4)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"result": {
"address1": "0759 Chambers Port",
"address2": "Gutierreztown, KS 25469",
"birth_date": "11/20/1954",
"expiration_date": "03/23/2022",
"match_info": "Best",
"name": "Amy Davis",
"plate": "KKM1789",
"status": "Stolen",
"vehicle_color": "black",
"vehicle_make": "Kia",
"vehicle_model": "LeSabre",
"vehicle_year": "2016"
}
}

1.2.16. V16 - jwt token spoofed by an attacker

Description	[V16] jwt token spoofed by an attacker		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[SPOOFING]

Vulnerabilities

For the jwt token issued by the server, it is not stored in association with user information.
Therefore, multiple logins with the same username are possible.
and there is no logic to handle expired tokens by user logout.
Also if the private key is exposed, it is impossible to validate the jwt token spoofed by an attacker on the server side because the issued jwt token format is simple.
it is defending on the config._JWT_SECRET_KEY_ only.

1. usermanagement.py

```
@user_management.route("/signin", methods=['GET','POST'])
def signin():

  if request.method == 'GET':
    form_user = request.args.get("username")
    form_pass = request.args.get("password")
    form_otp = request.args.get("otp")
  else:
    form_user = request.form.get("username")
    form_pass = request.form.get("password")
    form_otp = request.form.get("otp")

  if form_user is None or form_pass is None or form_otp is None:
    return {"result":{
            "status" : "fail",
            "message": "Invalid signin info."
          }}

  result, value = valid_username(form_user)
  if not result:
    return jsonify(value)

  data = User.query.filter_by(username=form_user).first()   # ID 조회Query 실행

  if data is not None:
    if data.password == form_pass:
      totp_instance = pyotp.TOTP(data.otp)
      valid = totp_instance.verify(form_otp)
      if valid:
        data.otpfcnt = 0
        data.passwordfcnt = 0
        token = jwt.encode({'public_id': data.username, 'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=45)}, config._JWT_SECRET_KEY_, "HS256")
        value = {"result":{
          "status" : "success",
          "token" : token,
          "message": ""
        }}
```

no stored associated with user information

2..utils.py

- the token is checked only for tampering without comparing the token stored in the server.

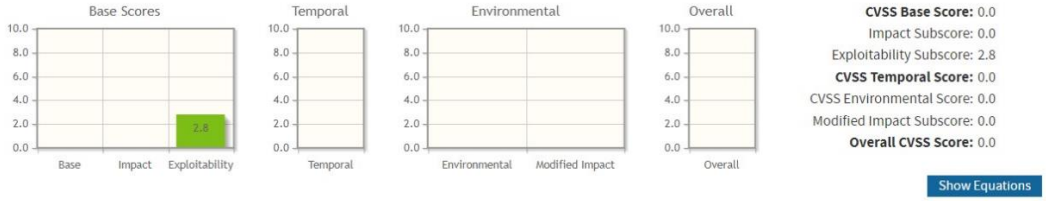
```
def token_required(f):
  @wraps(f)
  def decorator(*args, **kwargs):
    auth = request.headers.get("Authorization")
    if not auth:
      value = {"result":{
        "status" : "fail",
        "message": "Valid token is Missing"
      }}
      return jsonify(value)

    token = None
    if auth.startswith("Bearer "):
      split = auth.split("Bearer")
      token = split[1].strip()
    if not token:
      value = {"result":{
        "status" : "fail",
        "message": "Valid token is Missing"
      }}
      return jsonify(value)

    try:
      data = jwt.decode(token, config._JWT_SECRET_KEY_, algorithms=["HS256"])
      valid_user = User.query.filter_by(username=data['public_id']).first()
    except:
      value = {"result":{
        "status" : "fail",
        "message": "Invalid token"
      }}
      return jsonify(value)

    return f(valid_user, *args, **kwargs)

  return decorator
```

Relative component / source code

CVSS Score	0.0	Severity	None

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS Base Score: 0.0
Impact Subscore: 0.0
Exploitability Subscore: 2.8
CVSS Temporal Score: 0.0
CVSS Environmental Score: 0.0
Modified Impact Subscore: 0.0
Overall CVSS Score: 0.0

Show Equations

CVSS v3.1 Vector

AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:N/E:P/RL:X/RC:C/CR:L/IR:L/AR:L/MAV:N/MAC:L/MPR:N/MUI:N/MS:U/MC:N/MI:N/MA:N

Consequence / Impact analysis
An attacker can acquire the plate information

Recommended mitigation
Using JWT token revoke mechanism No multiple access with one account

Tools needed	Postman

Proof of concept / How to attack

1.2.17. V17 – Vulnerability of recovering user password

Description	[V17] Vulnerability of recovering user password		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[BRUTEFORCE]

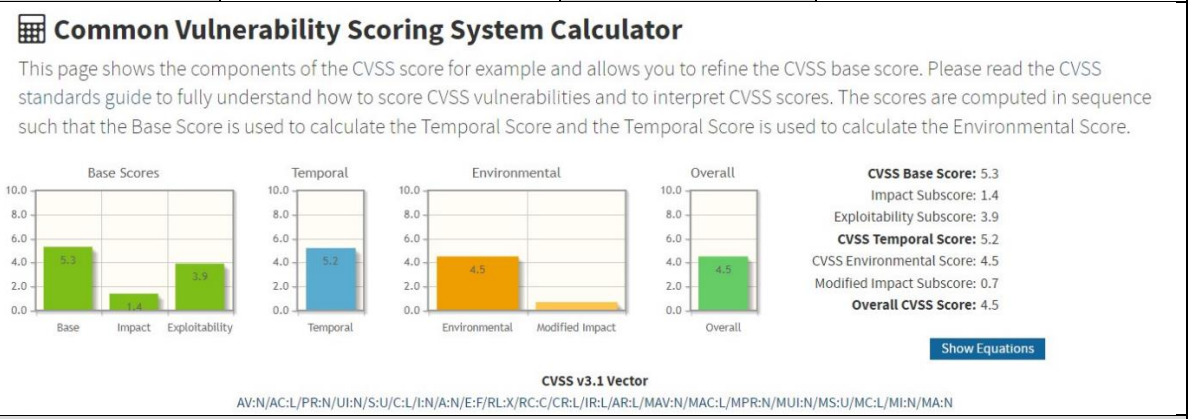
Vulnerabilities

If the password and OTP information are wrong 3 times or more, a new password is automatically sent to the account email without authentication.
In addition, the new password is the value in which the old passwords are listed in reverse order. So, the original password and the new password appear repeatedly.
Therefore, if the user`s account email is exposed, the attacker could easily obtain a password and may be vulnerable to brute force attacks.
- usermanagement.js

```
if data is not None:
  if data.password == form_pass:
    totp_instance = pyotp.TOTP(data.otp)
    valid = totp_instance.verify(form_otp)
    if valid:
      data.otpfcnt = 0
      data.passwordfcnt = 0
      db.session.commit()
      token = jwt.encode({'public_id': data.username, 'exp': datetime.datetime.utcnow() + datetime.
      value = {"result":{
        "status" : "success",
        "token" : token,
        "message": ""
      }}

    else:
      data.otpfcnt = data.otpfcnt + 1
      db.session.commit()
      if data.passwordfcnt + data.otpfcnt > 2 :
        password = data.password
        data.password = password[::-1]
        data.passwordfcnt = 0
        data.otpfcnt = 0
        db.session.commit()
        snedmail(data.username, f"new password : {data.password}")
        value = {"result":{
          "status" : "fail",
          "token" : "",
          "message": f"A new password has been sent to you by backup e-mail({data.username})"
        }}
      else:
        value = {"result":{
          "status" : "fail",
          "token" : "",
          "message": "Invalid OTP or Password Number"
        }}
```

Relative component / source code

CVSS Score	4.5	Severity	Medium



Consequence / Impact analysis

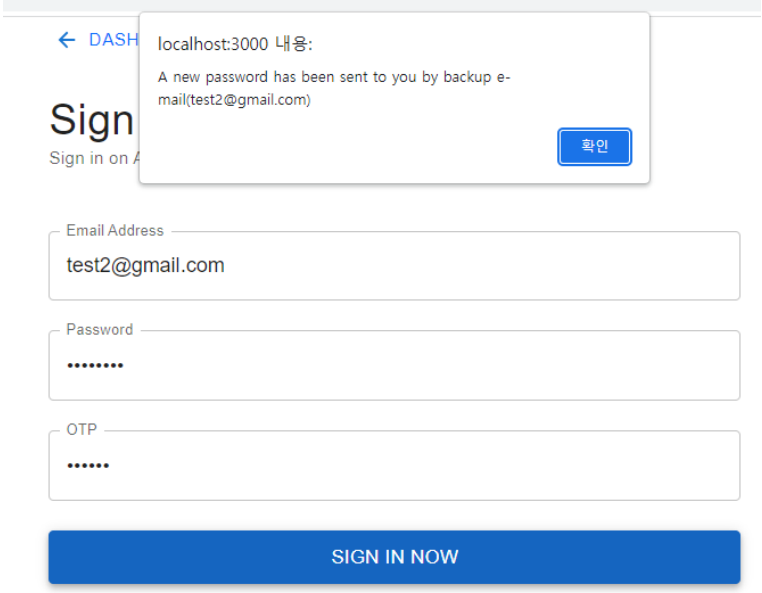
An attacker can try to steal admin or user account using brute force attacks.

Recommended mitigation

When the number of failed logins is exceeded, the account is locked.

Tools needed	brute force attacks
Proof of concept / How to attack	

1. password invalid two times.



2. repeat password invalid two times.
3. input same password and valid otp
4. login successful

1.2.18. V18 - An attacker can retrieve plate information with a stolen jwt token.

Description	[V18] An attacker can retrieve plate information with a stolen jwt token.		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[CRAFTPACKET] [SPOOFING]

Vulnerabilities

An attacker can retrieve plate information with a stolen jwt token.
If a jwt token is stolen, it cannot be dealt with until it expires.
1.server.js

```
function getFrameData(query){
    var jwt = query.jwt;
    var executablePath = path.resolve('alpr.exe');
    var parameters = ["-c", query.ct, "-s", query.save, "-f", query.res];

    if(query.type === "camera"){
        parameters.push(query.name);
    } else if(query.type === "video" || query.type === "image"){
        parameters.push(path.resolve(query.name));
    }

    if(pid === 0){
        var cmd = executablePath + " " + parameters.join(' ');
        const child = exec(cmd, (error, stdout, stderr) => {
            if (error) {
            }
        });

        pid = child.pid;
        child.stdout.on('data', (data) => {
            if(data.includes("plate")){
                let _data = data.split(",");
                try{
                    if(_data[1] !== null && _data[2] !== null && _data[2] >= 90){
                        if(jwt !== null){
                            var options = {
                                timeout: 10000,
                                time: true,
                                headers: {
                                    'Content-Type': 'application/json',
                                    'Authorization': `Bearer ${jwt}`
                                }
                            };
```

2. platequery.py

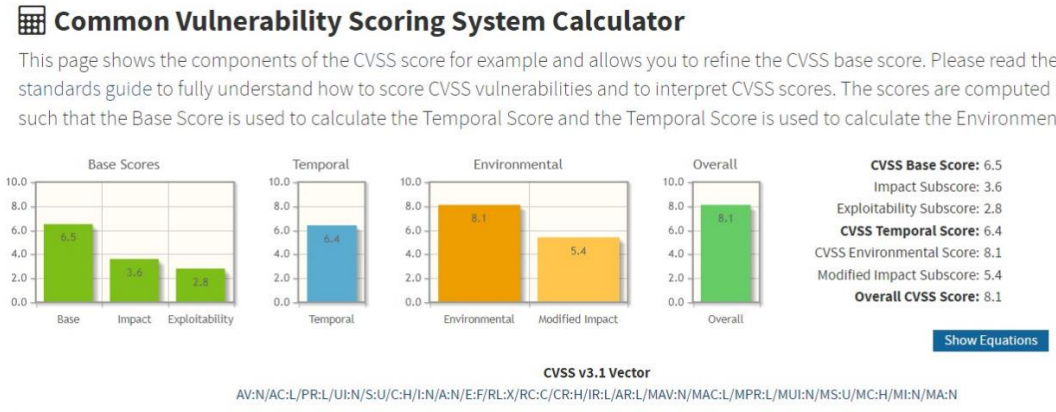
```
@plate_query.route("/get", methods=['GET'])
@token_required
def getPlate(valid_user):
    if valid_user is not None:

      plate_number = request.args.get('plate_number')

      if plate_number is None:
        value = {"result":{"status" : "fail",
                           "message":"Invailed plase number"}}
        return jsonify(value)

      result, value = valid_platenumber(plate_number)
      if not result:
        return jsonify(value)

      data = Plate.query.filter_by(plate=plate_number).first()
```

3. the token is the only access right to the API.
4. After logout of the client app, the token is still valid on the server side.
 - index.js : only clear client side`s token

```
const signOut = () => {
  sessionStorage.clear();
  setJwt(sessionStorage.getItem("key"));
}
```

Relative component / source code

CVSS Score	8.1	Severity	High



Consequence / Impact analysis

An attacker can acquire the plate information

Recommended mitigation

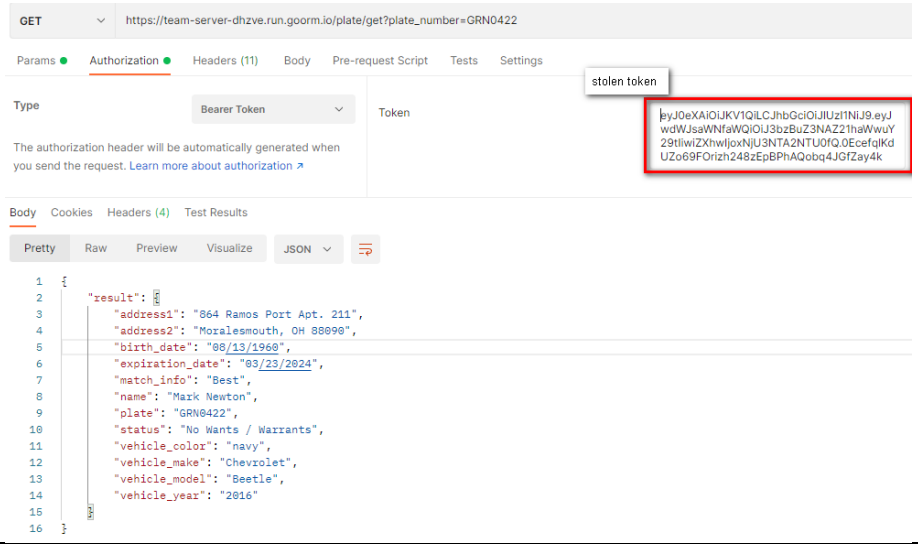
Using the PKI with SSL/TLS
Using JWT token revoke mechanism

Tools needed	Postman

Proof of concept / How to attack

1. Run Postman
2. make Headers with token & Params
3. send packet to server

   GET "http://team-server-dhzve.run.goorm.io/plate/get?plate_number=GRN0422"



1.2.19. V19 - There is no JWT token revoke mechanism

Description	[V19] There is no JWT token revoke mechanism		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/CODE] [REVIEW/DESIGN]	Exploit technique	[SNIFFING]
Vulnerabilities			

The expired time of the jwt token is as short as 45 minutes, but refresh token and logic procedure required to maintain a session do not exist.
therefore the user should repeatedly call the login GET method API, there is a risk of account information exposure.
"https://team-server-dhzve.run.goorm.io/signin?username=test2@gmail.com&password=Asdf!234&otp=932468
1.usermanagement.js
- only jwt token available

```
@user_management.route("/signin", methods=['GET','POST'])
def signin():

  if request.method == 'GET':
    form_user = request.args.get("username")
    form_pass = request.args.get("password")
    form_otp = request.args.get("otp")
  else:
    form_user = request.form.get("username")
    form_pass = request.form.get("password")
    form_otp = request.form.get("otp")

  if form_user is None or form_pass is None or form_otp is None:
    return {"result":{
              "status" : "fail",
              "message": "Invalid signin info."
            }}

  result, value = valid_username(form_user)
  if not result:
    return jsonify(value)

  data = User.query.filter_by(username=form_user).first()   # ID 조회Query 실행

  if data is not None:
    if data.password == form_pass:
      totp_instance = pyotp.TOTP(data.otp)
      valid = totp_instance.verify(form_otp)
      if valid:
        data.otpfcnt = 0
        data.passwordfcnt = 0
        db.session.commit()
        token = jwt.encode({'public_id': data.username, 'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=45)}, config._JWT_SECRET_KEY_, "HS256")
        value = {"result":{
          "status" : "success",
          "token" : token,
          "message": ""
        }}
```

2. platequery.py

```
@user_management.route("/signin", methods=['GET','POST'])
def signin():

  if request.method == 'GET':
    form_user = request.args.get("username")
    form_pass = request.args.get("password")
    form_otp = request.args.get("otp")
  else:
    form_user = request.form.get("username")
    form_pass = request.form.get("password")
    form_otp = request.form.get("otp")

  if form_user is None or form_pass is None or form_otp is None:
    return {"result":{
              "status" : "fail",
              "message": "Invalid signin info."
            }}

  result, value = valid_username(form_user)
  if not result:
    return jsonify(value)

  data = User.query.filter_by(username=form_user).first()   # ID 조회Query 실행

  if data is not None:
    if data.password == form_pass:
      totp_instance = pyotp.TOTP(data.otp)
      valid = totp_instance.verify(form_otp)
      if valid:
        data.otpfcnt = 0
        data.passwordfcnt = 0
        db.session.commit()
        token = jwt.encode({'public_id': data.username, 'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=45)}, config._JWT_SECRET_KEY_, "HS256")
        value = {"result":{
          "status" : "success",
          "token" : token,
          "message": ""
        }}
```

3.utils.py - only jwt token valid check.

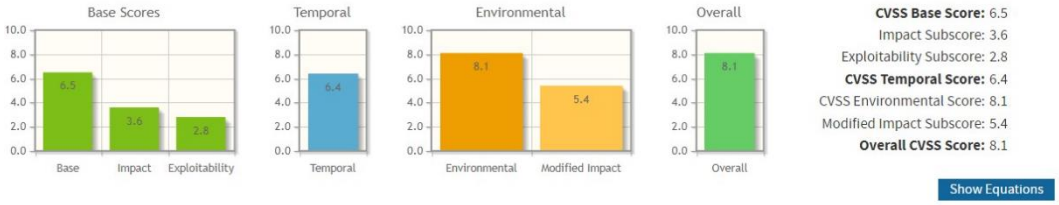
```python
def token_required(f):
  @wraps(f)
  def decorator(*args, **kwargs):
    auth = request.headers.get("Authorization")
    if not auth:
      value = {"result":{
        "status" : "fail",
        "message": "Valid token is Missing"
      }}
      return jsonify(value)

    token = None
    if auth.startswith("Bearer "):
      split = auth.split("Bearer")
      token = split[1].strip()
    if not token:
      value = {"result":{
        "status" : "fail",
        "message": "Valid token is Missing"
      }}
      return jsonify(value)

    try:
      data = jwt.decode(token, config._JWT_SECRET_KEY_, algorithms=["HS256"])
      valid_user = User.query.filter_by(username=data['public_id']).first()
    except:
      value = {"result":{
        "status" : "fail",
        "message": "Invalid token"
      }}
      return jsonify(value)

    return f(valid_user, *args, **kwargs)

  return decorator
```

Relative component / source code

CVSS Score	8.1	Severity	High

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS v3.1 Vector

AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/E:F/RL:X/RC:C/CR:H/IR:L/AR:L/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:H/MI:N/MA:N

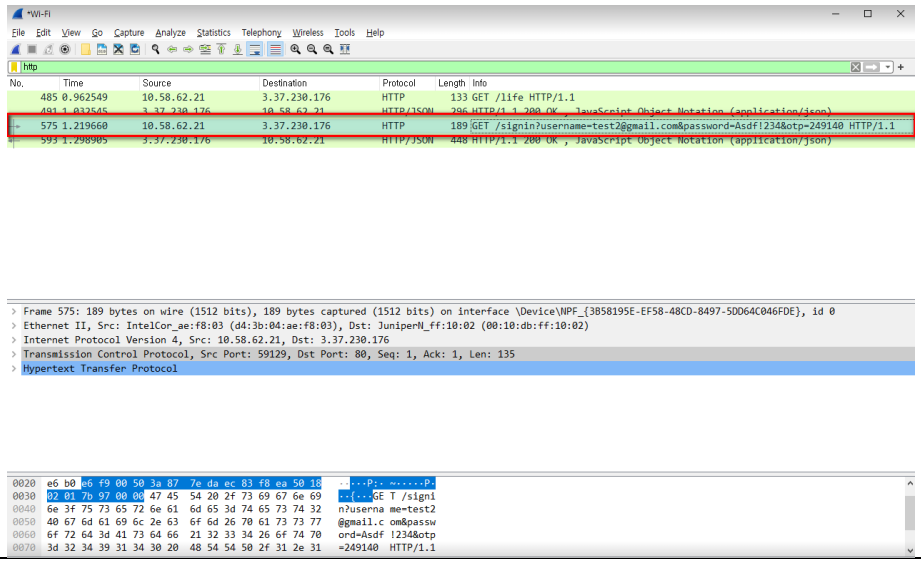
Consequence / Impact analysis
An attacker can steal general user credentials such as user name, password, otp, token and use its credentials to acquire the plate information

Recommended mitigation
1. Using JWT token revoke mechanism 2. Use “POST” method instead of “GET” method in login process.

Tools needed	Wireshark

Proof of concept / How to attack

sniffing


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
485 0.962549 10.58.62.21 3.37.230.176 HTTP 133 GET /life HTTP/1.1
575 1.219660 10.58.62.21 3.37.230.176 HTTP 189 GET /signin?username=test2@gmail.com&password=Asdf!234&otp=249140 HTTP/1.1
593 1.298905 3.37.230.176 10.58.62.21 HTTP/JSON 448 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
Frame 575: 189 bytes on wire (1512 bits), 189 bytes captured (1512 bits) on interface \Device\NPF_{3B58195E-EF58-48CD-8497-5DD64C046FDE}, id 0
Ethernet II, Src: IntelCor_ae:f8:03 (d4:3b:04:ae:f8:03), Dst: JuniperN_ff:10:02 (00:10:db:ff:10:02)
Internet Protocol Version 4, Src: 10.58.62.21, Dst: 3.37.230.176
Transmission Control Protocol, Src Port: 59129, Dst Port: 80, Seq: 1, Ack: 1, Len: 135
Hypertext Transfer Protocol
0020 e6 b0 e6 f9 00 50 3a 87 7e da ec 83 f8 ea 50 18
0030 02 01 7b 97 00 00 47 45 54 20 2f 73 69 67 6e 69
0040 6e 3f 75 73 65 72 6e 61 6d 65 3d 74 65 73 74 32
0050 40 67 6d 61 69 6c 2e 63 6f 6d 26 70 61 73 73 77
0060 6f 72 64 3d 41 73 64 66 21 32 33 34 26 6f 74 70
0070 3d 32 34 39 31 34 30 20 48 54 54 50 2f 31 2e 31

1.2.20. V20 - Actual account(test@gmail.com) is entered as an example in the Email Address field.

Description	[V20] actual account(test@gmail.com) is entered as an example in the Email Address field.		
CIA	[CONFIDENTIALITY]	Attack vector	[CLIENT/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[NOSPECIFIED]

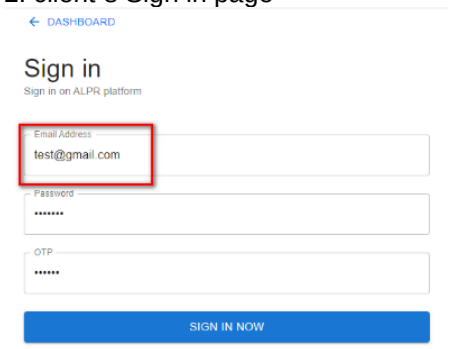
Vulnerabilities

In the "Sign in" page of the client, an actual account(test@gmail.com) is entered as an example in the Email Address field.
There is a risk that the account may be hijacked.

1.login.js

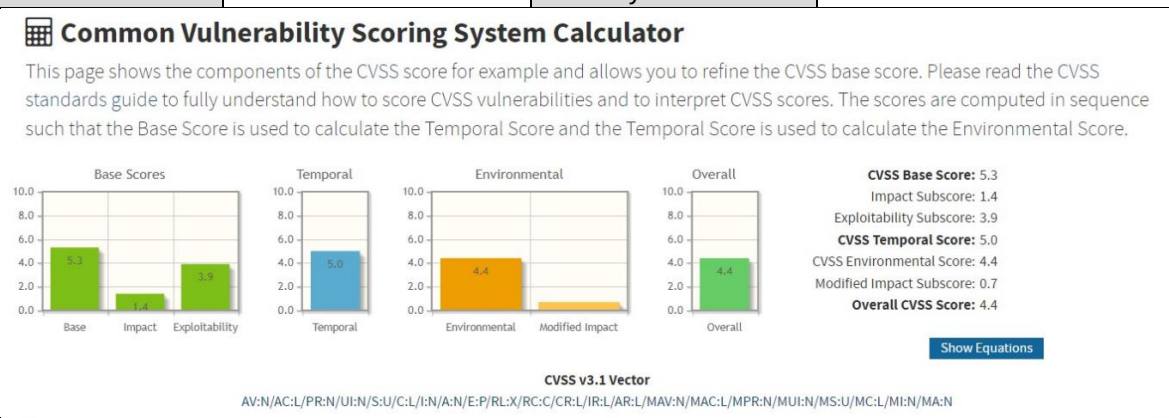
```
function Login(props) {
    const navigate = useNavigate();
    const onCancel = () =>{
        navigate(-1);
    }

  const formik = useFormik({
    initialValues: {
      email: 'test@gmail.com',
      password: 'test123',
      otp: '123456'
    },
    validationSchema: Yup.object({
    email: Yup
```

2. client`s Sign in page



Relative component / source code

CVSS Score	4.4	Severity	Medium



Consequence / Impact analysis

An attacker can try to steal “test@gamil.com” account using brute force attacks.
Recommended mitigation
Don't show real accounts as examples in the user interface.
Tools needed
Proof of concept / How to attack

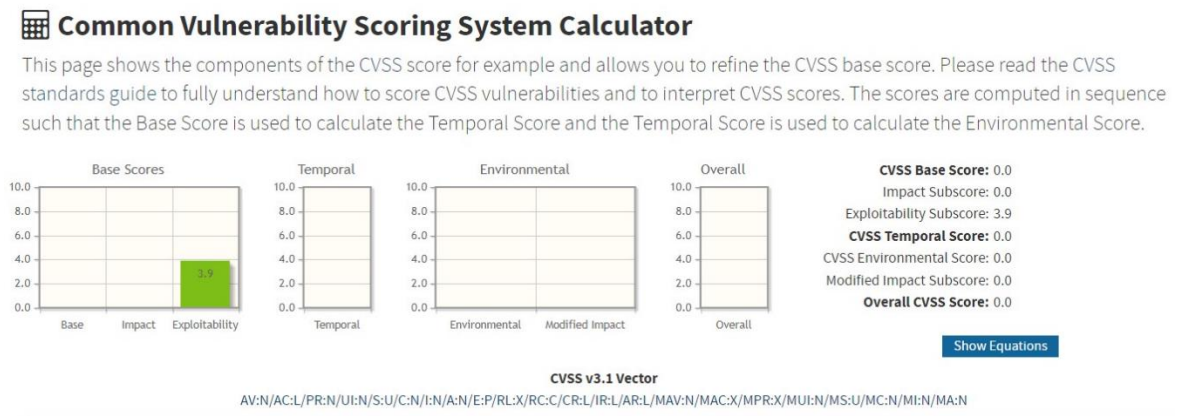
we retrieve the account information and to confirm that "test@gmail.com" actually exists.

User Log

Username	Query Per Second	Best Match Count	Partial Match Count
lg2.3team@gmail.com	0.0	0	0
test@gmail.com	1.0	24	1
test1@gmail.com	2.0	13	0
test2@gmail.com	2.0	7	0
test3@gmail.com	1.0	56	2
test4@gmail.com	0.0	0	0
wo0ngs@gmail.com	1.0	9	0

1.2.21. V21 - no log records for login with a general account on the server side.

Description	[V21] no log records for login with a general account on the server side.		
CIA	[CONFIDENTIALITY]	Attack vector	[SERVER/ACCESS]
Approach	[REVIEW/CODE]	Exploit technique	[NOSPECIFIED]
Vulnerabilities			
no log records for login with a general account on the server side. only a token is generated and delivered to the client.			
Relative component / source code			
CVSS Score	0.0	Severity	None



Consequence / Impact analysis
An attacker accesses the server with the stolen account, but the server does not record. So, it is impossible to track the attacker.
Recommended mitigation
Logging

Tools needed	NA

Proof of concept / How to attack

usermanagement.py - no log

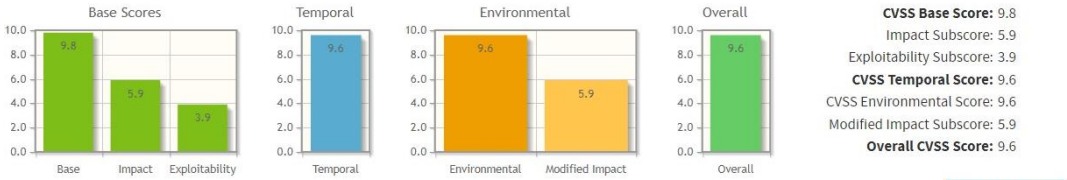
```
@user_management.route("/signin", methods=['GET','POST'])
def signin():

  if request.method == 'GET':
    form_user = request.args.get("username")
    form_pass = request.args.get("password")
    form_otp = request.args.get("otp")
  else:
    form_user = request.form.get("username")
    form_pass = request.form.get("password")
    form_otp = request.form.get("otp")

  if form_user is None or form_pass is None or form_otp is None:
    return {"result":{
              "status" : "fail",
              "message": "Invalid signin info."
            }}

  result, value = valid_username(form_user)
  if not result:
    return jsonify(value)

  data = User.query.filter_by(username=form_user).first()   # ID 조회Query 실행

  if data is not None:
    if data.password == form_pass:
      totp_instance = pyotp.TOTP(data.otp)
      valid = totp_instance.verify(form_otp)
      if valid:
        data.otpfcnt = 0
        data.passwordfcnt = 0
        db.session.commit()
        token = jwt.encode({'public_id': data.username, 'exp': datetime.datetime.utcnow() + datetime.timedelta(minutes=45)}, config._JWT_SECRET_KEY_, "HS256")
        value = {"result":{
          "status" : "success",
          "token" : token,
          "message": ""
        }}
```

1.2.22. V22 - Remote server spoofing

Description	[V22] Remote server spoofing		
CIA	[Availability] [Confidentiality]	Attack vector	[NETWORK]
Approach	[REVIEW/DESIGN]	Exploit technique	[SPOOFING]
Vulnerabilities			
1. Local server in client side and remote server do not proceed with mutual authentication. 2. The remote server is implemented with http, but Goorm hosting service(COTS) supports https(TLS) when packet is came into 443 port. And convert to http packet because remote server in container is bounded at 80 port(HTTP). It means that client authentication is omitted 3. Local server uses HTTPS/TLS1.2, but dose not authenticate server's certificate, just verify root CA of server certificate if this certificate is issued from authorized agency. It means that there is no authentication of server identification and encrypt packet on external network only. 4. If attacker can redirect remote server URL to attacker's URL/IP, server spoofing could be done without any additional procedure			
CVSS Score	9.6	Severity	Critical

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS v3.1 Vector

AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RL:X/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:L/MPR:N/MUI:N/MS:U/MC:H/MI:H/MA:H

Consequence / Impact analysis
Spoofing server can occur stealing sensitive data (e.g. user credential) and it is sold to illegal group for making money or used for following attack. Secondly, officer cannot execute law enforcement. It means that ALPR service is completely down (all stop) as long as DNS spoofing is worked. It damages to police office(government) reputation critically.
Recommended mitigation
Apply mutual authentication procedure between server and client like TLS(above 1.2)

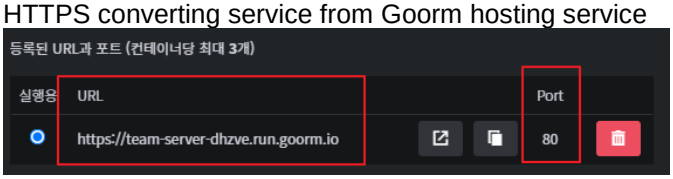
Tools needed	Goorm hosting service

Relative component / source code
Local Server in client side (there is no additional authentication procedure or option, just request with https)

client/web/src/images/server.js

```
173 app.post('/login', function(req, res){
174     let email = req.query.email;
175     let password = req.query.password;
176     let otp = req.query.otp;
177
178     const url = host +"/signin?username="+email+"&password="+password+"&otp="+otp;
179     const request = https.request(url, (response) => {
180         let data = '';
181         response.on('data', (chunk) => {
182             data = data + chunk.toString();
183         });
```

HTTPS converting service from Goorm hosting service



HTTP bind in remote server(server.py)

```python
app = Flask(__name__)

if __name__ == '__main__':
  app = Flask(__name__)
  app.config['DEBUG'] = True
  app.config['SQLALCHEMY_DATABASE_URI'] = config._DB_PATH_
  app.config["SECRET_KEY"] = config._SESSION_SECRET_KEY_

  db.init_app(app)
  if not os.path.isfile('./server.db'):
    with app.app_context():
      db.create_all()
      createdb()

  app.register_blueprint(usermanagement.user_management)
  app.register_blueprint(platequery.plate_query)

  app.run(host='0.0.0.0', port=80)
```

Proof of concept / How to attack

Constraints:

1. Attacker has capability for DNS spoofing in external network (DNS cache poisoning)
2. Attacker should have cert which is not self-signed
   → we use goorm hosting service for certificate which issued from root CA
   (real attackers prepare their site with root CA and IP)
   → But It works with container, cannot access directly remote server in ours container with URL or IP
   → So verifying server spoofing, we test with URL modification in code like below,

```
//const host = "https://team-server-dhzve.run.goorm.io";
const host = "https://con-t-ythyc.run.goorm.io";
```

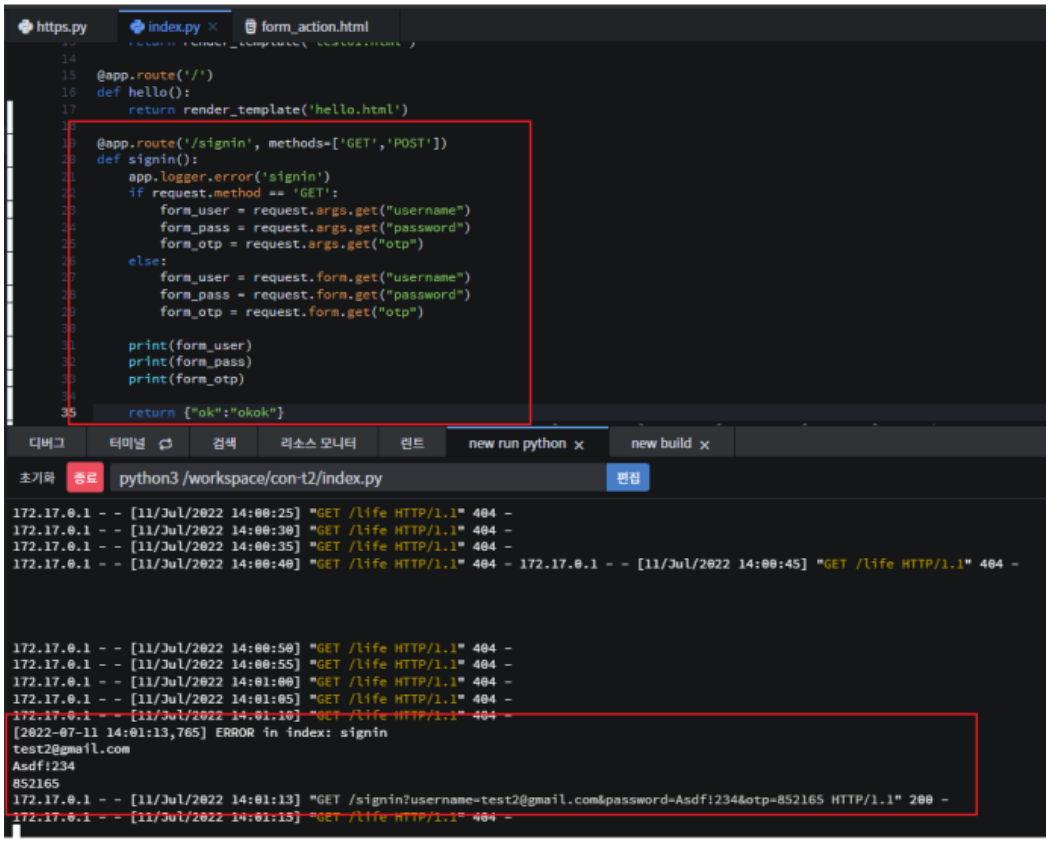
   (below is attacker's)

   ⇨ Assume that DNS spoofing is applied for team3's server URL by modifying URL in source code

Procedure:

1. make fake server with python flask and run in goorm hosting service
2. connect localhost:3000 and try to login
3. server is spoofed
4. check if user credential is received or not in attacker's server
5. check if ALPR service doses not worked well

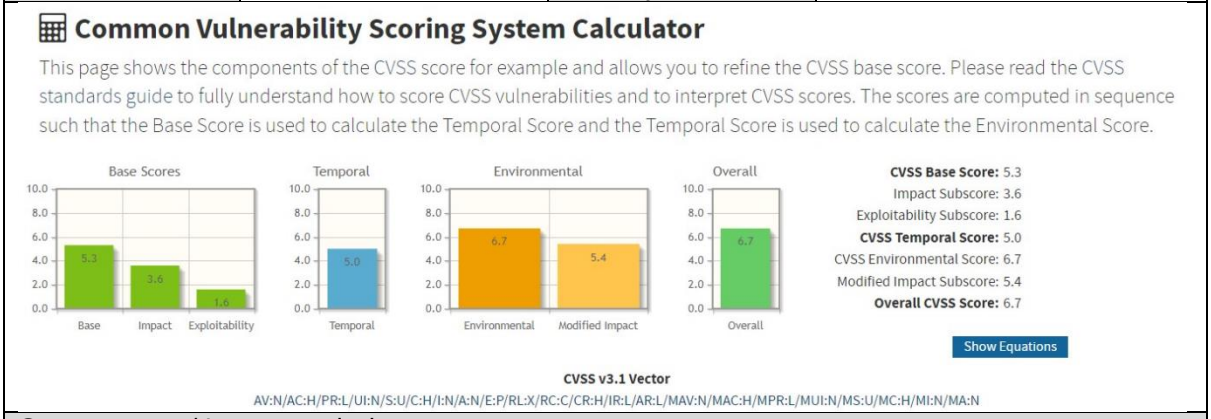
faker server's impl - signin method : sniffing user credential



without processing signin, we just sniffing
http get method and user credential is passed

1.2.23. V23 - phishing client web site

Description	[V23] phishing client web site		
CIA	[Availability] [Confidentiality]	Attack vector	[CLIENT/PRIVILEGED]
Approach	[REVIEW/DESIGN]	Exploit technique	[SPOOFING]
Vulnerabilities			
1. S/W structurally, on client side, there are UI app and local host server and there is no any authentication of client webapp. It means that there isn't any protection on client. 2. Attacker can spoof client UI web for phishing and steal sensitive data or prevent officer executing law enforcement.			
CVSS Score	6.7	Severity	Medium



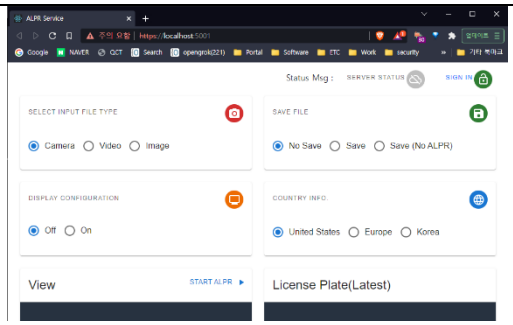
Consequence / Impact analysis
Phishing client can occur stealing sensitive data (e.g. user credential) and it is sold to illegal group for making money or used for following attack. Secondly, officer cannot execute law enforcement. In other words, officer connect to fake client, try to login, start APLR service. Everything is worked well visually. But fake client does not send any license plate data to server. So, ALPR service is not worked normally.
Recommended mitigation
Run client webapp in server side or apply authentication procedure with web server
Tools needed: socat(for port forwarding), yarn(running webapp)
Relative component / source code
Client UI Webapp (design problem)
Proof of concept / How to attack

Constraints:

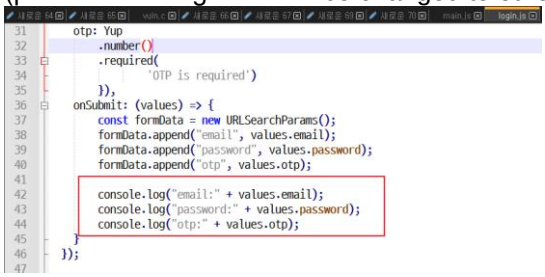
- client PC has received any malware which attacker injects for forwarding port and running attacker's fake webapp

Procedure:

1. Make fake client webapp and run with port 5001 like below



2. modify login button function
   (print console log → It can be changed to other method to transfer user credential)

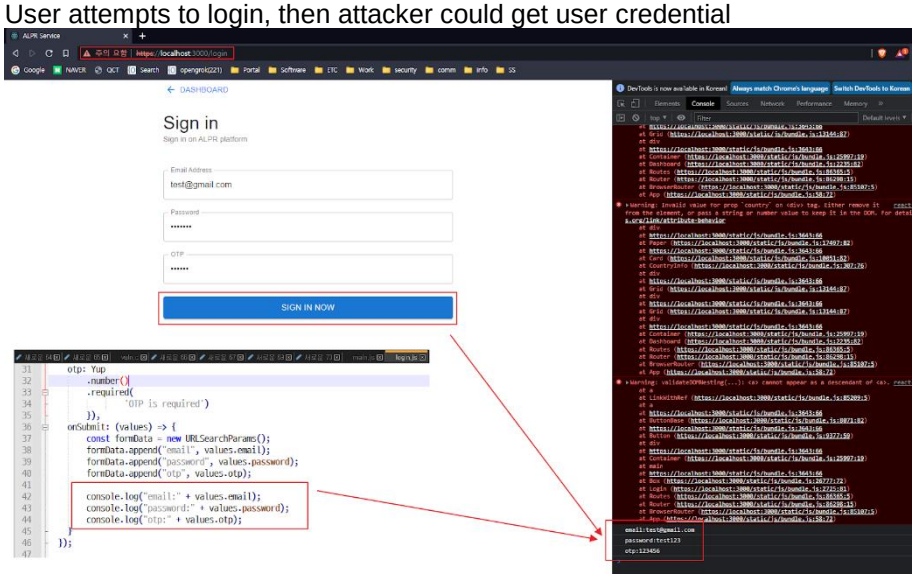


```
    otp: Yup
        .number()
        .required(
            'OTP is required')
        }),
    onSubmit: (values) => {
        const formData = new URLSearchParams();
        formData.append("email", values.email);
        formData.append("password", values.password);
        formData.append("otp", values.otp);

        console.log("email:" + values.email);
        console.log("password:" + values.password);
        console.log("otp:" + values.otp);
    }
});
```

3. apply port forwarding 3000 → 5001 with socat
   insert command like below, it forward 3000 port to 5001
   > socat tcp-listen:3000,reuseaddr,fork tcp:127.0.0.1:5001

```
PS C:¥Users¥User¥Downloads¥socat-1.7.3.2-1-x86_64¥socat-1.7.3.2-1-x86_64> ./¥socat tcp-listen:3000,reuseaddr,fork tcp:12
7.0.0.1:5001
2022/07/09 17:10:39 socat[24704] E write(6, 0x600042e50, 24): Broken pipe
```

4. officer connect to localhost:3000(ALPR web service UI URI)
5. It redirects the site to localhost:5001, officer sees the same graphical web page, but it is attacker's phishing site!
6. User attempts to login, then attacker could get user credential



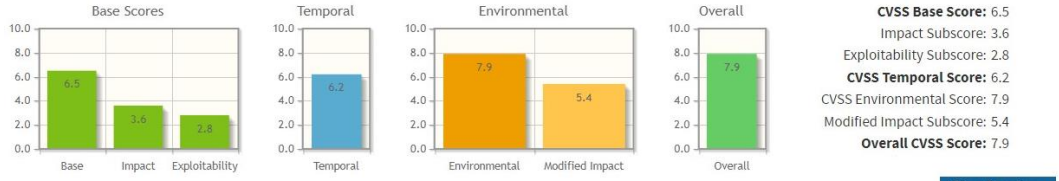
Although the officer connects to the localhost:3000, he connects to the attacker's site (localhost:5001). and his credential is stolen by attacker. And finally, he does not execute law enforcement because client is fake site

1.2.24. V24 - Stolen token, spoofing as valid user, sensitive information could be sniffed

Description	[V24] Stolen token, spoofing as valid user, sensitive information could be sniffed		
CIA	[CONFIDENTIALITY]	Attack vector	[NETWORK]
Approach	[REVIEW/CODE] [REVIEW/DESIGN]	Exploit technique	[SNIFFING]
Vulnerabilities			
Stolen token, spoofing as valid user, sensitive information could be sniffed			
Relative component / source code			
CVSS Score	7.9	Severity	High

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS Base Score: 6.5
Impact Subscore: 3.6
Exploitability Subscore: 2.8
CVSS Temporal Score: 6.2
CVSS Environmental Score: 7.9
Modified Impact Subscore: 5.4
Overall CVSS Score: 7.9

Show Equations

CVSS v3.1 Vector

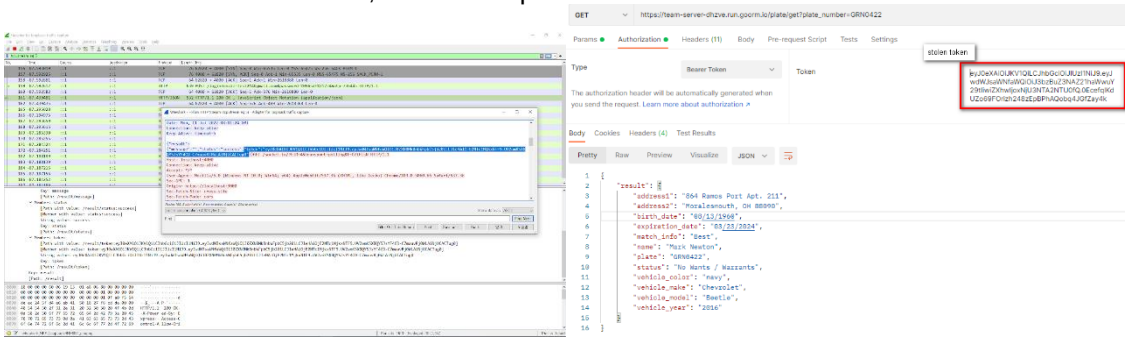
AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N/E:P/RL:X/RC:C/CR:H/IR:L/AR:L/MAV:N/MAC:L/MPR:N/MUI:R/MS:U/MC:H/MI:N/MA:N

Consequence / Impact analysis
An attacker can steal general user credentials such as user name, password, otp seed, token and use this credentials to acquire the plate information
Recommended mitigation
Provide the client interface in the form of WebApp. So, Eliminate exposed IPC channels by operating in one process.

Tools needed	Wireshark

Proof of concept / How to attack

1. packet capture start with localhost
2. connect webpage to https://localhost:3000/
3. proceed login
4. check token in wireshark packet
5. Insert token to header, could be impersonate as validated



1.2.25. V25 - The external server provided by goorm does not support mutual authentication using tls based on pki.

Description	[V25] The external server provided by goorm does not support mutual authentication using tls based on pki.		
CIA	[CONFIDENTIALITY] [INTEGRITY]	Attack vector	[NETWORK]
Approach	[REVIEW/CODE]	Exploit technique	[SNIFFING]
Vulnerabilities			

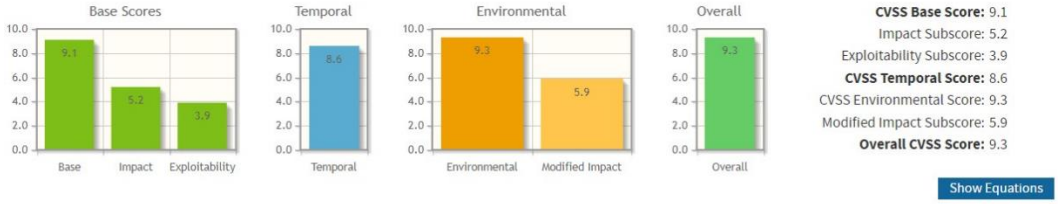
server.py
- does not implement mutual authentication using tls based on pki.

```
app = Flask(__name__)

if __name__ == '__main__':
  app = Flask(__name__)
  app.config['DEBUG'] = True
  app.config['SQLALCHEMY_DATABASE_URI'] = config._DB_PATH_
  app.config["SECRET_KEY"] = config._SESSION_SECRET_KEY_

  db.init_app(app)
  if not os.path.isfile('./server.db'):
    with app.app_context():
      db.create_all()
      createdb()

  app.register_blueprint(usermanagement.user_management)
  app.register_blueprint(platequery.plate_query)

  app.run(host='0.0.0.0', port=80)
```

Relative component / source code			
CVSS Score	9.3	Severity	Critical

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



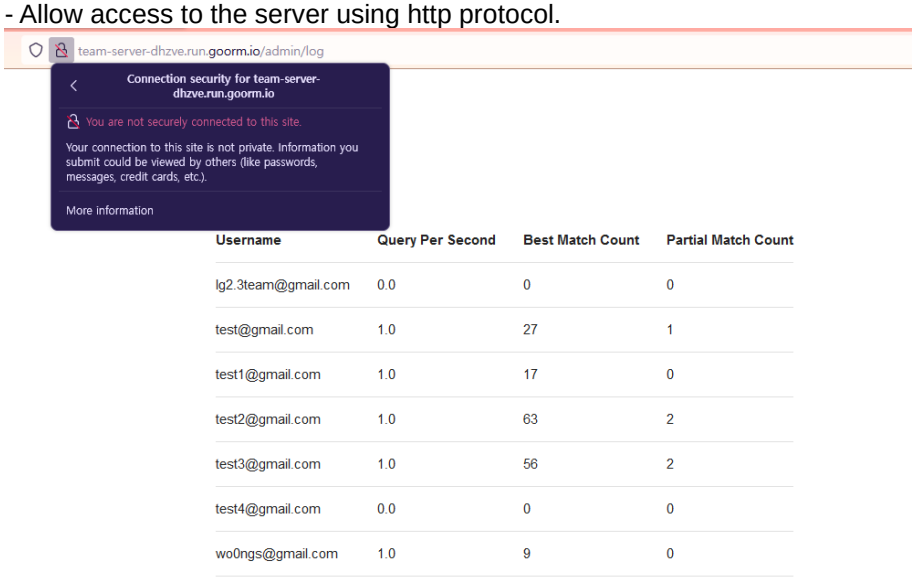
CVSS v3.1 Vector

AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N/E:P/RL:X/RC:C/CR:H/IR:H/AR:L/MAV:N/MAC:L/MPR:N/MUI:N/MS:U/MC:H/MI:H/MA:N

Consequence / Impact analysis
An attacker can steal admin or general account information by fake remote server using DNS spoofing An attacker can sniff all information by changing HTTPS to HTTP through the network
Recommended mitigation
Using the PKI with SSL/TLS

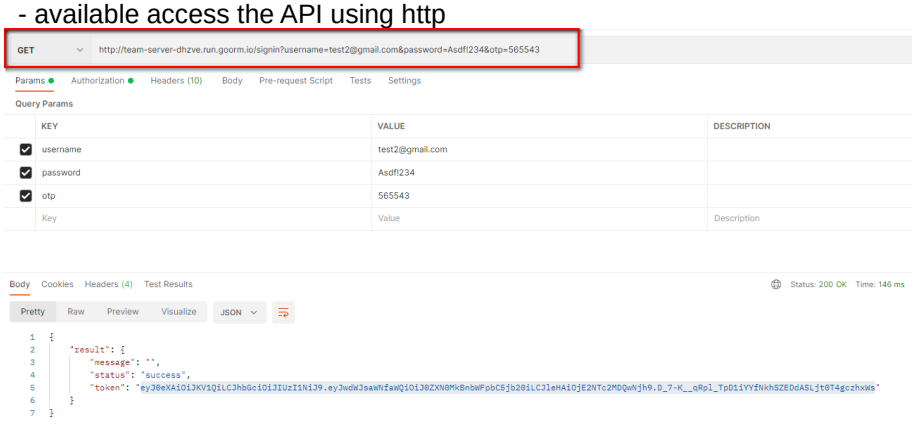
Tools needed	Wireshark, Unprivileged new browser
Proof of concept / How to attack	

- Allow access to the server using http protocol.



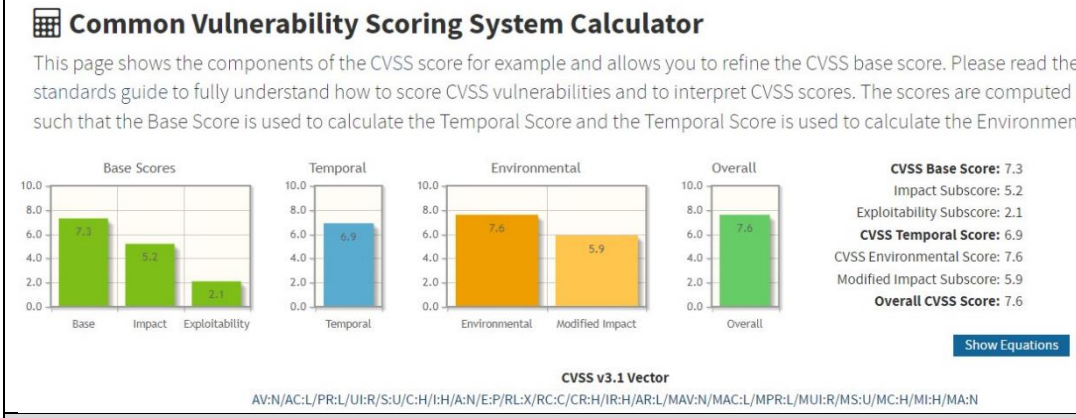
Username	Query Per Second	Best Match Count	Partial Match Count
lg2.3team@gmail.com	0.0	0	0
test@gmail.com	1.0	27	1
test1@gmail.com	1.0	17	0
test2@gmail.com	1.0	63	2
test3@gmail.com	1.0	56	2
test4@gmail.com	0.0	0	0
wo0ngs@gmail.com	1.0	9	0

- available access the API using http



1.2.26. V26 - Available to change https to http by tempering the client's server.js file.

Description	[V26] Available to change https to http by tempering the client's server.js file. because mutual authentication with an external server is not performed.		
CIA	[CONFIDENTIALITY] [INTEGRITY]	Attack vector	[CLIENT/SOURCE]
Approach	[REVIEW/CODE]	Exploit technique	[TAMPERING]
Vulnerabilities			
Available to change https to http by tempering the client's server.js file. because mutual authentication with an external server is not performed.			
Relative component / source code			
CVSS Score	7.6	Severity	High



Consequence / Impact analysis
An attacker can sniff all information by changing HTTPS to HTTP by tampering with the internal server of the client app. And using its information, an attacker can acquire the plate information.
Recommended mitigation
Using the PKI with SSL/TLS

Tools needed	Wireshark
Proof of concept / How to attack	

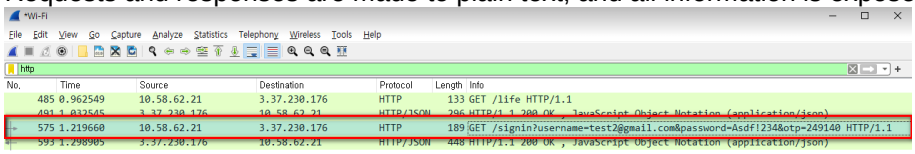
server.js

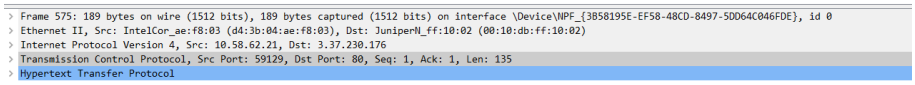
: tempering code

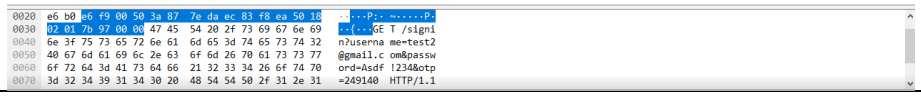
- const https = require('http');
- const host = http://team-server-dhzve.run.goorm.io;

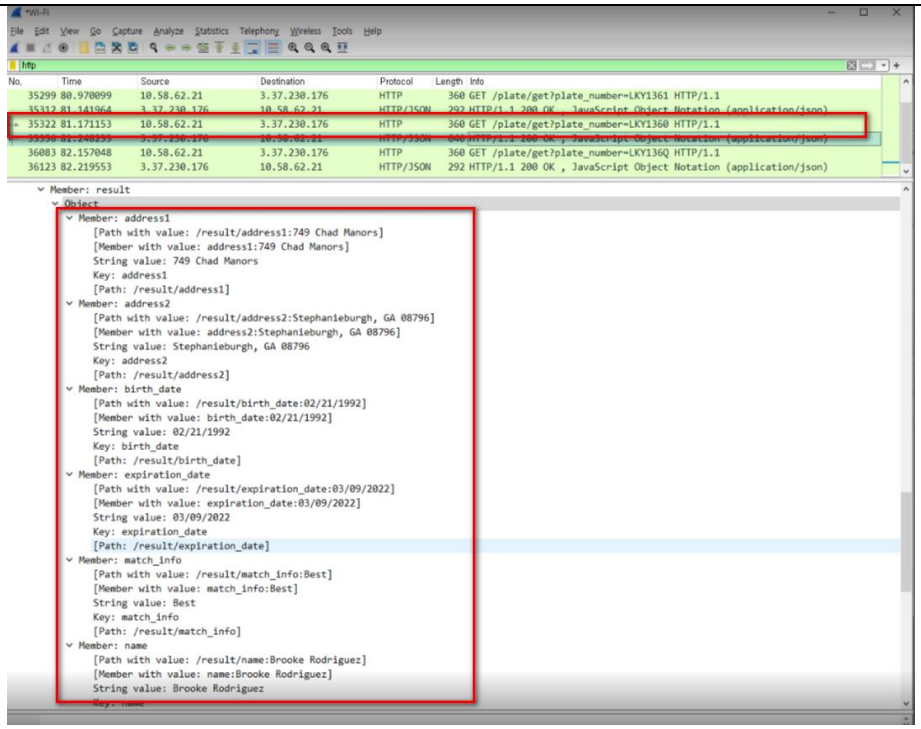
```
var path = require("path");
const https = require('http');
const http = require('http');

const ssl_config = require('./ssl-config');

const options = {
    key: ssl_config.privateKey,
    cert: ssl_config.certificate
}

// const server = https.createServer(options, app);
const server = http.createServer(app);
const cors = require('cors');
const io = require('socket.io')(server,{
    cors : {
        origin :"*",
        credentials :true
    }
});

app.use(cors());
app.use(express.urlencoded({
    extended: true
}))

const host = "http://team-server-dhzve.run.goorm.io";

var socketId = "";
io.on('connection', socket => {
```

Communication with an external server has been changed to http.
Requests and responses are made to plain text, and all information is exposed.







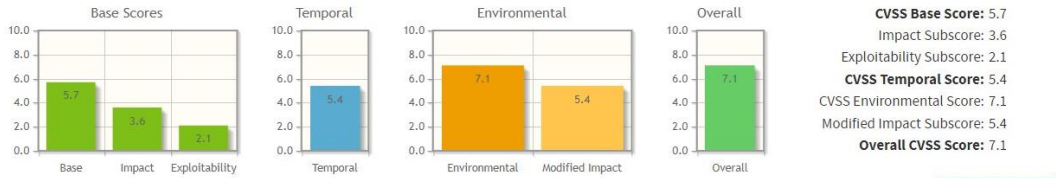

*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
35299 80.970099 10.58.62.21 3.37.230.176 HTTP 360 GET /plate/get?plate_number=LKY1361 HTTP/1.1
35312 81.141964 3.37.230.176 10.58.62.21 HTTP/JSON 292 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
35322 81.171153 10.58.62.21 3.37.230.176 HTTP 360 GET /plate/get?plate_number=LKY1360 HTTP/1.1
35330 81.240235 3.37.230.176 10.58.62.21 HTTP/JSON 640 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
36083 82.157048 10.58.62.21 3.37.230.176 HTTP 360 GET /plate/get?plate_number=LKY136Q HTTP/1.1
36123 82.219553 3.37.230.176 10.58.62.21 HTTP/JSON 292 HTTP/1.1 200 OK , JavaScript Object Notation (application/json)
Member: result
Object
Member: address1
[Path with value: /result/address1:749 Chad Manors]
[Member with value: address1:749 Chad Manors]
String value: 749 Chad Manors
Key: address1
[Path: /result/address1]
Member: address2
[Path with value: /result/address2:Stephanieburgh, GA 08796]
[Member with value: address2:Stephanieburgh, GA 08796]
String value: Stephanieburgh, GA 08796
Key: address2
[Path: /result/address2]
Member: birth_date
[Path with value: /result/birth_date:02/21/1992]
[Member with value: birth_date:02/21/1992]
String value: 02/21/1992
Key: birth_date
[Path: /result/birth_date]
Member: expiration_date
[Path with value: /result/expiration_date:03/09/2022]
[Member with value: expiration_date:03/09/2022]
String value: 03/09/2022
Key: expiration_date
[Path: /result/expiration_date]
Member: match_info
[Path with value: /result/match_info:Best]
[Member with value: match_info:Best]
String value: Best
Key: match_info
[Path: /result/match_info]
Member: name
[Path with value: /result/name:Brooke Rodriguez]
[Member with value: name:Brooke Rodriguez]
String value: Brooke Rodriguez

1.2.27. V27 - A TOCTOU attack is possible because there is a time gap between file selection and file playback.

Description	[V27] A TOCTOU attack is possible because there is a time gap between file selection and file playback.		
CIA	Integrity	Attack vector	[CLIENT/SOURCE]
Approach	[REVIEW/DESIGN]	Exploit technique	[TAMPERING]
Vulnerabilities			
A TOCTOU attack is possible because there is a time gap between file selection and file playback.			
Relative component / source code			
Source/client/web/src/images/playback files			
CVSS Score	7.1	Severity	High

Common Vulnerability Scoring System Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score.



CVSS Base Score: 5.7
Impact Subscore: 3.6
Exploitability Subscore: 2.1
CVSS Temporal Score: 5.4
CVSS Environmental Score: 7.1
Modified Impact Subscore: 5.4
Overall CVSS Score: 7.1

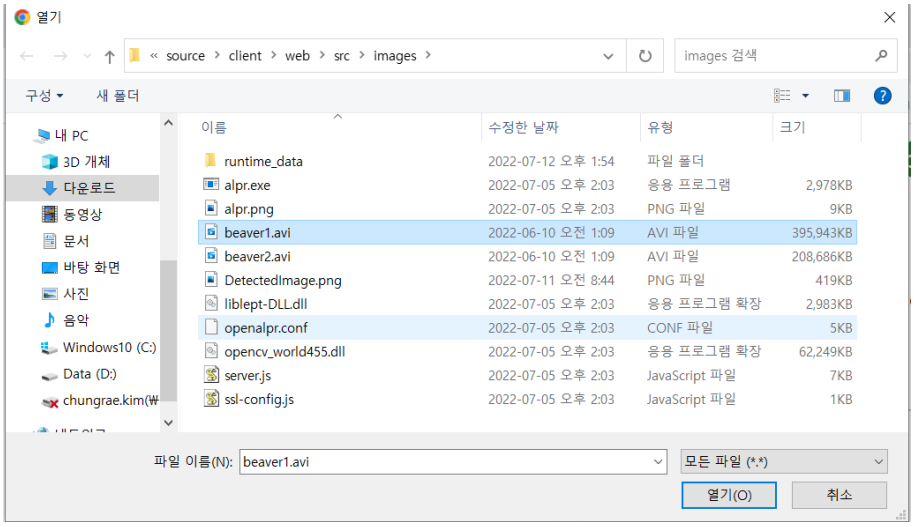
Show Equations

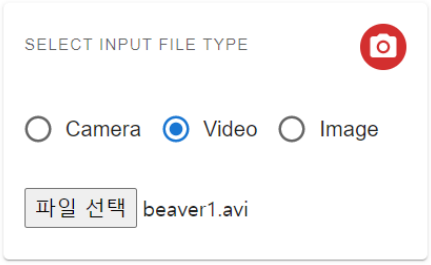
CVSS v3.1 Vector

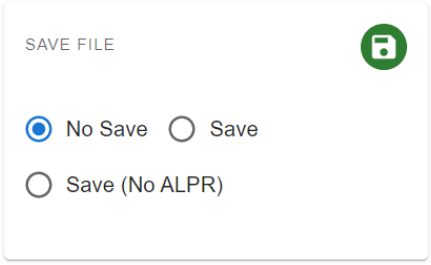
AV:N/AC:L/PR:L/UI:R/S:U/C:N/I:H/A:N/E:P/RL:X/RC:C/CR:L/IR:H/AR:L/MAV:N/MAC:L/MPR:L/MUI:R/MS:U/MC:N/MI:H/MA:N

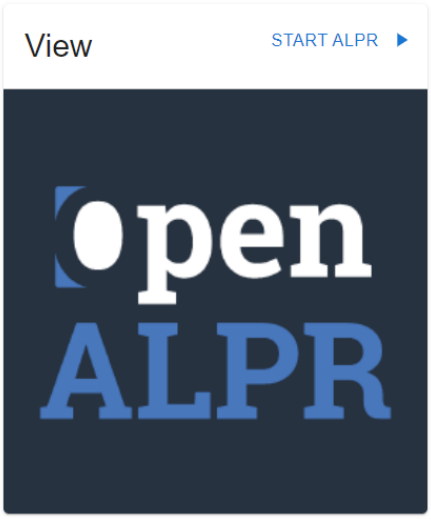
Consequence / Impact analysis
1. Attacker can change playback file, so a criminal driver can evade to be looked up
Recommended mitigation
1. Do not use an additional button to start ALPR 2. If choosing the file is done, run ALPR right now
Tools needed: NA
Proof of concept / How to attack

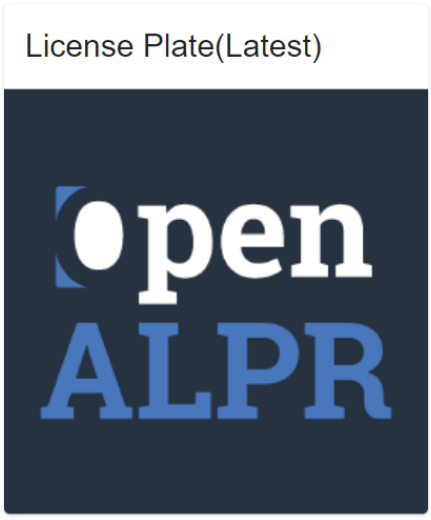
1. SELECT INPUT FILE TYPE to Video
2. Select a playback file (beaver1.avi) via “파일선택” button
3. if attacker change file beaver2.avi to beaver1.avi before pushing “START ALPR”, the officer will run "beaver2.avi" named "beaver1.avi" instead of original "beaver1.avi"











runtime_data	2022-07-12 오후 1:54	파일 폴더	
alpr.exe	2022-07-05 오후 2:03	응용 프로그램	2,978KB
alpr.png	2022-07-05 오후 2:03	PNG 파일	9KB
beaver1.avi	2022-06-10 오전 1:09	AVI 파일	395,943KB
beaver2.avi	2022-06-10 오전 1:09	AVI 파일	208,686KB
DetectedImage.png	2022-07-11 오전 8:44	PNG 파일	419KB
liblept-DLL.dll	2022-07-05 오후 2:03	응용 프로그램 확장	2,983KB
openalpr.conf	2022-07-05 오후 2:03	CONF 파일	5KB
opencv_world455.dll	2022-07-05 오후 2:03	응용 프로그램 확장	62,249KB
server.js	2022-07-05 오후 2:03	JavaScript 파일	7KB
ssl-config.js	2022-07-05 오후 2:03	JavaScript 파일	1KB

runtime_data	2022-07-12 오후 1:54	파일 폴더	
alpr.exe	2022-07-05 오후 2:03	응용 프로그램	2,978KB
alpr.png	2022-07-05 오후 2:03	PNG 파일	9KB
beaver1.avi	2022-06-10 오전 1:09	AVI 파일	208,686KB
beaver12.avi	2022-06-10 오전 1:09	AVI 파일	395,943KB
DetectedImage.png	2022-07-11 오전 8:44	PNG 파일	419KB
liblept-DLL.dll	2022-07-05 오후 2:03	응용 프로그램 확장	2,983KB
openalpr.conf	2022-07-05 오후 2:03	CONF 파일	5KB
opencv_world455.dll	2022-07-05 오후 2:03	응용 프로그램 확장	62,249KB
server.js	2022-07-05 오후 2:03	JavaScript 파일	7KB
ssl-config.js	2022-07-05 오후 2:03	JavaScript 파일	1KB

SELECT INPUT FILE TYPE

Camera Video Image

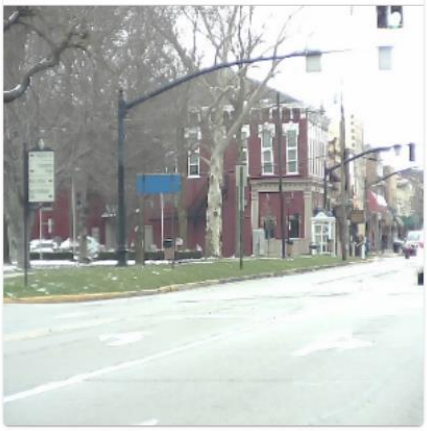
파일 선택 beaver1.avi

SAVE FILE

No Save Save

Save (No ALPR)

View STOP ALPR



License Plate(Latest)